GLOW 2024

Fakultas Informatika
School of Computing
Telkom University

Universiti Tenaga Nasional
The Energy University

CCI
College of Computing and Informatics

# DATA PREPROCESSING

Nur Laila Ab Ghani
Department of informatics
College of Computing and Informatics
Universiti Tenaga Nasional
Laila@uniten.edu.my

Adapted from Dr. Mahmud Dwi Sulistiyo Slides

# ABOUT ME

I am a lecturer in the Department of Informatics at Universiti Tenaga Nasional (UNITEN) in Malaysia. I hold an MSc. in Information Technology and Quantitative Sciences and a Bachelor of Computer Science (Hons.) First Class from Universiti Teknologi MARA, Malaysia. I am currently completing my PhD in Information Technology at Universiti Teknologi Petronas, Malaysia focusing on high-dimensional data streams and clustering techniques. My research interests include data analytics and machine learning, and I have led various projects funded by UNITEN, Malaysia's Ministry of Higher Education, and Tenaga Nasional Berhad. I am a member of the Association for Computing Machinery, the Association of Human-Computer Interaction Malaysia, and is recognized as a Professional Technologist by the Malaysia Board of Technologists.

GLOW 2024

# Outline

- Understanding the Data
- Data Preprocessing
- Feature Selection and Extraction
- Feature Analysis

Adapted from Dr. Mahmud Dwi Sulistiyo Slides

GLOW 2024

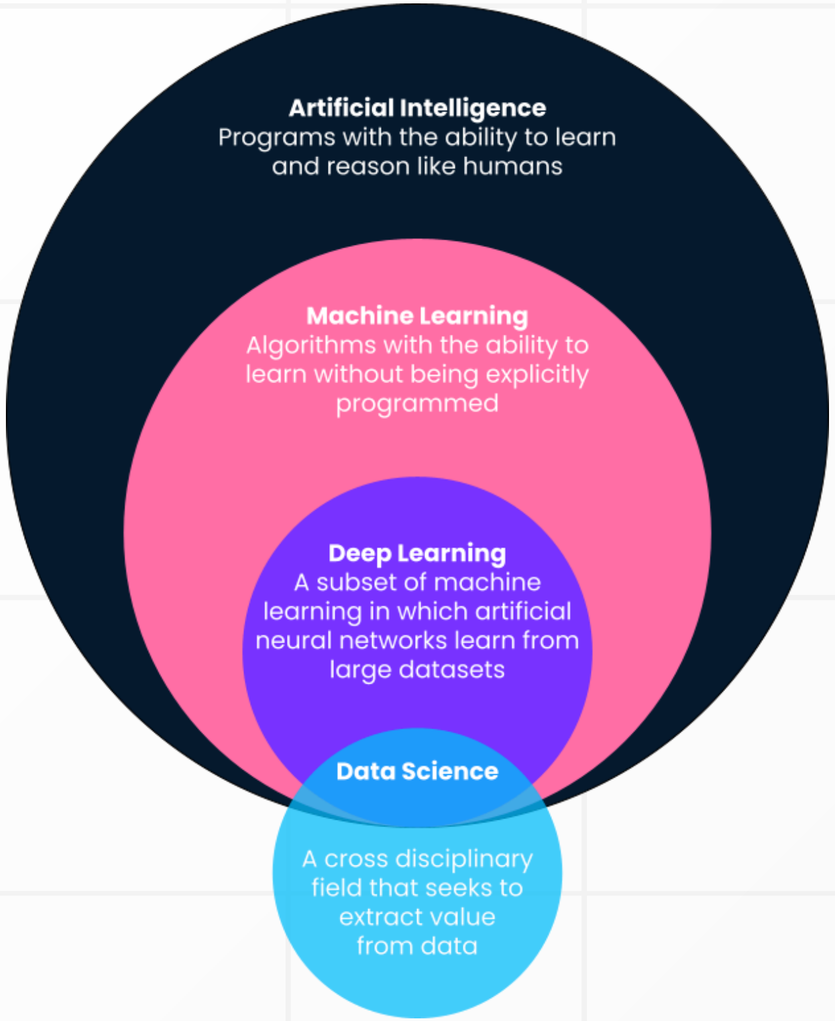
Fakultas Informatika
School of Computing
Telkom University

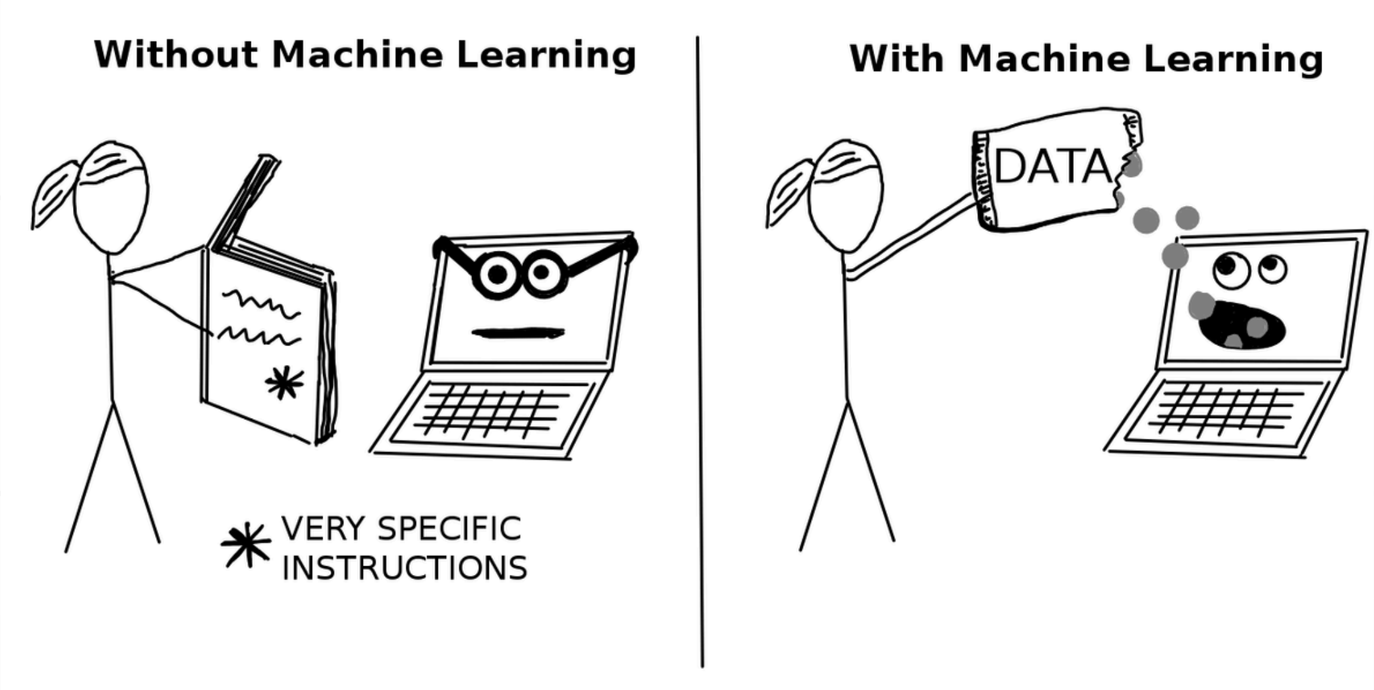
CCI
College of Computing and Informatics

# What is Machine Learning?

Adapted from Dr. Mahmud Dwi Sulistiyo Slides

# What is Machine Learning?



# Has this ever happened to you?

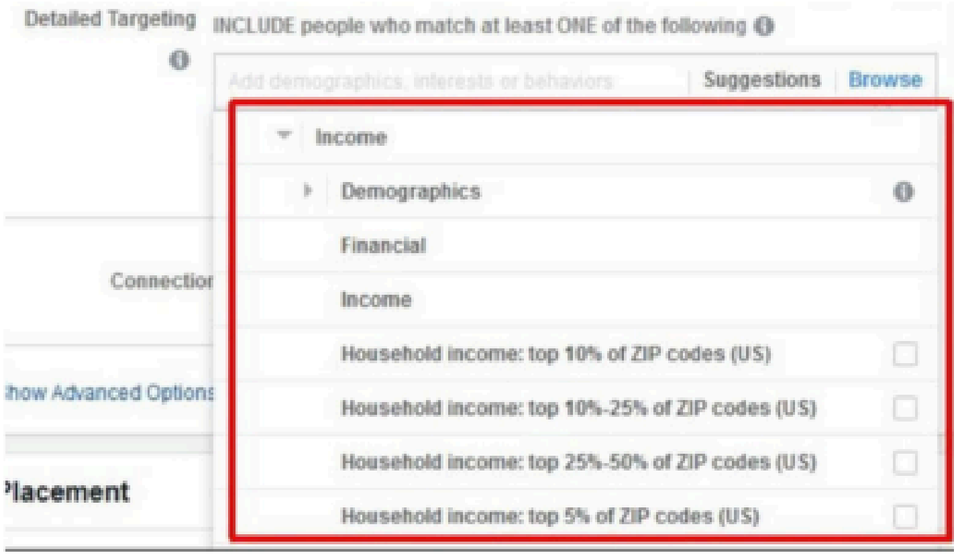
- You're thinking about something, maybe it's the new printer you wanted, or a skiing trip you were planning. Or perhaps thinking about starting a gym.
- Then BAM! You see an online Ad for the exact thing you wanted.

# How do those online giants like Google and Facebook know you so well?

- Unfortunately, it's not magic
- It's DATA
- And it's everywhere (including yours)
- Let's take a look at 5 super interesting examples of how Data Driven companies are changing the business landscape

# Online Advertising

- Many people still ask, how does Google and Facebook make money when everything is free?
- The answer is Advertising. These tech giants have become so good at targeting ads!
- Using their users data, both companies can target ads



# How Targeted Are Online Ads?

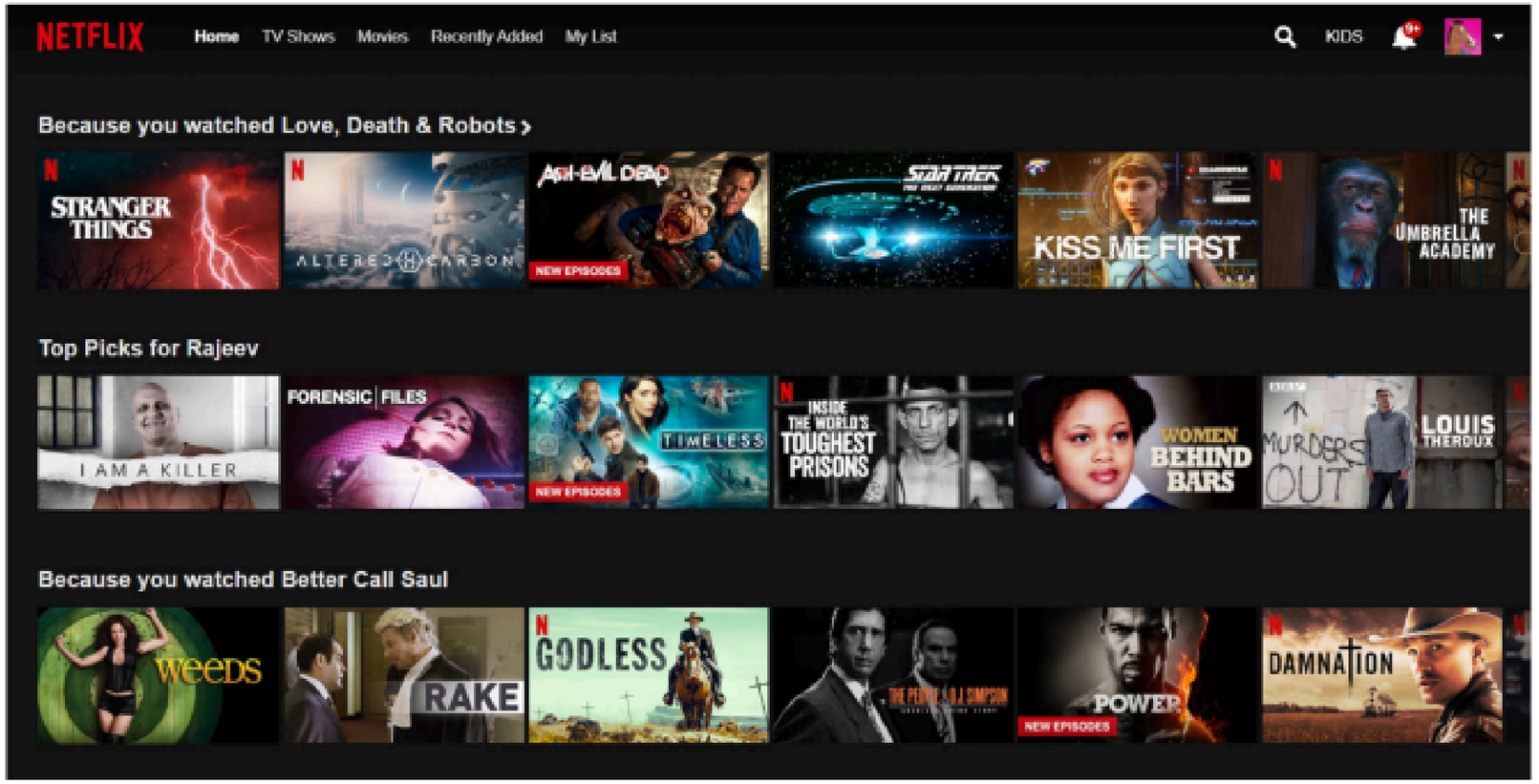
- **Let's say if you wanted to target grand parents, who live in one city who enjoy outdoor activities and barbeques and recently purchased a Mercedes Benz……well you could!**

- *"If you want to tailor a Facebook ad to a single user out of its universe of 2.2 billion, you could"*

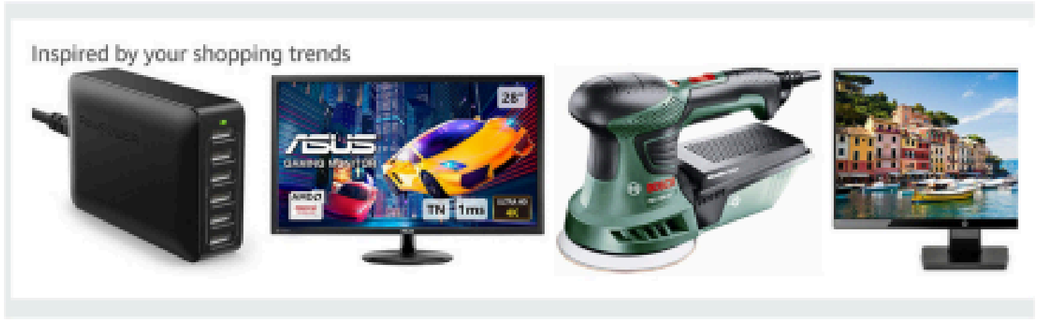
# Let's look at Uber

- Estimating that "Your driver will be in here in 6 minutes." to estimating fares, showing up surge prices and heat maps to the drivers on where to position themselves within the city.
- Uber relies heavily on data to provide and optimize their services

# Netflix's Recommendations



# Amazon's Recommendations



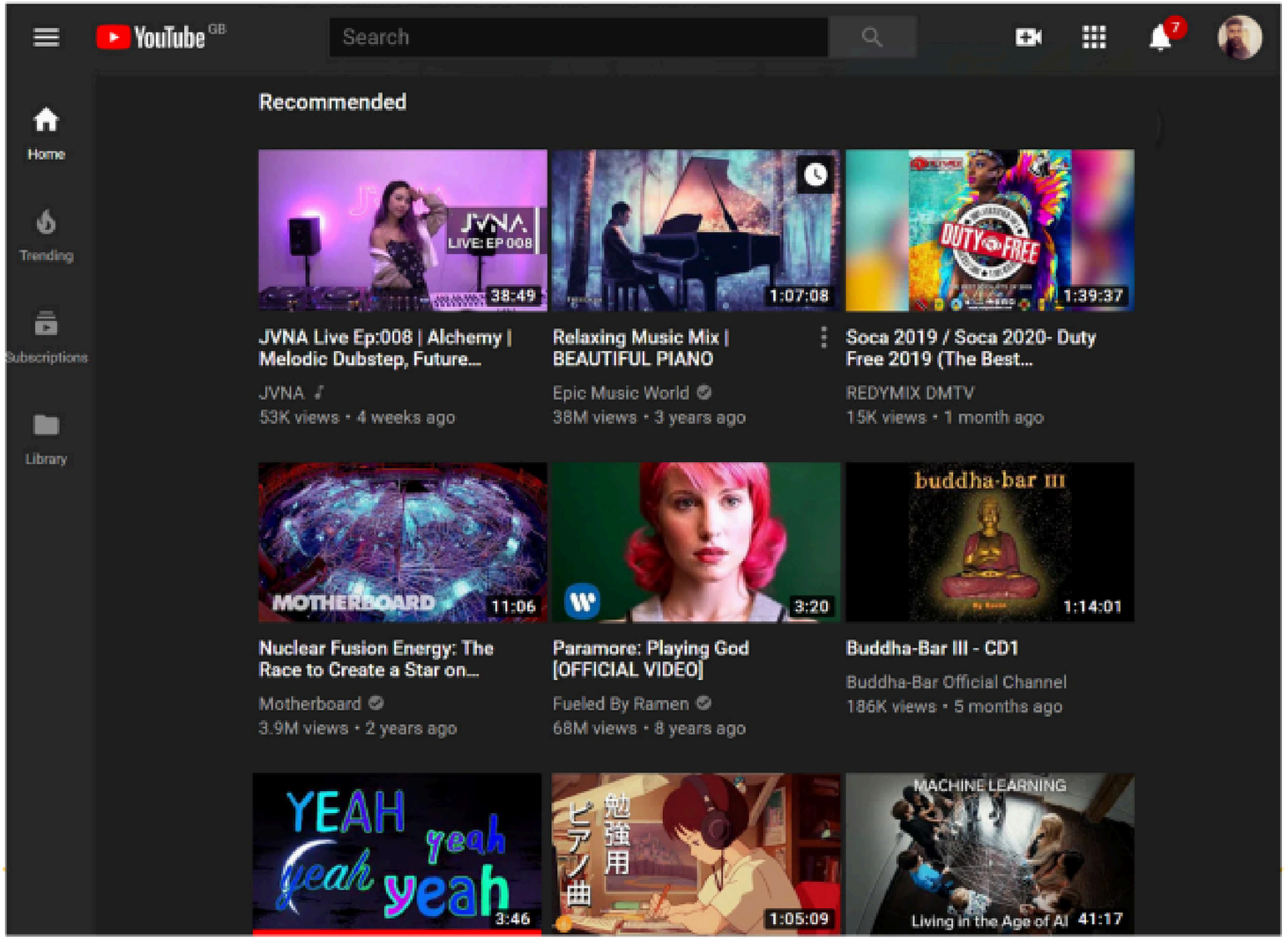

**Frequently bought together**


Total price: £477.42

Add all three to Basket

These items are dispatched from and sold by different sellers. Show details

- **This item:** Canon EF 70-300 mm f/4-5.6 IS II USM Lens - Black £449.00
- JJC Reversible Lens Hood Shade for Canon EF 70-300mm f/4-5.6 IS II USM Replaces Canon Lens Hood ET... £14.99
- Hoya 67mm UV(C) Digital HMC Screw-in Filter,YSUVC067 £13.43




Adapted from Dr. Mahmud Dwi Sulistiyo Slides

# Banking

- For better or worse, banks are harnessing their data to determine which customers to give loans to, which they should extend their credit line, and even who might file for bankruptcy

## Data = Value = Better Decisions = More Profits!

GLOW 2024

Fakultas Informatika
School of Computing
Telkom University

CCI
College of Computing and Informatics

# What is Data?

Adapted from Dr. Mahmud Dwi Sulistiyo Slides

# What is Data?

- **Data** are characteristics or information, usually numeric, that are collected through observation
- **Data** are a set of values of **qualitative** or **quantitative** variables about one or more persons or objects,
- while a **datum** (singular of data) is a single value of a single variable

# Dataset

- A collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer
- The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set
- Each value is known as a datum

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Sepal Length | Sepal Width | Petal Length | Petal Width | Class |
| 2 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 3 | 4.9 | 3 | 1.4 | 0.2 | Iris-setosa |
| 4 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 5 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 6 | 5 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 7 | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 8 | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 9 | 5 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 10 | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 11 | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| 12 | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| 13 | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| 14 | 4.8 | 3 | 1.4 | 0.1 | Iris-setosa |
| 15 | 4.3 | 3 | 1.1 | 0.1 | Iris-setosa |
| 16 | 5.8 | 4 | 1.2 | 0.2 | Iris-setosa |
| 17 | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| 18 | 5.4 | 3.9 | 1.3 | 0.4 | Iris-setosa |
| 19 | 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| 20 | 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |
| 21 | 5.1 | 3.8 | 1.5 | 0.3 | Iris-setosa |
| 22 | 5.4 | 3.4 | 1.7 | 0.2 | Iris-setosa |
| 23 | 5.1 | 3.7 | 1.5 | 0.4 | Iris-setosa |
| 24 | 4.6 | 3.6 | 1 | 0.2 | Iris-setosa |
| 25 | 5.1 | 3.3 | 1.7 | 0.5 | Iris-setosa |

# Dataset Example

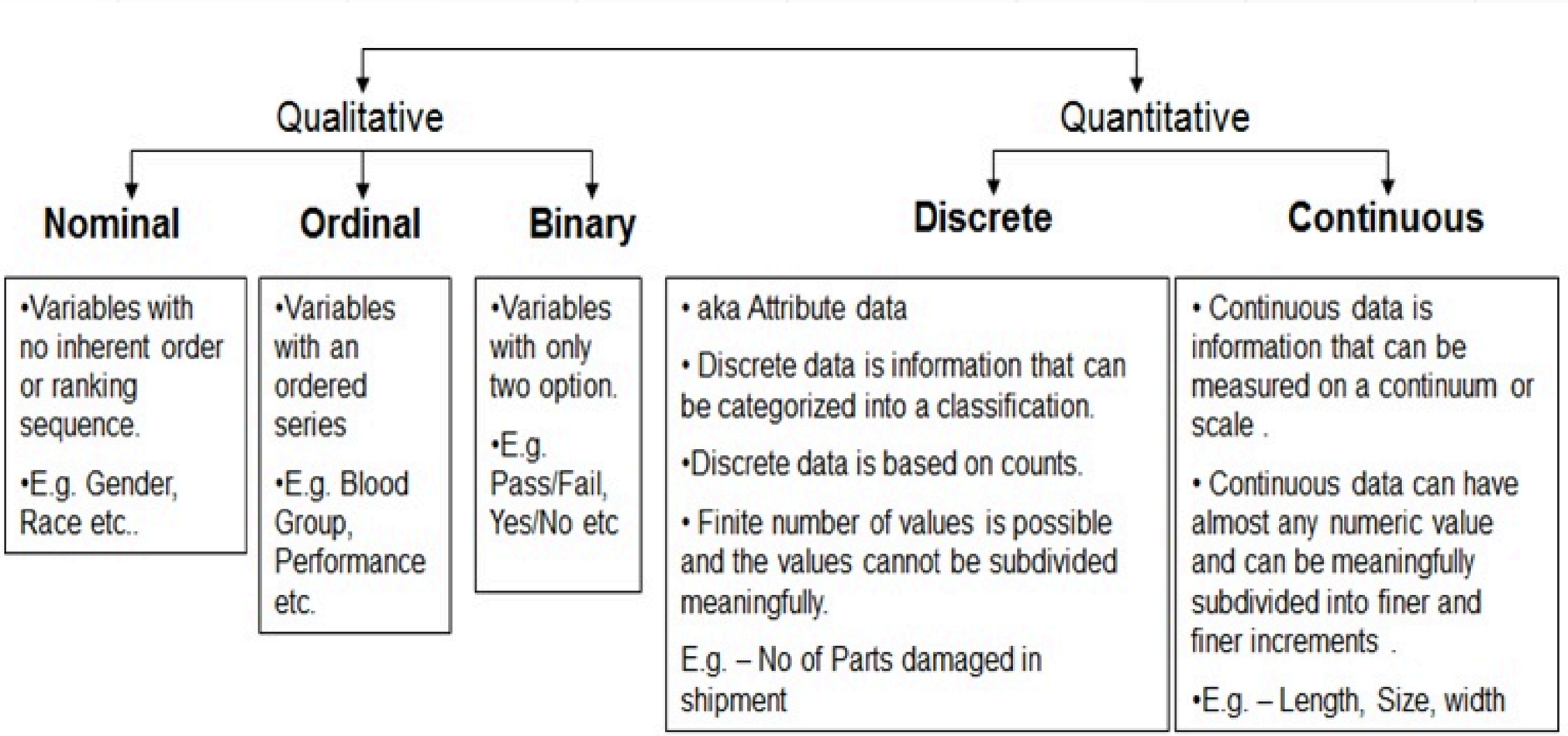
| ID | Name | Quiz | Assignment | Test | Index |
|---|---|---|---|---|---|
| 11023029 | Udin | 30 | 40 | 10 | D |
| 11023035 | Adi | 40 | 45 | 20 | D |
| 11023046 | Ari | 40 | 45 | 30 | C |
| 11023049 | Cecep | 50 | 65 | 40 | C |
| 11023051 | Deden | 60 | 65 | 50 | C |
| 11023054 | Komar | 60 | 65 | 60 | B |
| 11023067 | Ajeng | 70 | 70 | 70 | B |
| 11023076 | Budi | 80 | 75 | 80 | B |
| 11023078 | Indra | 80 | 75 | 90 | A |
| 11023098 | Wahyu | 90 | 90 | 100 | A |

# Attribute

- Data can be constructed from a set of objects or samples
- Each object represents an entity and can be expressed with its attributes or set of attributes

- Example:
  - Data of Final Grade consists of several students,
  - with attributes such as Student' ID, Name, Quiz, Assignments, Tests, Index

# Types of Attributes



# Types of Attributes

- Qualitative attribute that constructs a data may vary:
  - **Nominal**: also called Categorical that describes categories, states, codes, or anything without any order. Example: colors of the traffic light.
  - **Binary**: also called Boolean that only has two values: 0 or 1; 0 usually represents no/negative, while 1 is for yes/positive.
  - **Ordinal**: attribute that states an order, but the difference between values is unknown. Example: the price of orchids are represented as cheap, reasonable, and expensive

# Types of Attributes

- Quantitative attribute that constructs a data may vary:
  - **Discrete**: has a finite value and can be calculated, usually in an integer value. Example: the number of positive cases of Covid-19
  - **Continuous**: has a fractional value which is usually represented in real value, can be unlimited. Example: distance in km which can be worth 10.56 km

# Exercise

- Identify nominal, ordinal, binary, continuous and discrete attribute.

| Person ID | Car Color | Education Level | Subscribed (Binary) | Number of Children | Height (cm) |
|---|---|---|---|---|---|
| 1 | Red | Bachelor's | Yes | 2 | 175.4 |
| 2 | Blue | Master's | No | 0 | 160.7 |
| 3 | Green | High School | Yes | 3 | 182.3 |
| 4 | Red | Doctorate | No | 1 | 167.5 |
| 5 | Blue | Bachelor's | Yes | 2 | 180.9 |

GLOW 2024

Fakultas Informatika
School of Computing
Telkom University

CCI
College of Computing and Informatics

# Data Understanding

Adapted from Dr. Mahmud Dwi Sulistiyo Slides

# Understanding the Data

- The attributes that compose data objects need to be understood so that the data processing can be done correctly
- Tools in understanding the data include:
  - **Central tendency**; or data center tendency can be measured by mean, median, mode, or midrange
  - **Distribution of data**; can be used to characterize the data and further analyze the data that belongs to outliers or not
  - **Statistical graphs**; become a tool in the statistical approach in understanding the data

# Central Tendency

- Stating the central tendency of the data distribution on an attribute. Can be measured by:
  - **Average**: the sum of all values of an attribute divided by the number of data.
    - **Weaknesses**: cannot be used on categorical data, sensitive to outliers.
  - **Median**: the center of the set of $N$ values in attribute $X$. For odd $N$, the media is exactly in the middle. For even $N$, the median is the average of two values exactly in the middle.
  - **Mode**: the value that appears most frequently from the data.
  - **Midrange**: the average of the minimum and maximum values of an attribute.

# Data Distribution

- Can be used to characterize and further analyze the data. One that can be used is to determine whether a data is an outlier or not
- Two ways:
  - **Interquartile range (IQR)**: the distance between the first quarter of the data and the third quarter (Q3-Q1)
  - **Standard Deviation (STD)**: standard deviation is calculated by adding up all distance values by the average, divided by the number of data ($N$). A small STD indicates that the data tend to be close each other.

# Statistical Graph

- Usually used to understand data in graphical form
- Some that can be used:
  - Histogram
  - Scatterplots
  - Quantile plots
  - QQ-Plots

# Demo: Data Understanding

GLOW 2024

Fakultas Informatika
School of Computing
Telkom University

CCI
College of Computing and Informatics

# Data Pre-Processing

Adapted from Dr. Mahmud Dwi Sulistiyo Slides

# The Machine Learning Process

**Data Pre-Processing**
- Import the data
- Clean the data
- Split into training & test sets
- Feature Scaling

**Modelling**
- Build the model
- Train the model
- Make predictions

**Evaluation**
- Calculate performance metrics
- Make a verdict

# Why Do  We Need to Clean Our Data?

# Data Type Constraints

| Datatype | Example |
|---|---|
| Text data | First name, last name, address ... |
| Integers | # Subscribers, # products sold ... |
| Decimals | Temperature, $ exchange rates ... |
| Binary | Is married, new customer, yes/no, ... |
| Dates | Order dates, ship dates ... |
| Categories | Marriage status, gender ... |

| Python data type |
|---|
| `str` |
| `int` |
| `float` |
| `bool` |
| `datetime` |
| `category` |

# Data Type Constraints

E.g. Strings to integers

```
1  # import CSV and output header
2  fifa = pd.read_csv('fifa-21.csv')
3  fifa.head(10)
```

| name | age | dob | height_cm | weight_kg | nationality | wage_eur | international_reputation |
|---|---|---|---|---|---|---|---|
| L. Messi | 34 | 24/6/1987 | 170 | 158.0 | Argentina | RM560000 | 6 |
| Cristiano Ronaldo | 36 | 5/2/1985 | 187 | 83.0 | Portugal | RM220000 | 6 |
| J. Oblak | 28 | 7/1/1993 | 188 | 87.0 | Slovenia | RM125000 | 3 |
| R. Lewandowski | 32 | 21/8/1988 | 184 | 80.0 | Poland | RM240000 | 4 |
| R. Lewandowski | 32 | 21/8/1988 | 184 | 80.0 | Poland | RM240000 | 4 |
| Neymar Jr | 29 | 5/2/1992 | 175 | 68.0 | Brazil | RM270000 | 6 |
| K. De Bruyne | . | 28/6/1991 | 181 | 70.0 | Belgium | RM370000 | 4 |
| K. Mbappé | 22 | 20/12/1998 | 178 | 73.0 | France | RM160000 | 3 |
| M. ter Stegen | 29 | 30/4/1992 | 187 | 85.0 | Germany | RM260000 | 3 |

# Data Type Constraints

E.g. Strings to integers

```
1  # import CSV and output header
2  fifa = pd.read_csv('fifa-21.csv')
3  fifa.head(10)
```

| name | age | dob | height_cm | weight_kg | nationality | wage_eur | international_reputation |
|---|---|---|---|---|---|---|---|
| L. Messi | 34 | 24/6/1987 | 170 | 158.0 | Argentina | RM560000 | 6 |
| Cristiano Ronaldo | 36 | 5/2/1985 | 187 | 83.0 | Portugal | RM220000 | 6 |
| J. Oblak | 28 | 7/1/1993 | 188 | 87.0 | Slovenia | RM125000 | 3 |
| R. Lewandowski | 32 | 21/8/1988 | 184 | 80.0 | Poland | RM240000 | 4 |
| R. Lewandowski | 32 | 21/8/1988 | 184 | 80.0 | Poland | RM240000 | 4 |
| Neymar Jr | 29 | 5/2/1992 | 175 | 68.0 | Brazil | RM270000 | 6 |
| K. De Bruyne | . | 28/6/1991 | 181 | 70.0 | Belgium | RM370000 | 4 |
| K. Mbappé | 22 | 20/12/1998 | 178 | 73.0 | France | RM160000 | 3 |
| M. ter Stegen | 29 | 30/4/1992 | 187 | 85.0 | Germany | RM260000 | 3 |

# Data Type Constraints

E.g. Strings to integers

```
1  # Get DataFrame information
2  fifa.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104 entries, 0 to 103
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          104 non-null    int64
 1   name                        104 non-null    object
 2   age                         104 non-null    object
 3   dob                         104 non-null    object
 4   height_cm                   104 non-null    object
 5   weight_kg                   101 non-null    float64
 6   nationality                 104 non-null    object
 7   wage_eur                    104 non-null    object
 8   international_reputation    104 non-null    int64
 9   preferred_foot              101 non-null    object
 10  attacking_crossing          101 non-null    float64
 11  attacking_finishing         101 non-null    float64
 12  attacking_heading_accuracy  101 non-null    float64
 13  attacking_short_passing     101 non-null    float64
 14  attacking_volleys           101 non-null    float64
dtypes: float64(6), int64(2), object(7)
memory usage: 12.3+ KB
```

Due to 'RM'

# Data Type Constraints

E.g. Strings to integers

```
# Remove 'RM' from Wage_Eur column
fifa['wage_eur'] = fifa['wage_eur'].str.strip('RM')
# Convert the Wage_Eur column to integer
fifa['wage_eur'] = fifa['wage_eur'].astype('int')
```

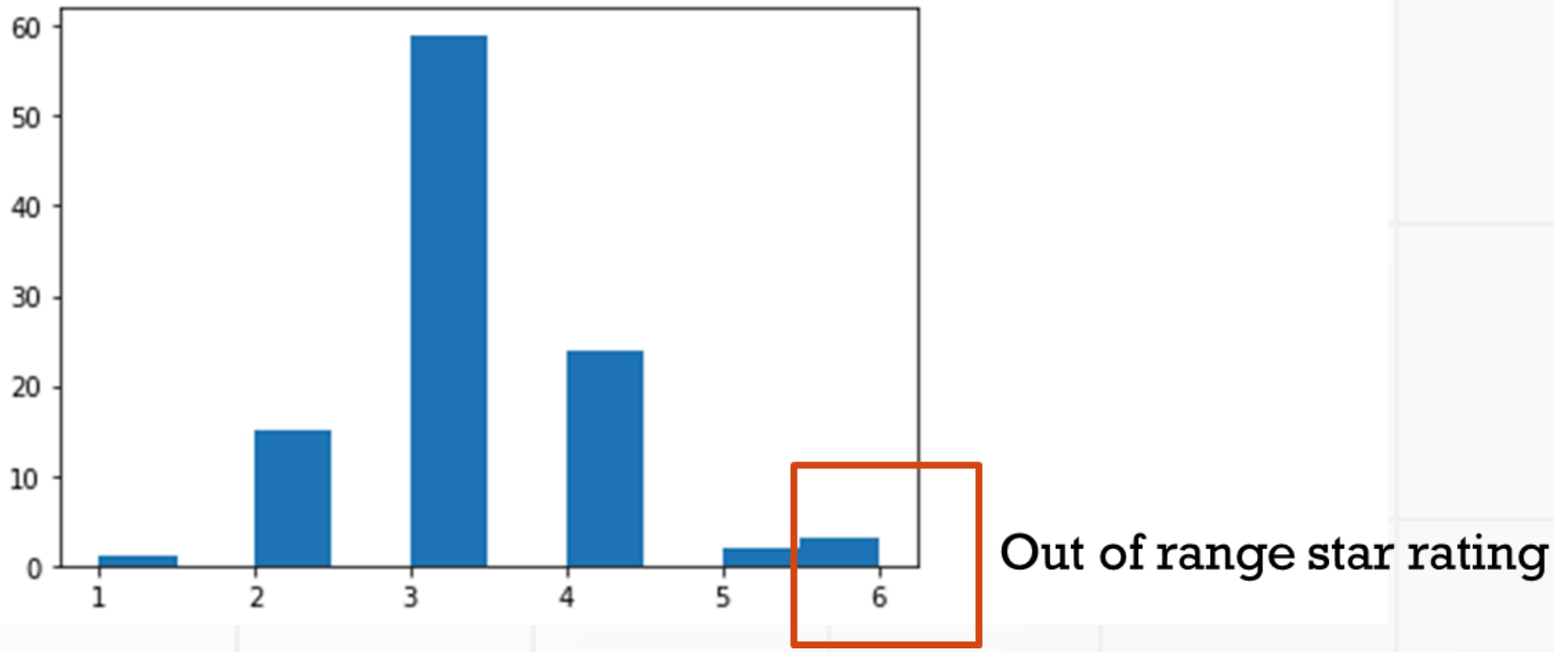
```
# Verify that the Wage_Eur column is now an integer
fifa.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104 entries, 0 to 103
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          104 non-null    int64
 1   name                        104 non-null    object
 2   age                         101 non-null    float64
 3   dob                         104 non-null    object
 4   height_cm                   101 non-null    float64
 5   weight_kg                   101 non-null    float64
 6   nationality                 104 non-null    object
 7   wage_eur                    104 non-null    int32
 8   international_reputation    104 non-null    int64
 9   preferred_foot              101 non-null    object
 10  attacking_crossing          101 non-null    float64
 11  attacking_finishing         101 non-null    float64
 12  attacking_heading_accuracy  101 non-null    float64
 13  attacking_short_passing     101 non-null    float64
 14  attacking_volleys           101 non-null    float64
dtypes: float64(8), int32(1), int64(2), object(4)
memory usage: 11.9+ KB
```

# Data Range Constraints

```
1 # Check the out of range International_Reputation column
2 import matplotlib.pyplot as plt
3 plt.hist(fifa['international_reputation'])
```

```
(array([ 1.,  0., 15.,  0., 59.,  0., 24.,  0.,  2.,  3.]),
 array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. , 5.5, 6. ]),
 <BarContainer object of 10 artists>)
```




Out of range star rating

# How to deal with out of range data?

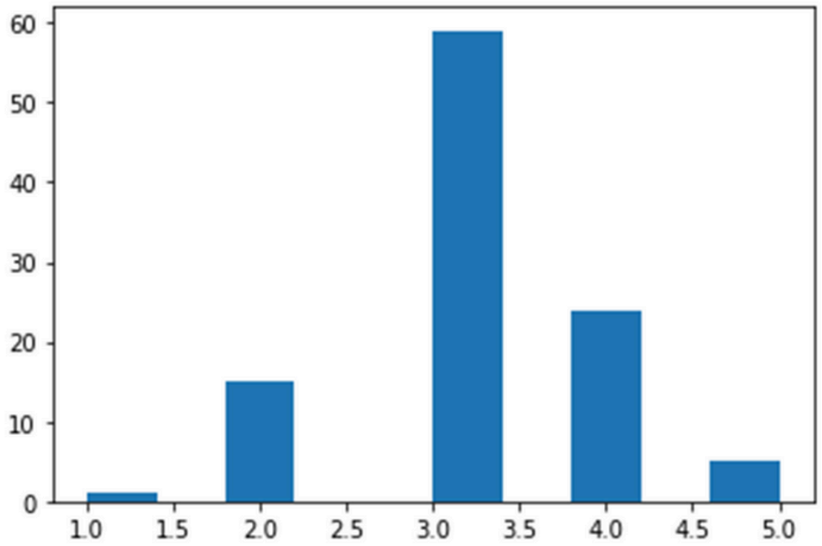
- Dropping data
- Setting custom minimums and maximums
- Treat as missing and impute
- Setting custom value depending on business assumptions

# How to deal with out of range data?

- E.g. Setting custom value depending on business assumptions

```
1  # Convert International_Reputation column
2  fifa.loc[fifa['international_reputation'] > 5, 'international_reputation'] = 5
3  # Check the new range of International_Reputation column
4  plt.hist(fifa['international_reputation'])
```

```
(array([ 1.,  0., 15.,  0.,  0., 59.,  0., 24.,  0.,  5.]),
 array([1. , 1.4, 1.8, 2.2, 2.6, 3. , 3.4, 3.8, 4.2, 4.6, 5. ]),
 <BarContainer object of 10 artists>)
```



# How to deal with out of range data?

- E.g. Data type constraints - Numerical or categorical?

```
# Convert the International_Reputation column to categorical
fifa['international_reputation'] = fifa['international_reputation'].astype('category')
# Verify that the International_Reputation column is now categorical
fifa.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 104 entries, 0 to 103
Data columns (total 15 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          104 non-null    int64
 1   name                        104 non-null    object
 2   age                         104 non-null    object
 3   dob                         104 non-null    object
 4   height_cm                   104 non-null    object
 5   weight_kg                   101 non-null    float64
 6   nationality                 104 non-null    object
 7   wage_eur                    104 non-null    int32
 8   international_reputation    104 non-null    category
 9   preferred_foot              101 non-null    object
 10  attacking_crossing          101 non-null    float64
 11  attacking_finishing         101 non-null    float64
 12  attacking_heading_accuracy  101 non-null    float64
 13  attacking_short_passing     101 non-null    float64
 14  attacking_volleys           101 non-null    float64
dtypes: category(1), float64(6), int32(1), int64(1), object(6)
memory usage: 11.4+ KB
```

# Uniqueness Constraints

- What are duplicate values?

**All columns have the same values**

| first_name | last_name | address | height | weight |
|---|---|---|---|---|
| Justin | Saddlemyer | Boulevard du Jardin Botanique 3, Bruxelles | 193 cm | 87 kg |
| Justin | Saddlemyer | Boulevard du Jardin Botanique 3, Bruxelles | 193 cm | 87 kg |

**Most columns have the same values**

| first_name | last_name | address | height | weight |
|---|---|---|---|---|
| Justin | Saddlemyer | Boulevard du Jardin Botanique 3, Bruxelles | 193 cm | 87 kg |
| Justin | Saddlemyer | Boulevard du Jardin Botanique 3, Bruxelles | ***194 cm*** | 87 kg |

# Uniqueness Constraints

- Finding duplicate values

**The `.duplicated()` method**

`subset` : List of column names to check for duplication.

`keep` : Whether to keep **first** ( `'first'` ), **last** ( `'last'` ) or **all** ( `False` ) duplicate values.

```
# Get duplicate rows by id
duplicates = fifa.duplicated(subset='id', keep=False)
# Output duplicate values
fifa[duplicates]
```

| | id | name | age | dob | height_cm | weight_kg | nationality | wage_eur |
|---|---|---|---|---|---|---|---|---|
| 3 | 188545 | R. Lewandowski | 32 | 21/8/1988 | 184 | 80.0 | Poland | 240000 |
| 4 | 188545 | R. Lewandowski | 32 | 21/8/1988 | 184 | 80.0 | Poland | 240000 |
| 12 | 209331 | M. Salah | . | 15/6/1992 | 175 | 71.0 | Egypt | 250000 |
| 13 | 209331 | M. Salah | 29 | 15/6/1992 | 175 | 71.0 | Egypt | 250000 |
| 16 | 155862 | Sergio Ramos | 35 | 30/3/1986 | . | 82.0 | Spain | 300000 |
| 17 | 155862 | Sergio Ramos | 35 | 30/3/1986 | 184 | 180.0 | Spain | 300000 |
| 20 | 200145 | Casemiro | 29 | 23/2/1992 | 185 | 84.0 | Brazil | 310000 |
| 21 | 200145 | Casemiro | 29 | 23/2/1992 | 185 | NaN | Brazil | 310000 |

# Uniqueness Constraints

- Treating duplicate values

**The `.drop_duplicates()` method**

`subset` : List of column names to check for duplication.

`keep` : Whether to keep **first** ( `'first'` ), **last** ( `'last'` ) or **all** ( `False` ) duplicate values.

`inplace` : Drop duplicated rows directly inside DataFrame without creating new object ( `True` ).

```
1  # Drop duplicates
2  fifa.drop_duplicates(inplace=True)
3  # Check again for duplicates
4  duplicates = fifa.duplicated()
5  fifa[duplicates]
```

| id | name | age | dob | height_cm | weight_kg | nationality | wage_eur | international_repu |
|---|---|---|---|---|---|---|---|---|

# Uniqueness Constraints

- Treating duplicate values

**The `.drop_duplicates()` method**

`subset` : List of column names to check for duplication.

`keep` : Whether to keep **first** ( `'first'` ), **last** ( `'last'` ) or **all** ( `False` ) duplicate values.

`inplace` : Drop duplicated rows directly inside DataFrame without creating new object ( `True` ).

```
1  # Drop duplicates
2  fifa.drop_duplicates(inplace=True)
3  # Check again for duplicates
4  duplicates = fifa.duplicated()
5  fifa[duplicates]
```

| id | name | age | dob | height_cm | weight_kg | nationality | wage_eur | international_repu |
|---|---|---|---|---|---|---|---|---|

# Text & Categorical Inconsistencies

**I) Value inconsistency**

- *Inconsistent fields:* `'married'`, `'Maried'`, `'UNMARRIED'`, `'not married'` ..
- _Trailing white spaces: _ `'married '`, `' married '` ..

**II) Collapsing too many categories to few**

- *Creating new groups:* `0-20K`, `20-40K` categories ... from continuous household income data
- *Mapping groups to new ones:* Mapping household income categories to 2 `'rich'`, `'poor'`

**III) Making sure data is of type** `category`

# Text & Categorical Inconsistencies

- E.g. Value inconsistencies

```
1  # Get Preferred_Foot column
2  prefered_foot = fifa['preferred_foot']
3  prefered_foot.value_counts()
```

```
Right      74
Left       21
 Right      1
Up          1
 Left       1
left        1
RIGHT       1
Name: preferred_foot, dtype: int64
```

Capitalization: ‘Right’, ‘RIGHT’, ‘Left’, ‘left’
Trailing whitespaces: ‘ Right’, ‘ Left ‘

# Text & Categorical Inconsistencies

- E.g. Value inconsistencies

```
[ ]:
1 # Lowercase
2 fifa['preferred_foot'] = fifa['preferred_foot'].str.lower()
3 fifa['preferred_foot'].value_counts()
```

```
[ ]:
right     75
left      22
 right     1
 left      1
up         1
Name: preferred_foot, dtype: int64
```

```
[ ]:
1 # Strip all whitespaces
2 fifa['preferred_foot'] = fifa['preferred_foot'].str.strip()
3 fifa['preferred_foot'].value_counts()
```

```
[ ]:
right    76
left     23
up        1
Name: preferred_foot, dtype: int64
```

# Text & Categorical Inconsistencies

- E.g. Find inconsistent categories

```
right     76
left      23
up         1
Name: preferred_foot, dtype: int64
```

right
left

```
# Set categories
foot_cat = ['right', 'left']
# Get and print inconsistent categories
consistent_rows = fifa['preferred_foot'].isin(foot_cat)
fifa[~consistent_rows]
```

| _eur | international_reputation | preferred_foot | attacking_crossing | attacking_finishing | atta |
|---|---|---|---|---|---|
| 5000 | 3 | up | 13.0 | 11.0 | |
| 0000 | 4 | NaN | 94.0 | 82.0 | |
| 0000 | 4 | NaN | 66.0 | 65.0 | |
| 5000 | 3 | NaN | 91.0 | 66.0 | |

# Text & Categorical Inconsistencies

- E.g. Dropping inconsistent categories

```
# Drop inconsistent categories
new_fifa = fifa[consistent_rows]
# Check the Preferred_Foot column again
new_fifa['preferred_foot'].value_counts()
```

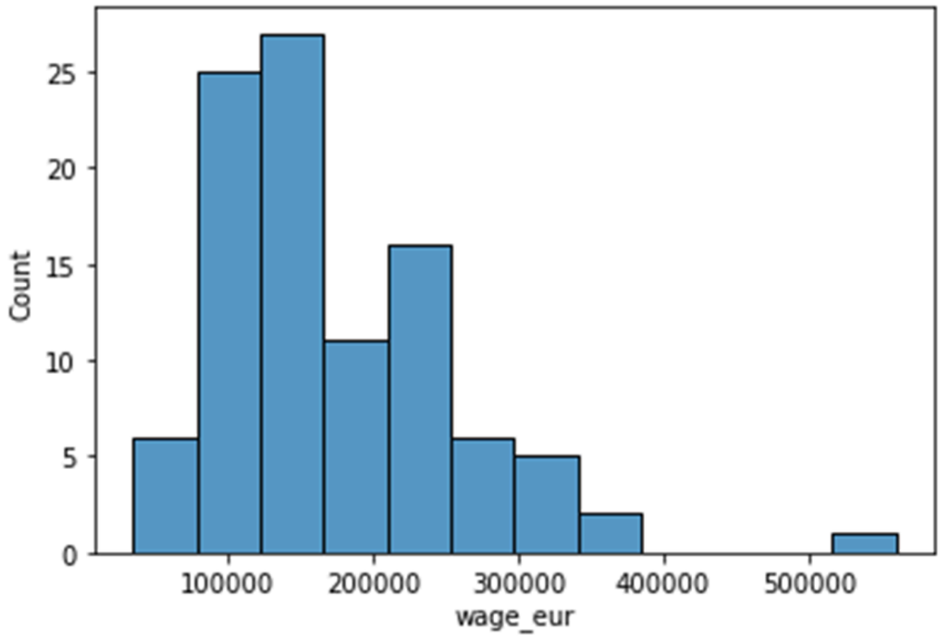
```
right    76
left     23
Name: preferred_foot, dtype: int64
```

# Text & Categorical Inconsistencies

- E.g. Create categories out of data

```
1  # Check Wage_Eur distribution using histogram
2  import seaborn as sns
3  sns.histplot(data=new_fifa, x="wage_eur")
```

```
<AxesSubplot:xlabel='wage_eur', ylabel='Count'>
```



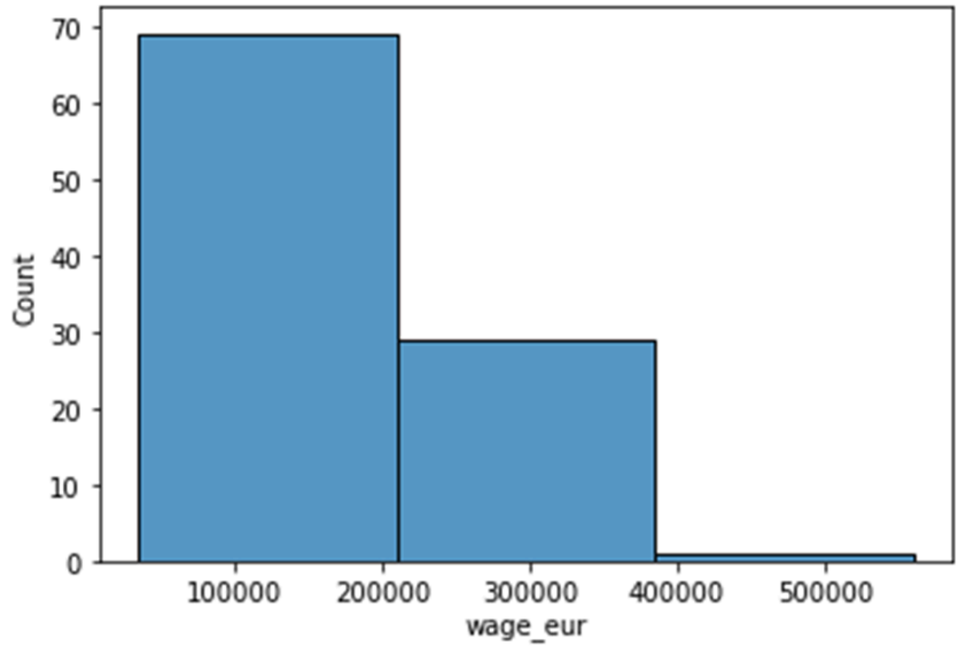

Many categories

# Text & Categorical Inconsistencies

- E.g. Create categories out of data

```
1  # Check range four 4 groups
2  sns.histplot(data=new_fifa, x="wage_eur", bins=3)
```

```
<AxesSubplot:xlabel='wage_eur', ylabel='Count'>
```




Reduce categories

# Text & Categorical Inconsistencies

- E.g. Create categories out of data

```
# Set group names
group_names = ['0-200K','200K-400K', '400K-600K+']
# Create new Wage_Group column from Wage_Eur column
new_fifa['wage_group'] = pd.cut(new_fifa['wage_eur'], bins = 3, labels = group_names)
# Print Wage_Group column
new_fifa[['wage_group', 'wage_eur']]
```

| | wage_group | wage_eur |
|---|---|---|
| 0 | 400K-600K+ | 560000 |
| 1 | 200K-400K | 220000 |
| 3 | 200K-400K | 240000 |
| 5 | 200K-400K | 270000 |
| 7 | 0-200K | 160000 |
| ... | ... | ... |
| 99 | 0-200K | 120000 |
| 100 | 0-200K | 92000 |
| 101 | 0-200K | 120000 |
| 102 | 200K-400K | 210000 |
| 103 | 0-200K | 56000 |

99 rows × 2 columns

Create new category using cut()

# Uniformity

| Column | Unit |
| --- | --- |
| Temperature | `32°C` **is also** `89.6°F` |
| Weight | `70 Kg` **is also** `11 st.` |
| Date | `26-11-2019` **is also** `26, November, 2019` |
| Money | `100$` **is also** `10763.90¥` |

# Uniformity

- E.g. Ensuring uniformity across column

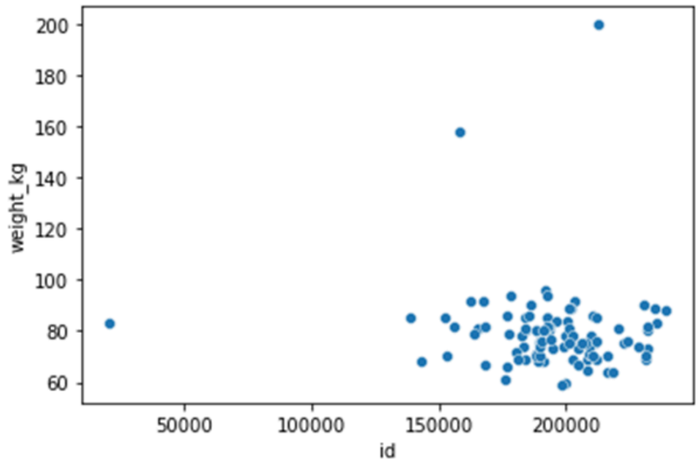
```
# Inspect Weight column
new_fifa['weight_kg']
```

```
0      158.0
1       83.0
3       80.0
5       68.0
7       73.0
       ...
99      80.0
100     77.0
101     59.0
102     67.0
103     88.0
Name: weight_kg, Length: 99, dtype: float64
```

Value should be in kilograms, this value is in pounds (more than 100)

```
# Inspect using scatterplot to check outlier
sns.scatterplot(x = 'id', y = 'weight_kg', data = new_fifa)
```

```
<AxesSubplot:xlabel='id', ylabel='weight_kg'>
```



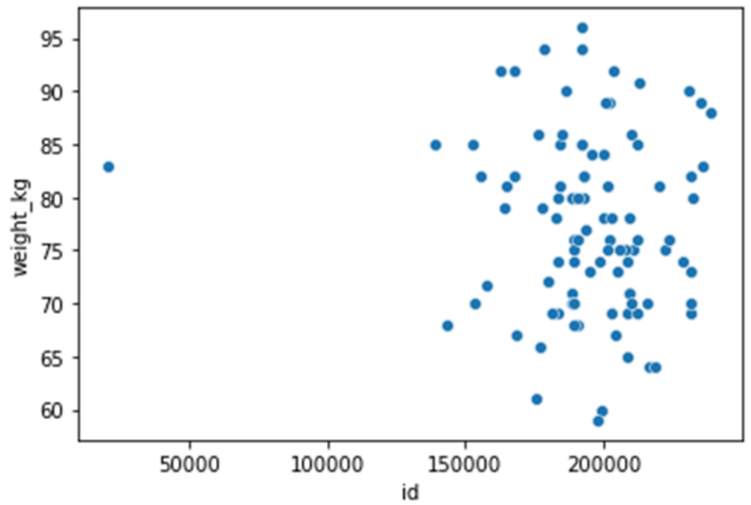
# Uniformity

- E.g. Ensuring uniformity across column

```
# Treating Weight_Kg column
get_weight_in_pounds = new_fifa.loc[new_fifa['weight_kg'] > 100, 'weight_kg']
pounds_to_kg = get_weight_in_pounds * 0.45359237
new_fifa.loc[new_fifa['weight_kg'] > 100, 'weight_kg'] = pounds_to_kg
```

```
# Confirming pounds coversion to kg
sns.scatterplot(x = 'id', y = 'weight_kg', data = new_fifa)
```

```
<AxesSubplot:xlabel='id', ylabel='weight_kg'>
```




No more value > 100

# Uniformity

- E.g. Cross field validation

```
# Cross field validation - age and dob
new_fifa[['age', 'dob']]
```

| | age | dob |
|---|---|---|
| 0 | 34 | 24/6/1987 |
| 1 | 36 | 5/2/1985 |
| 3 | 32 | 21/8/1988 |
| 5 | 29 | 5/2/1992 |
| 7 | 22 | 20/12/1998 |
| ... | ... | ... |
| 99 | 30 | 26/1/1991 |
| 100 | 29 | 8/1/1992 |
| 101 | 30 | 4/6/1991 |
| 102 | 30 | 21/2/1991 |
| 103 | 20 | 21/7/2000 |

# Uniformity

- E.g. Cross field validation

```
# Convert DOB column to datetime
import datetime as dt
new_fifa['dob'] = pd.to_datetime(new_fifa['dob'])
# Get today's date
today = dt.date.today()
# Calculate year difference for each row in DOB column
age_diff = today.year - new_fifa['dob'].dt.year
# Find rows where ages match
age_match = age_diff == new_fifa['age']
age_match.value_counts()
```

```
True      66
False     33
dtype: int64
```

33 unmatched age

```
# Create New_Age column
new_fifa['new_age'] = age_diff
```

# Incomplete Data

**What is missing data?**

*Occurs when no data value is stored for a variable in an observation*

Can be represented as `NA` , `nan` , `0` , `.` ...

Technical error

Human error

**Missing values are usually filled with values like 'NA', '-', or '.' etc.**

# Incomplete Data

- E.g. Replace missing values '.' with NaN

```
1 # Replace missing values '.' with NaN
2 import numpy as np
3 new_fifa['height_cm'] = new_fifa['height_cm'].replace('.', np.nan)
```

# Incomplete Data

- E.g. Convert missing values '.' to NaN

```
# Convert missing values '.' with NaN
import numpy as np
new_fifa['height_cm'] = new_fifa['height_cm'].replace('.', np.nan)
```

```
new_fifa.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 99 entries, 0 to 103
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          99 non-null     int64
 1   name                        99 non-null     object
 2   age                         99 non-null     int64
 3   dob                         99 non-null     datetime64[ns]
 4   height_cm                   97 non-null     object
 5   weight_kg                   96 non-null     float64
 6   nationality                 99 non-null     object
 7   wage_eur                    99 non-null     int32
 8   international_reputation    99 non-null     category
 9   preferred_foot              99 non-null     object
 10  attacking_crossing          96 non-null     float64
 11  attacking_finishing         96 non-null     float64
 12  attacking_heading_accuracy  96 non-null     float64
 13  attacking_short_passing     96 non-null     float64
 14  attacking_volleys           96 non-null     float64
 15  wage_group                  99 non-null     category
 16  new_age                     99 non-null     int64
dtypes: category(2), datetime64[ns](1), float64(6), int32(1), int64
(3), object(4)
```

Missing values:
height_cm
weight_kg
attacking_crossing
attacking_finishing
attacking_heading_accuracy
attacking_short_passing
attacking_volleys

**How to deal with missing data?**

1. Drop missing data

2. Impute with statistical measures (mean, median, mode..)

# Incomplete Data

- E.g. Impute with statistical measures

```python
# Convert Height_kg to float
new_fifa['height_cm'] = pd.to_numeric(new_fifa['height_cm'])
# Impute Height_kg and Weight_kg columns
new_fifa['height_cm'] = new_fifa['height_cm'].replace(np.nan,new_fifa['height_cm'].mean())
new_fifa['weight_kg'] = new_fifa['weight_kg'].replace(np.nan,new_fifa['weight_kg'].median())
# Check null counts
new_fifa.info()
```

```
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   id                        99 non-null     int64
 1   name                      99 non-null     object
 2   age                       99 non-null     int64
 3   dob                       99 non-null     datetime64[ns]
 4   height_cm                 99 non-null     float64
 5   weight_kg                 99 non-null     float64
 6   nationality               99 non-null     object
 7   wage_eur                  99 non-null     int32
 8   international_reputation  99 non-null     category
 9   preferred_foot            99 non-null     object
 10  attacking_crossing        96 non-null     float64
 11  attacking_finishing       96 non-null     float64
 12  attacking_heading_accuracy 96 non-null    float64
 13  attacking_short_passing   96 non-null     float64
 14  attacking_volleys         96 non-null     float64
 15  wage_group                99 non-null     category
 16  new_age                   99 non-null     int64
```

# Incomplete Data

- E.g. Remove rows with missing data

```
# Remove rows with missing values in Attacking columns
attacking = ['attacking_crossing', 'attacking_finishing', 'attacking_heading_accuracy', 'attacking_short_passing', 'attackin
new_fifa = new_fifa.dropna(subset=attacking)
new_fifa.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 96 entries, 0 to 103
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   id                          96 non-null     int64
 1   name                        96 non-null     object
 2   age                         96 non-null     int64
 3   dob                         96 non-null     datetime64[ns]
 4   height_cm                   96 non-null     float64
 5   weight_kg                   96 non-null     float64
 6   nationality                 96 non-null     object
 7   wage_eur                    96 non-null     int32
 8   international_reputation    96 non-null     category
 9   preferred_foot              96 non-null     object
 10  attacking_crossing          96 non-null     float64
 11  attacking_finishing         96 non-null     float64
 12  attacking_heading_accuracy  96 non-null     float64
 13  attacking_short_passing     96 non-null     float64
 14  attacking_volleys           96 non-null     float64
 15  wage_group                  96 non-null     category
 16  new_age                     96 non-null     int64
dtypes: category(2), datetime64[ns](1), float64(7), int32(1), int64(3), object(3)
memory usage: 12.1+ KB
```