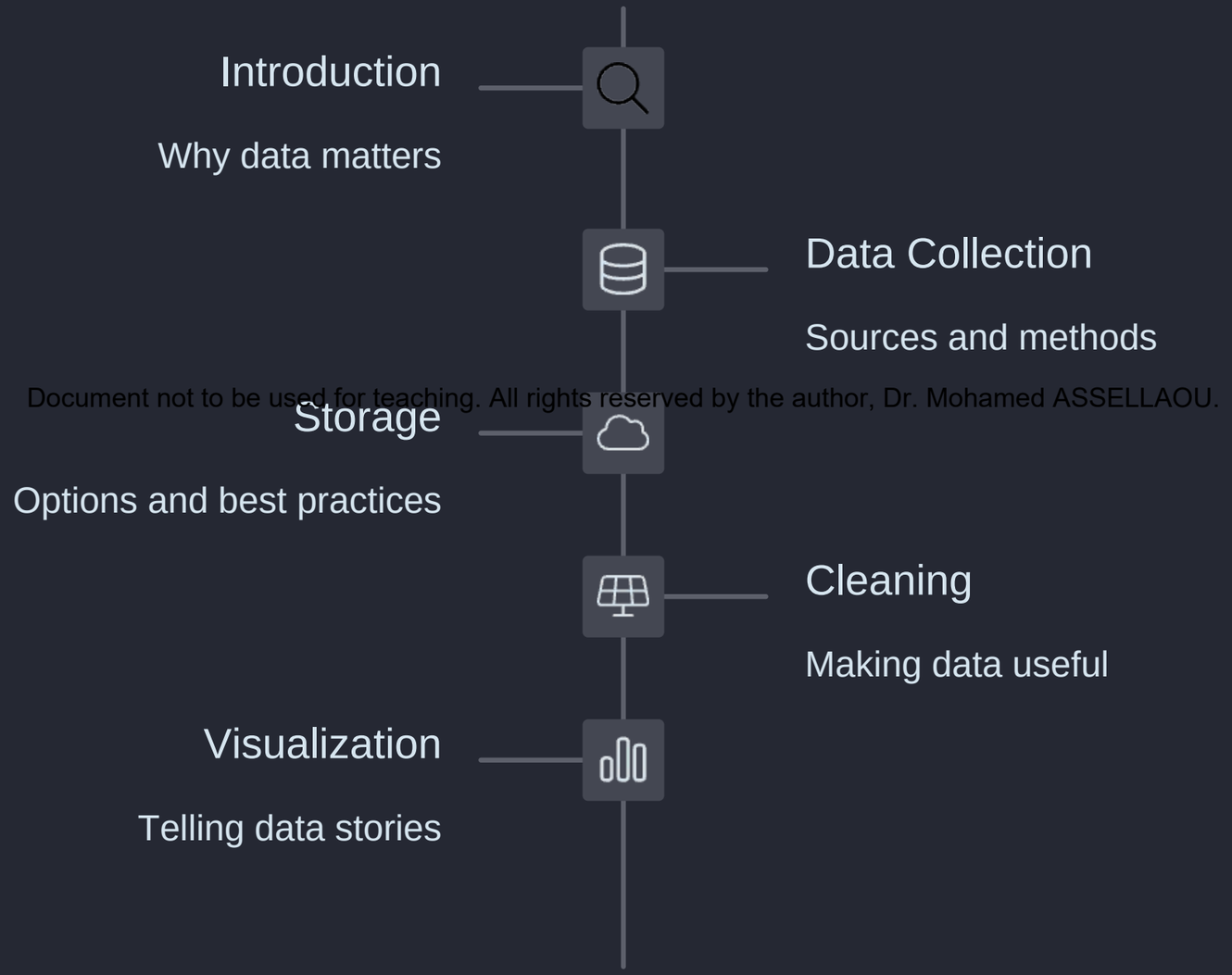
Data Collection, Storage, Cleaning & Visualization

Dr. Mohamed ASSELLAOU

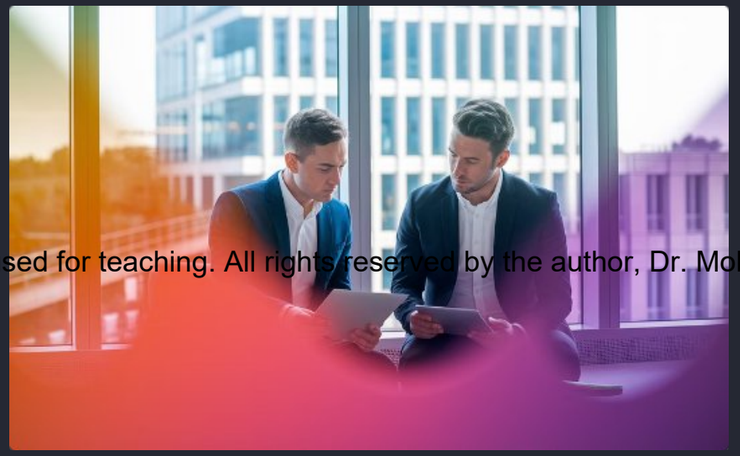
Today's Journey



Data is the New Oil

Powers Innovation

Fuels AI systems and smart solutions



Drives Decisions

Transforms guesswork into evidence-based action

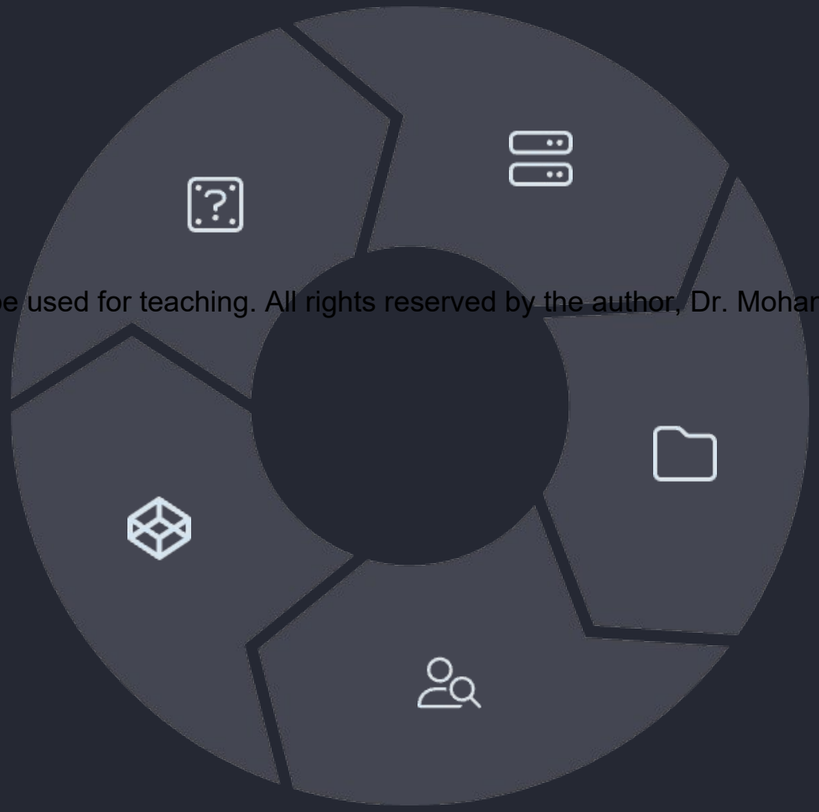
Creates Value

Generates insights that create competitive edge

Data Science Project Lifecycle

Problem Understanding

Defining goals and requirements



Data Collection

Gathering relevant information

Data Storage & Cleaning

Organizing and preparing data

EDA

Exploratory Data Analysis

Model Building & Deployment

Creating and implementing solutions

First Step: Data Collection

Web Data

APIs, scraping

Existing Files

CSV, JSON, databases

User Input

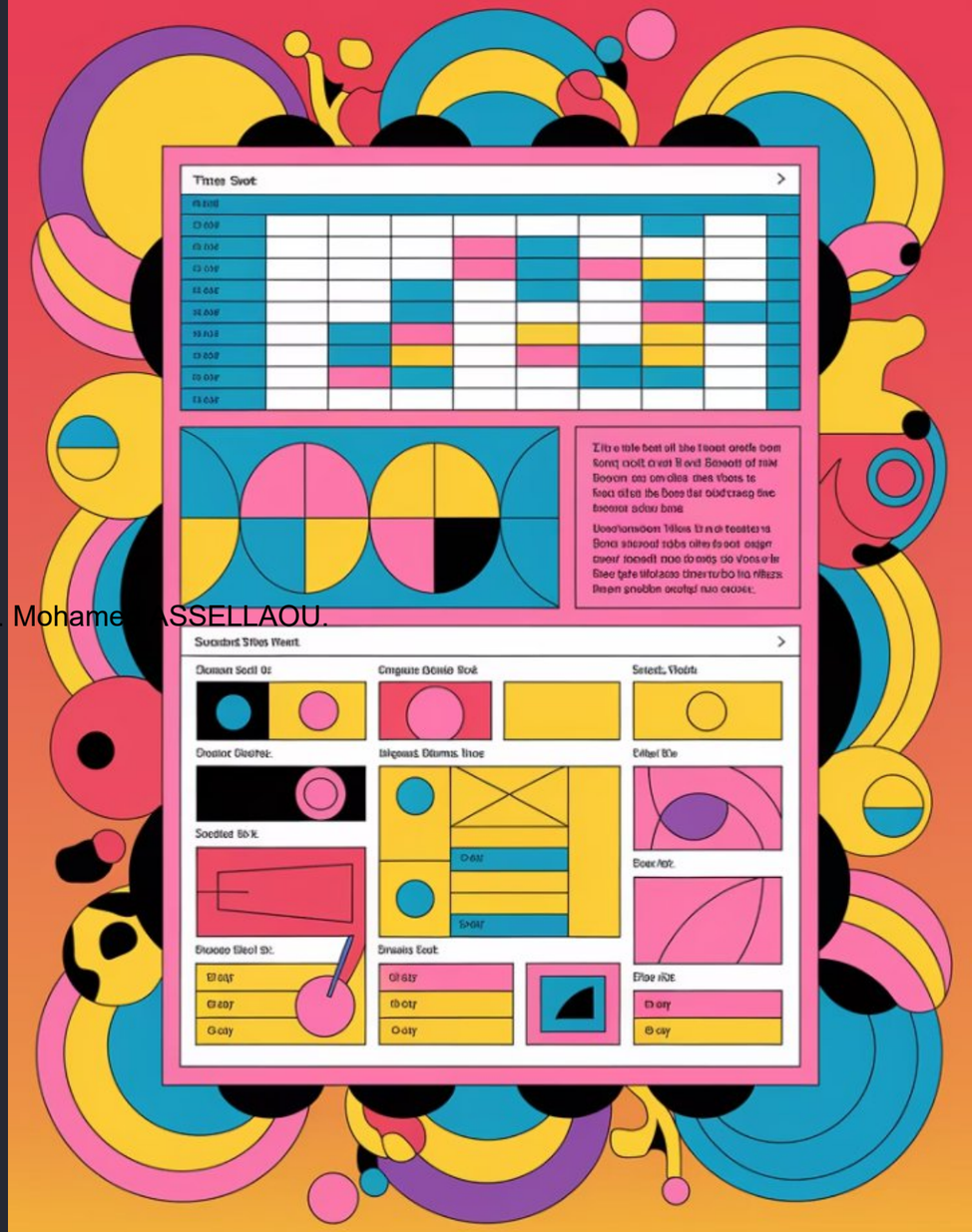
Forms, surveys

Sensors/IoT

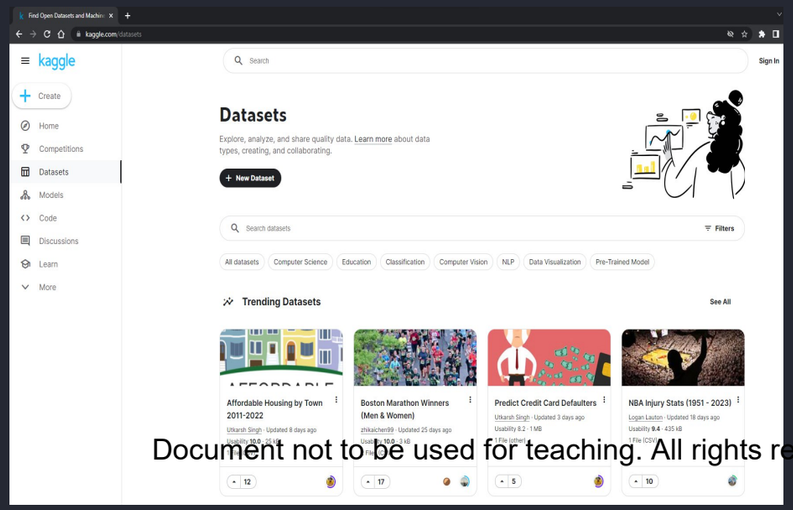
Continuous streams

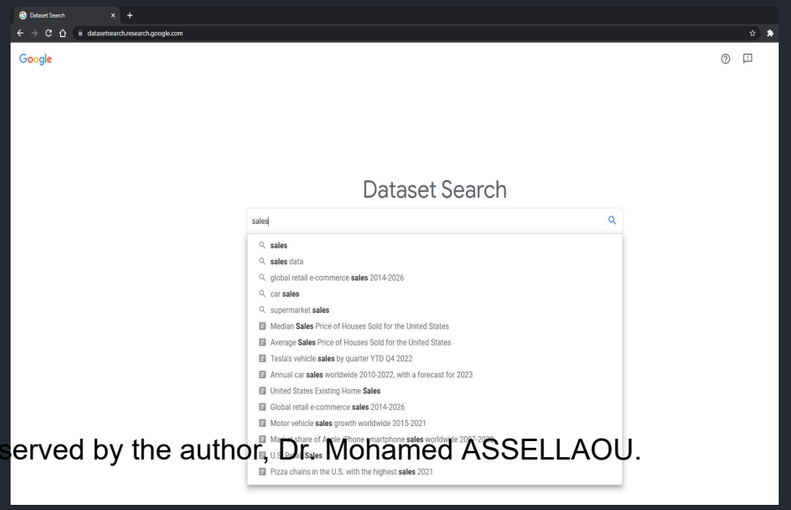
What Kind of Data?

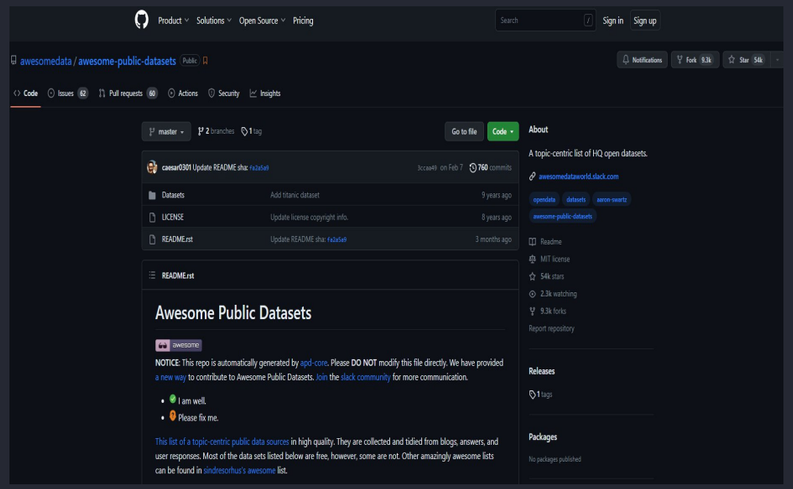
Type	Format	Examples
Structured	Tables	SQL, CSV, Excel
Semi-structured	Tagged	JSON, XML, YAML
Unstructured	Raw	Text, images, audio

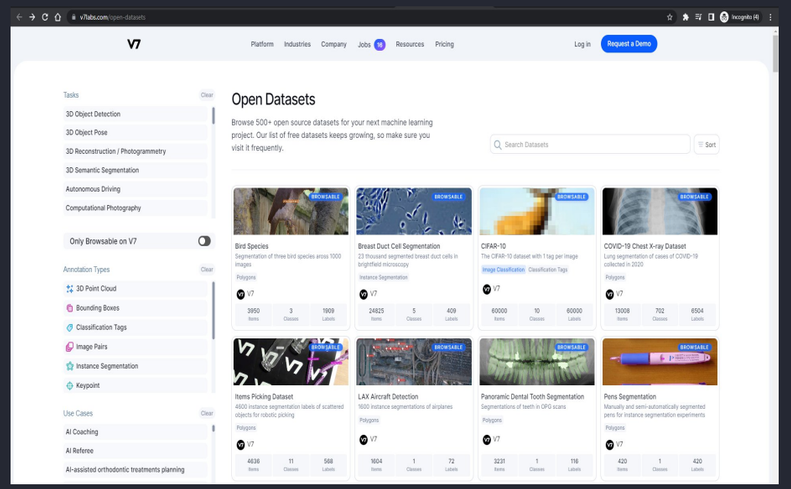


Free Datasets









APIs – Your Data Gateway

Request

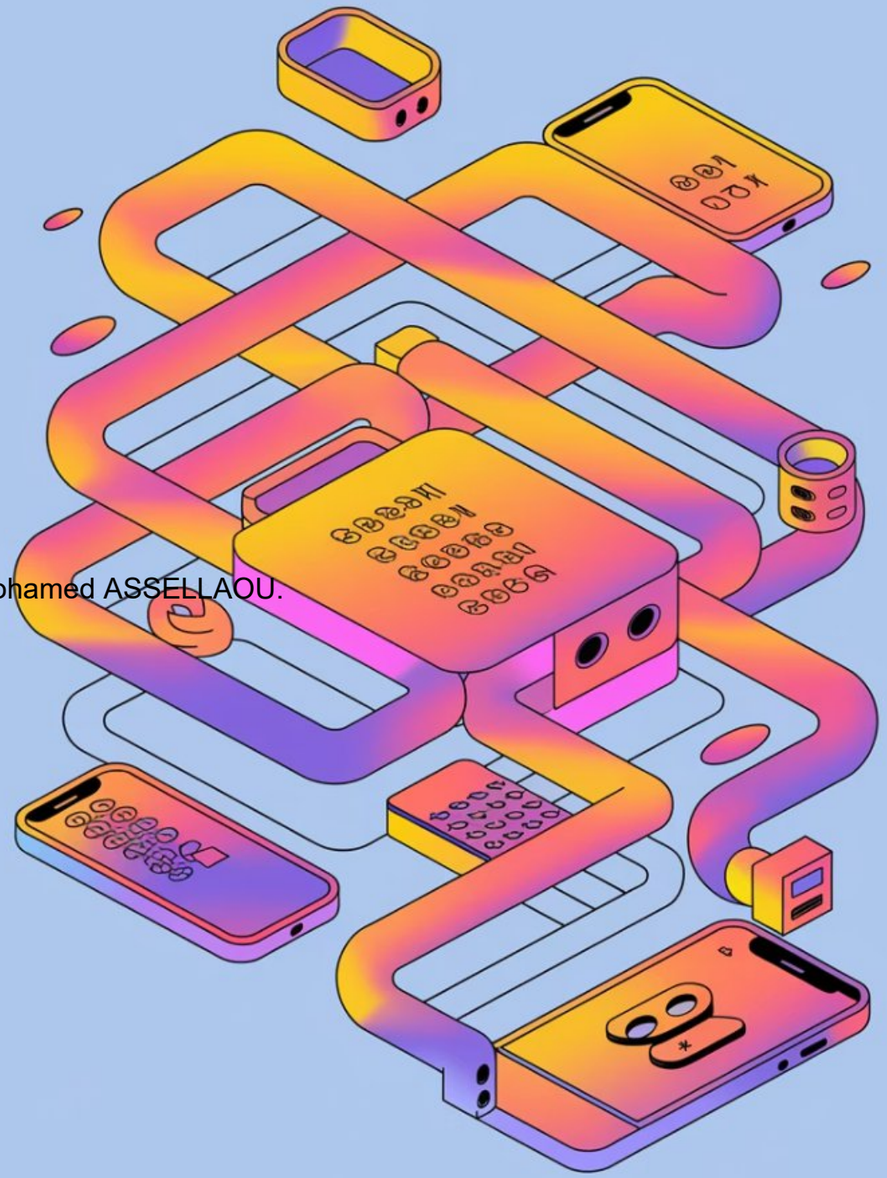
App sends query to API endpoint

Processing

API interprets request and retrieves data

Response

Data returned in structured format



Try This!

- Collect data from the https://www.fruityvice.com/ API
- Clean and store the data into a DataFrame
- Save the data into a CSV file

Colab Notebook

Scrape the Web

What

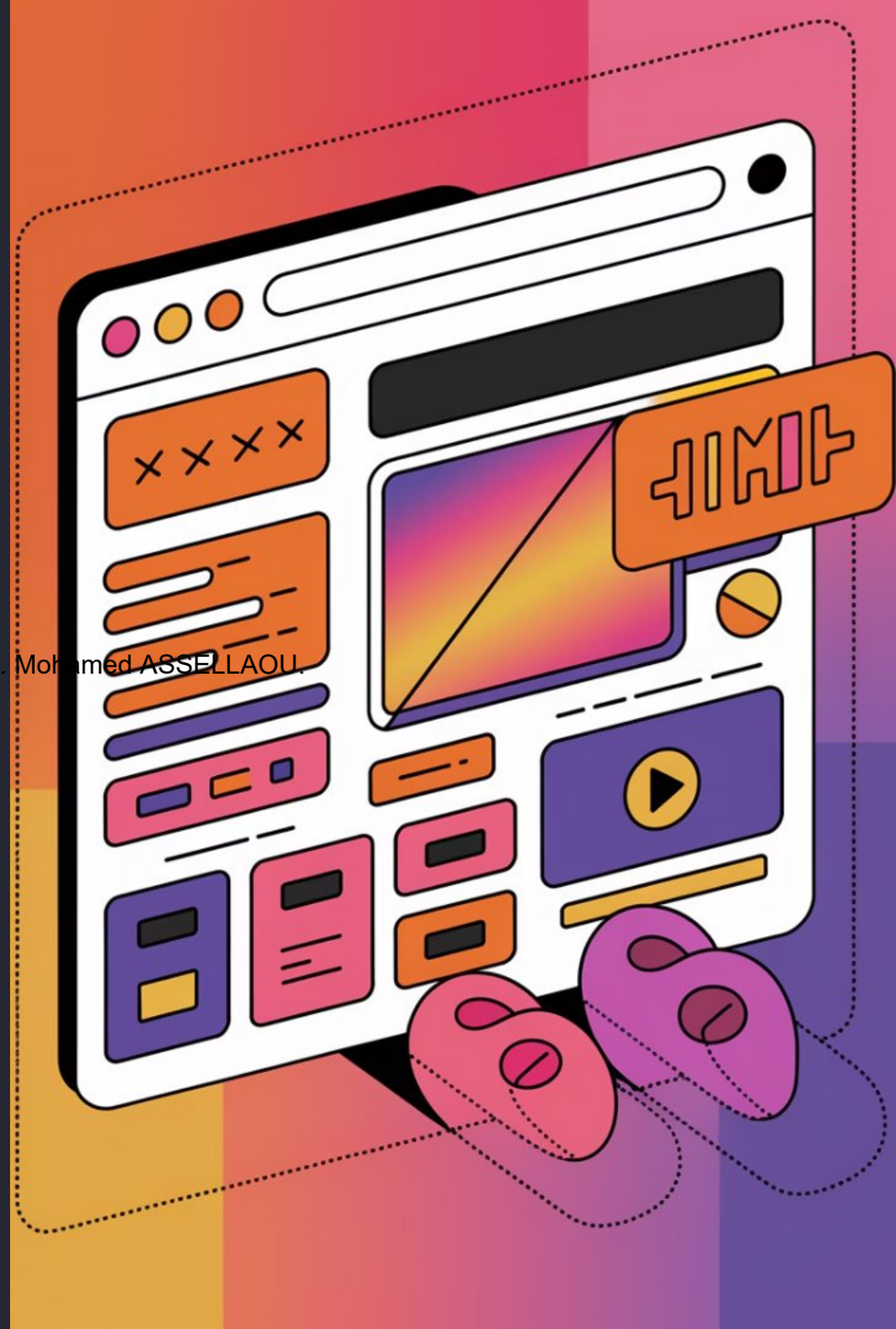
Extracting data from websites automatically

Why

Access data without API

How

BeautifulSoup, Scrapy, Selenium



Scrape Responsibly

Check Terms & Conditions

Many sites prohibit scraping

Avoid Personal Data

Privacy laws protect individual information

Respect robots.txt

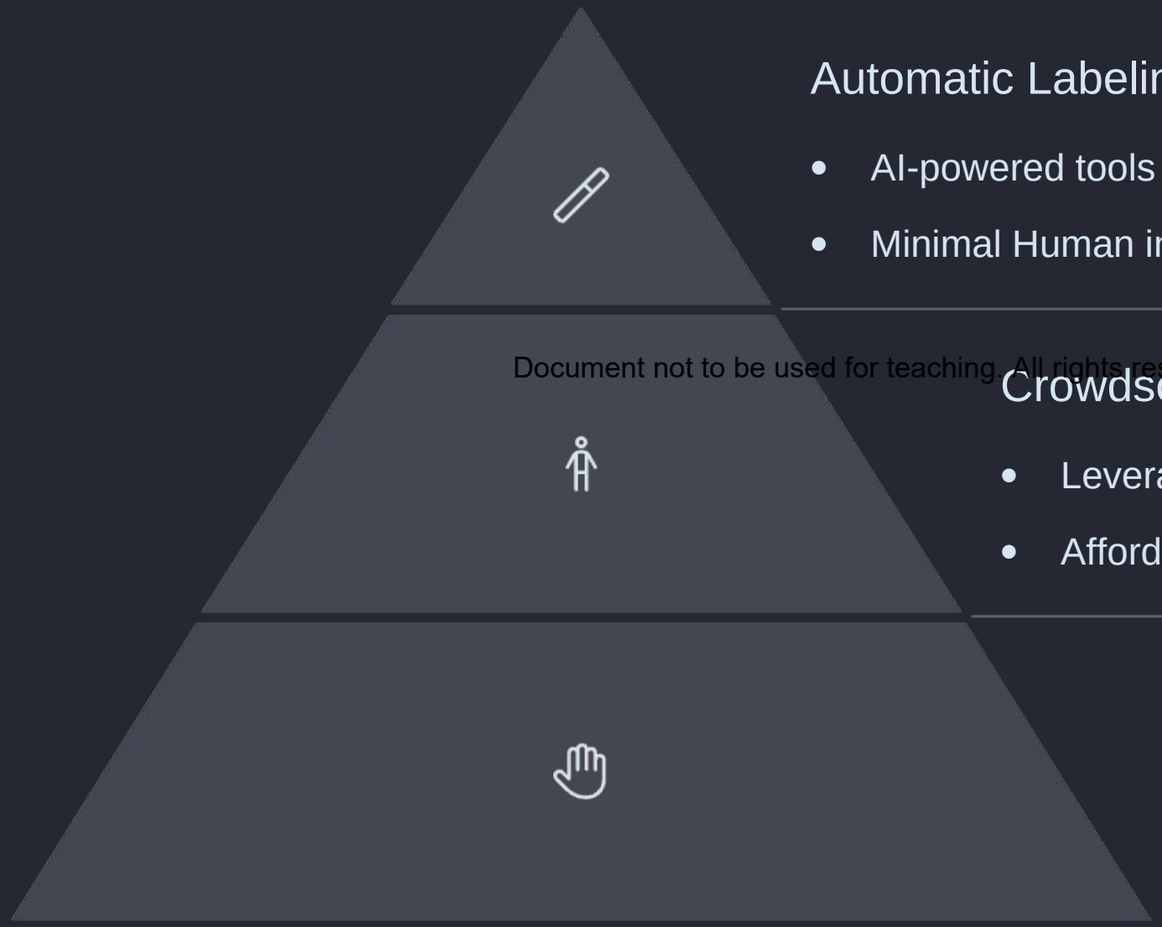
Follow site's crawling rules

Try This!

- Scrap http://books.toscrape.com/index.html
- Extract Books titles, prices and URLs
- Store the data in a DataFrame
- Save the data into a CSV file

Colab Notebook

Data Labeling approaches



Automatic Labeling

- AI-powered tools for automated data annotation
- Minimal Human intervention is still necessary to verify the accuracy of labeling.

Crowdsourced Labeling

- Leveraging a crowd of online workers to label data.
- Affordable and fast but can’t guarantee high labeling accuracy.

Manual Labeling

- Expert annotators provide precise and high-quality labels
- Time consuming.

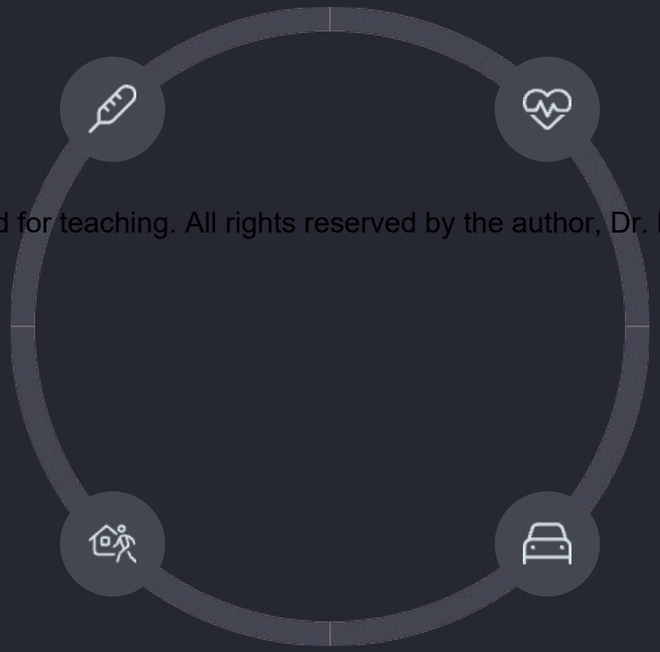
Real-Time Streams

Weather Sensors

Temperature, humidity, pressure

Health Devices

Heart rate, activity, sleep



Smart Homes

Energy usage, security, presence

Connected Vehicles

Location, speed, diagnostics

Key Takeaways

4

Major Sources

APIs, web, files, sensors

3

Data Types

Structured, semi-structured, unstructured

24/7

Availability

Continuous streams from connected devices

Keeping Your Data Safe

Decide what to store

Raw data or processed

2

Choose storage method

Files, databases, local or cloud

Implement security

Encryption, access control

Plan for backups

Prevent data loss

File-Based Storage

Local Storage

Fast access, limited space

- Personal projects
- Development work

Network Storage

Shared access, central management

- Team collaboration
- Departmental data

Cloud Storage

Accessible anywhere, scalable

- Remote teams
- Large datasets

Beyond Files

What is Object Storage?

Flat structure of data with metadata

Infinitely scalable

Designed for cloud-native applications

Data in the Cloud

Amazon S3

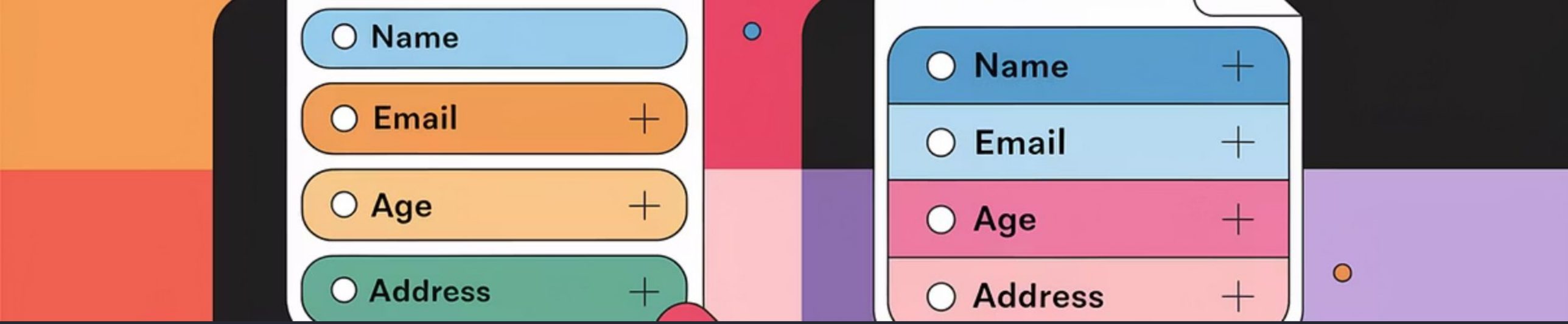
Industry standard object storage

Google Cloud Storage

Deep integration with Google services

Azure Blob Storage

Enterprise-focused solution



Organizing with Tables

Relational (SQL)

- Structured tables
- Enforced relationships
- MySQL, PostgreSQL

Document (NoSQL)

- Flexible schema
- JSON-like documents
- MongoDB, Firestore

Key-Value (NoSQL)

- Simple lookups
- High performance
- Redis, DynamoDB

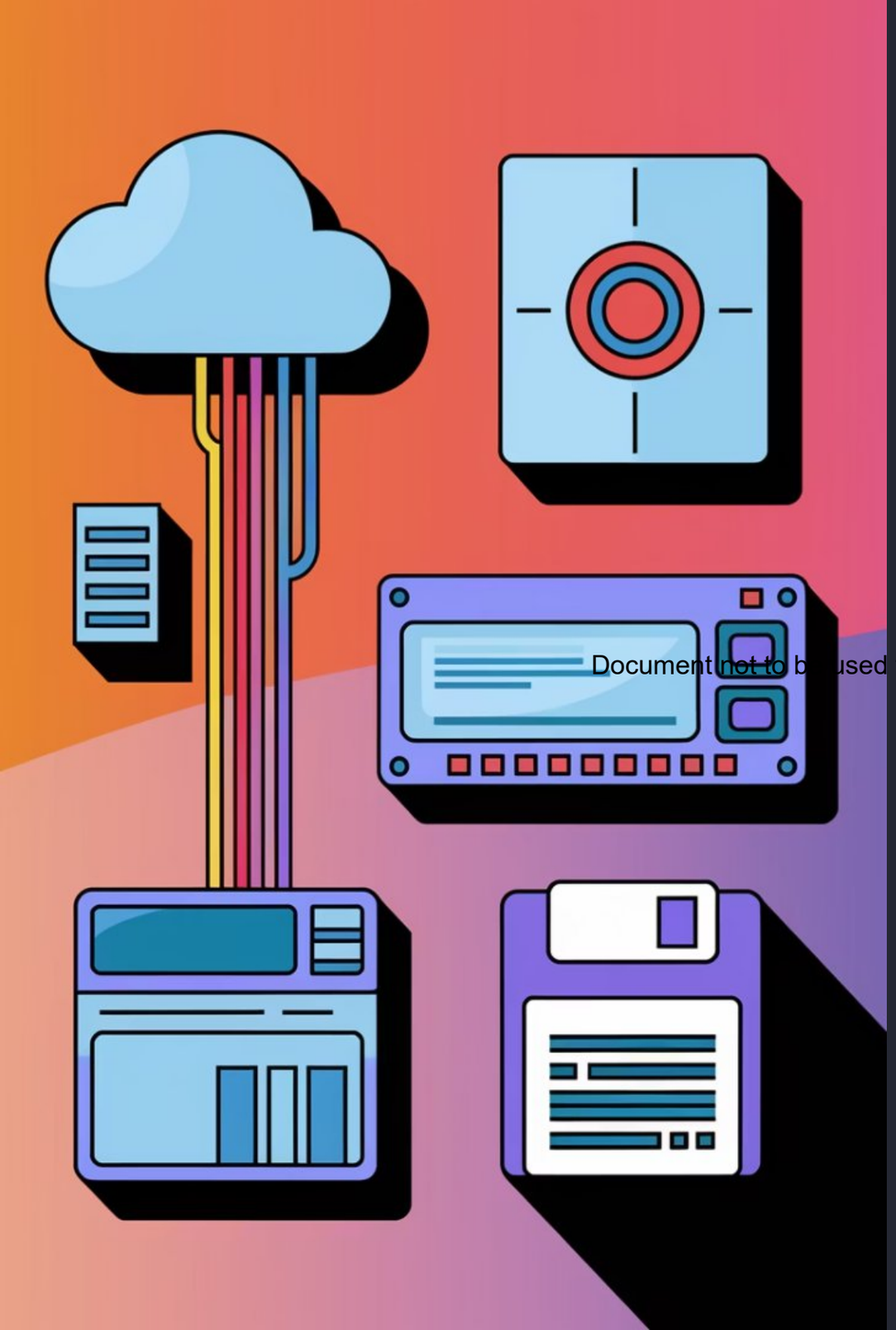
Raw vs. Refined

Data Lake

- Raw, unprocessed data
- All formats welcome
- For data scientists

Data Warehouse

- Processed, structured data
- Optimized for analysis
- Historical & current data
- For business analysts



Storage Essentials

Files

Simple, familiar

Objects

Scalable, metadata-rich

Databases

Structured, queryable

Cloud

Accessible, managed

The Messy Reality

Dirty Data

Typos, duplicates, outdated formats

Missing Information

Incomplete records, NULL values

Inconsistent Sources

Different standards, conflicting information

Data quality

Accuracy

Data correctly represents the real-world values it intends to describe.

Completeness

All required data is present without missing elements.

Relevance

Data is applicable and useful for the intended purpose.

Consistency

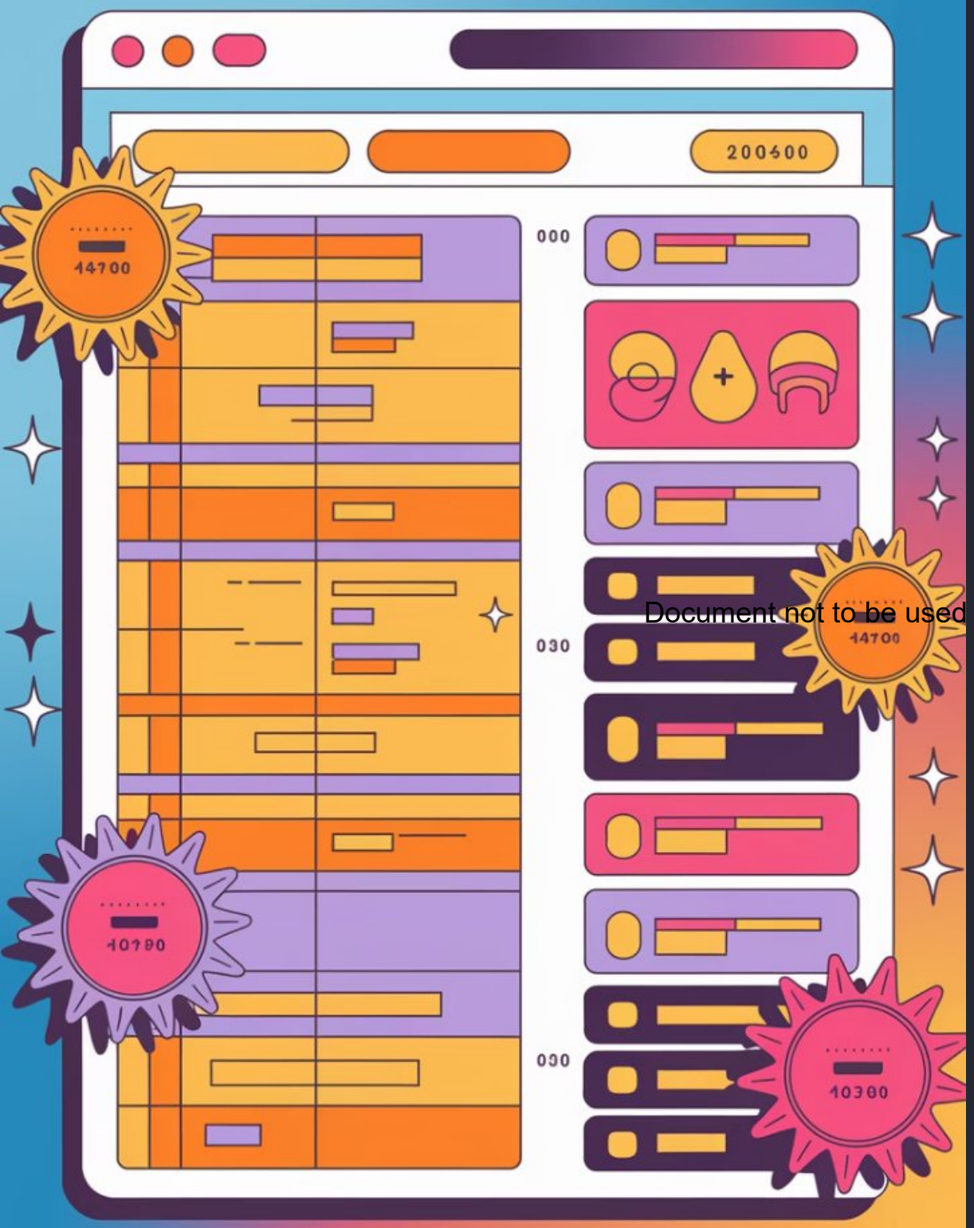
Data is uniform and does not conflict across different datasets or systems.

Timeliness

Data is up-to-date and available when needed.

Validity

Data conforms to defined formats, rules, and standards.



What Could Go Wrong?

Missing Values

NULL, NaN, empty strings

Incorrect Data

Typos, wrong formats, impossible values

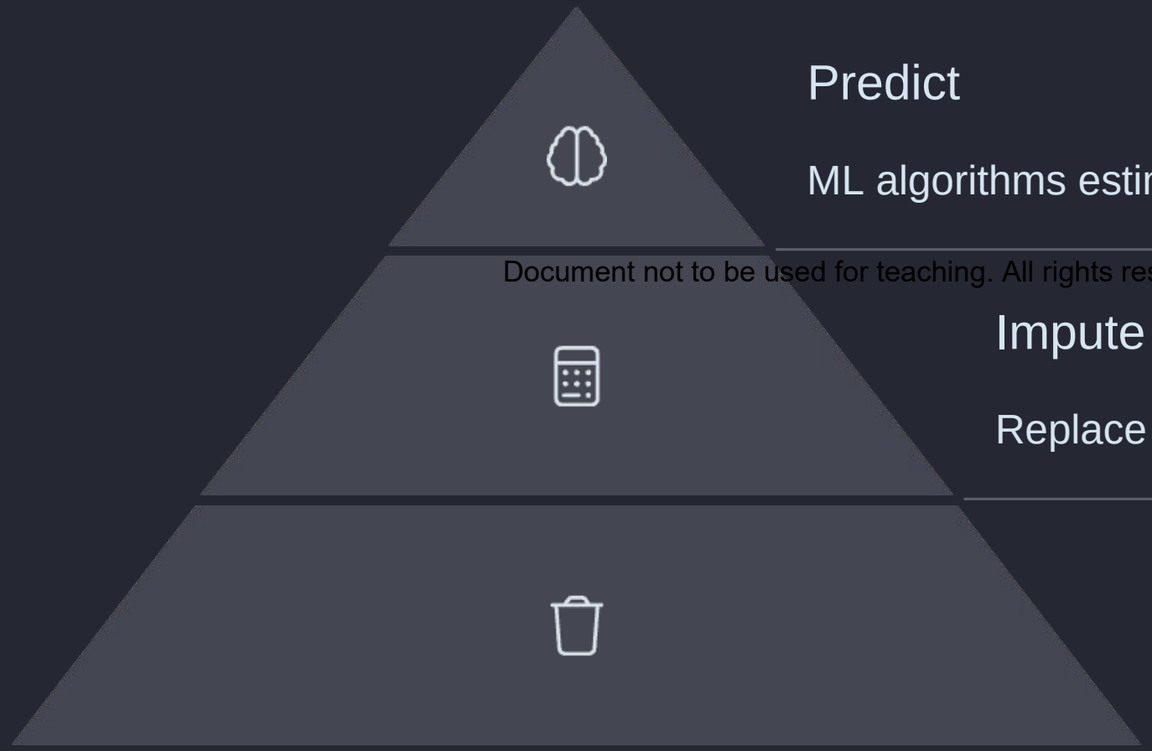
Duplicates

Same record appears multiple times

Inconsistencies

Different formats, units, or spellings

Filling the Gaps



Predict

ML algorithms estimate missing values

Impute

Replace with mean, median, or mode

Delete

Remove rows with missing critical data

Remove Duplicates

```python
import pandas as pd

names = ['Khaled', 'Fatima', 'Alex', 'Ali', 'Khaled', 'Youssef', 'Mohamed', 'Maryem']
CIN = ['JA874659', 'DC451236', 'AB969541', 'JC787414', 'JA874659', 'FA656321', 'JB965547', 'LA5436963']
ages = [23, 22, 31, 19, 23, 45, 25, 32]

data = pd.DataFrame({'name': names, 'CIN': CIN, 'age': ages})
print(data)
```

```
      name        CIN  age
0   Khaled   JA874659   23
1   Fatima   DC451236   22
2     Alex   AB969541   31
3      Ali   JC787414   19
4   Khaled   JA874659   23
5  Youssef   FA656321   45
6  Mohamed   JB965547   25
7   Maryem  LA5436963   32
```

```python
data.drop_duplicates()
```

```
      name        CIN  age
0   Khaled   JA874659   23
1   Fatima   DC451236   22
2     Alex   AB969541   31
3      Ali   JC787414   19
5  Youssef   FA656321   45
6  Mohamed   JB965547   25
7   Maryem  LA5436963   32
```

Remove Irrelevant Data

```python
import pandas as pd

names = ['Khaled', 'Fatima', 'Alex', 'Ali', 'Youssef', 'Mohamed', 'Maryem']
CIN = ['JA874659', 'DC451236', 'AB969541', 'JC787414', 'FA656321', 'JB965547', 'LA5436963']
ages = [23, 22, 31, 19, 45, 25, 32]
birthyear = [2000, 2001, 1992, 2004, 1978, 1998, 1991]

data = pd.DataFrame({'name': names, 'CIN': CIN, 'age': ages, 'birthyear': birthyear})
print(data)
```

```
      name        CIN  age  birthyear
0   Khaled   JA874659   23       2000
1   Fatima   DC451236   22       2001
2     Alex   AB969541   31       1992
3      Ali   JC787414   19       2004
4  Youssef   FA656321   45       1978
5  Mohamed   JB965547   25       1998
6   Maryem  LA5436963   32       1991
```

```python
new_data = data.drop('birthyear', axis=1)
print(new_data)
```

```
      name        CIN  age
0   Khaled   JA874659   23
1   Fatima   DC451236   22
2     Alex   AB969541   31
3      Ali   JC787414   19
4  Youssef   FA656321   45
5  Mohamed   JB965547   25
6   Maryem  LA5436963   32
```

Handling Missing Data or Null values

```python
import pandas as pd

names = [np.nan, 'Fatima', 'Alex', 'Ali', np.nan, 'Mohamed', 'Maryem']
CIN = ['JA874659', np.nan, 'AB969541', 'JC787414', 'FA656321', 'JB965547', np.nan]
ages = [23, np.nan, 31, 19, np.nan, 25, 32]

data = pd.DataFrame({'name': names, 'CIN': CIN, 'age': ages})
print(data)
```

```
      name       CIN   age
0      NaN  JA874659  23.0
1   Fatima       NaN   NaN
2     Alex  AB969541  31.0
3      Ali  JC787414  19.0
4      NaN  FA656321   NaN
5  Mohamed  JB965547  25.0
6   Maryem       NaN  32.0
```

```python
# detecting null data

print(data.isnull())
```

```
    name    CIN    age
0   True  False  False
1  False   True   True
2  False  False  False
3  False  False  False
4   True  False   True
5  False  False  False
6  False   True  False
```



```python
# droping null values

print(data.dropna())
```

```
      name       CIN   age
2     Alex  AB969541  31.0
3      Ali  JC787414  19.0
5  Mohamed  JB965547  25.0
```

```python
# filling missing values

print(data.fillna(0))
```

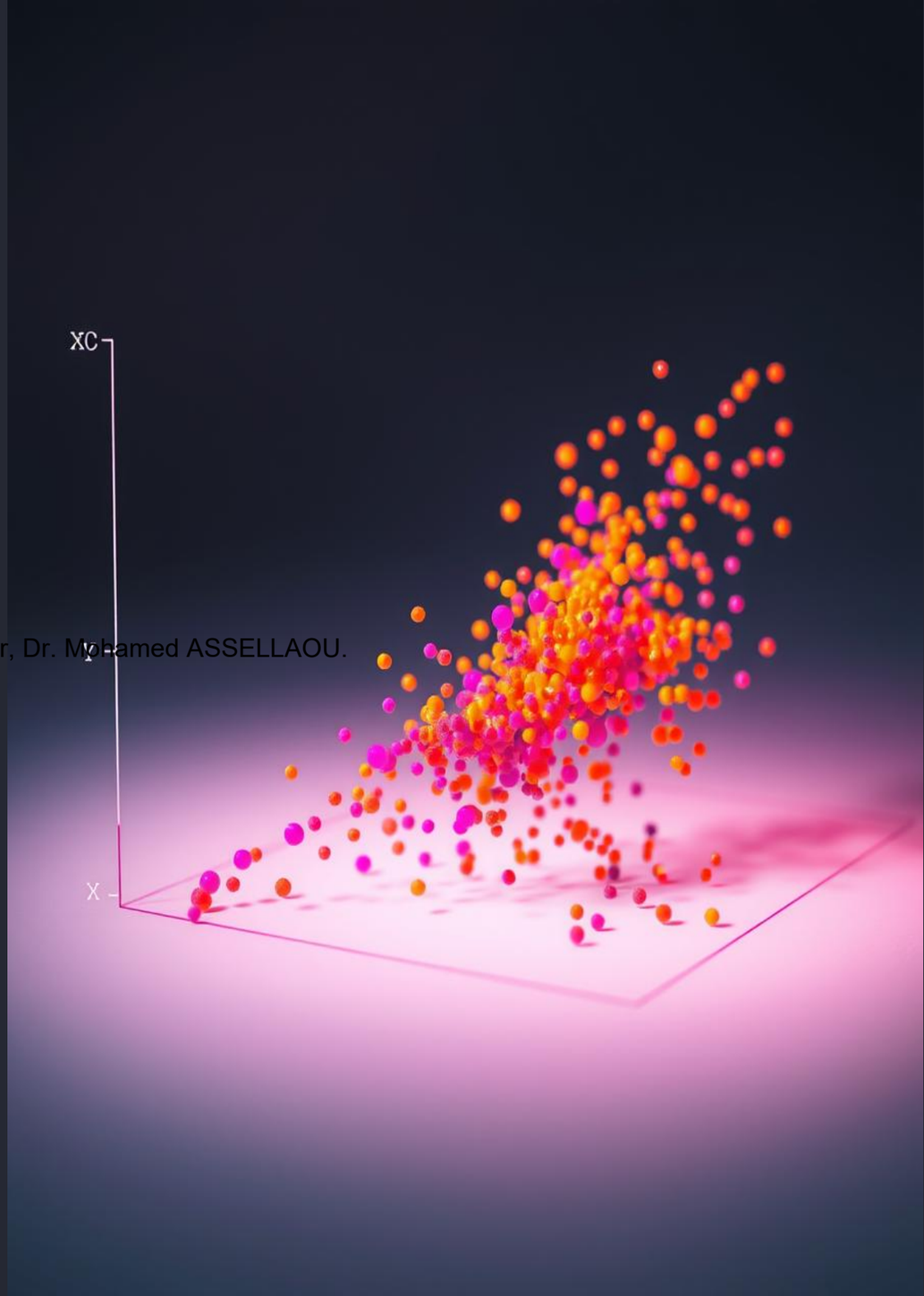
```
      name       CIN   age
0        0  JA874659  23.0
1   Fatima         0   0.0
2     Alex  AB969541  31.0
3      Ali  JC787414  19.0
4        0  FA656321   0.0
5  Mohamed  JB965547  25.0
6   Maryem         0  32.0
```

Handling outliers

Outlier removal: Removing outliers from the dataset.

Winsorization: Replacing outliers with values from the upper or lower ends of the distribution.

Data Transformations: Applying mathematical functions to reduce the effect of outliers.



Make It Uniform

Before	After	Method
5 meters, 12 feet	5m, 3.66m	Standardization
$10,000 salary	0.1 (scaled 0-1)	Min-Max Normalization
Ages: 25, 65, 18	-0.5, 1.5, -1.0	Z-score Normalization

Set It and Forget It

Raw Data

Ingestion from source

Validation

Check against rules

Transformation

Apply cleaning operations

Clean Data

Ready for analysis

Stay Tidy

Explore

Understand data issues first

Remove

Drop duplicates and outliers

Transform

Fix formats and standardize

Automate

Create reproducible pipeline

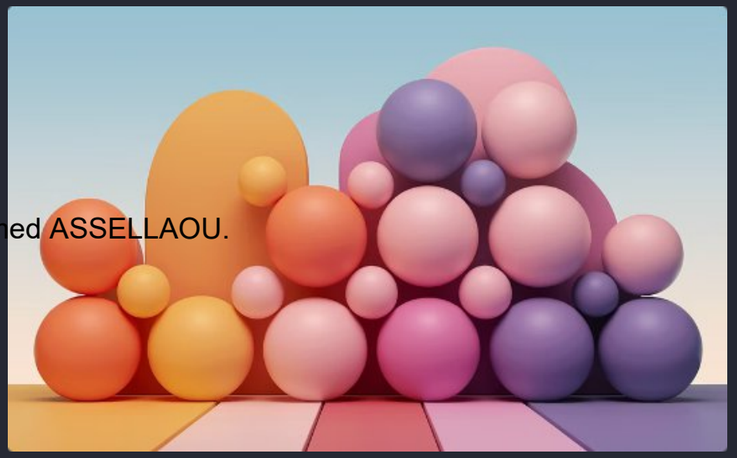
Make Data Speak

Present Insights

Find and highlight key patterns

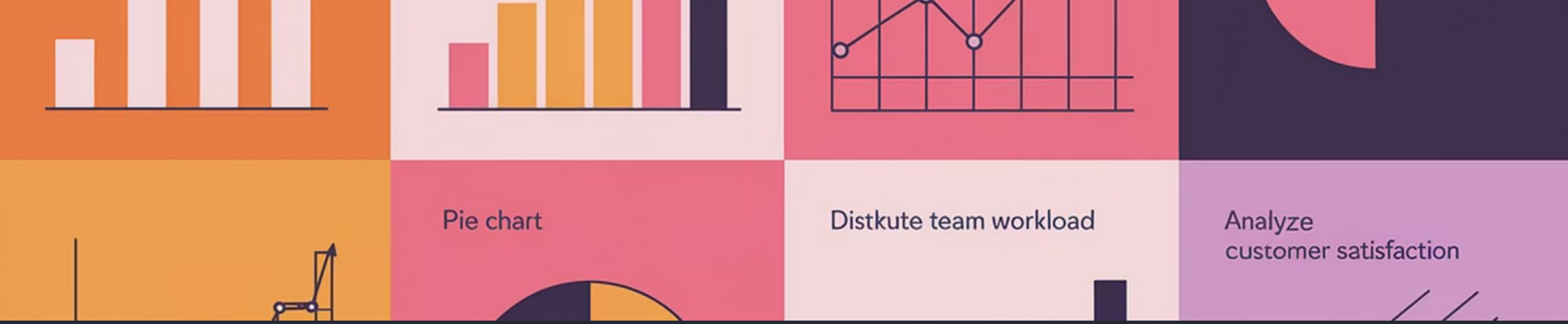
Drive Decisions

Convert data to action



Simplify Complexity

Make patterns immediately visible



Pick the Right Chart

Comparison

- Bar: categories
- Line: over time

Composition

- Pie: parts of whole
- Stacked Bar: grouped parts

Distribution

- Histogram: single variable
- Scatter: two variables

Design for Clarity

Do

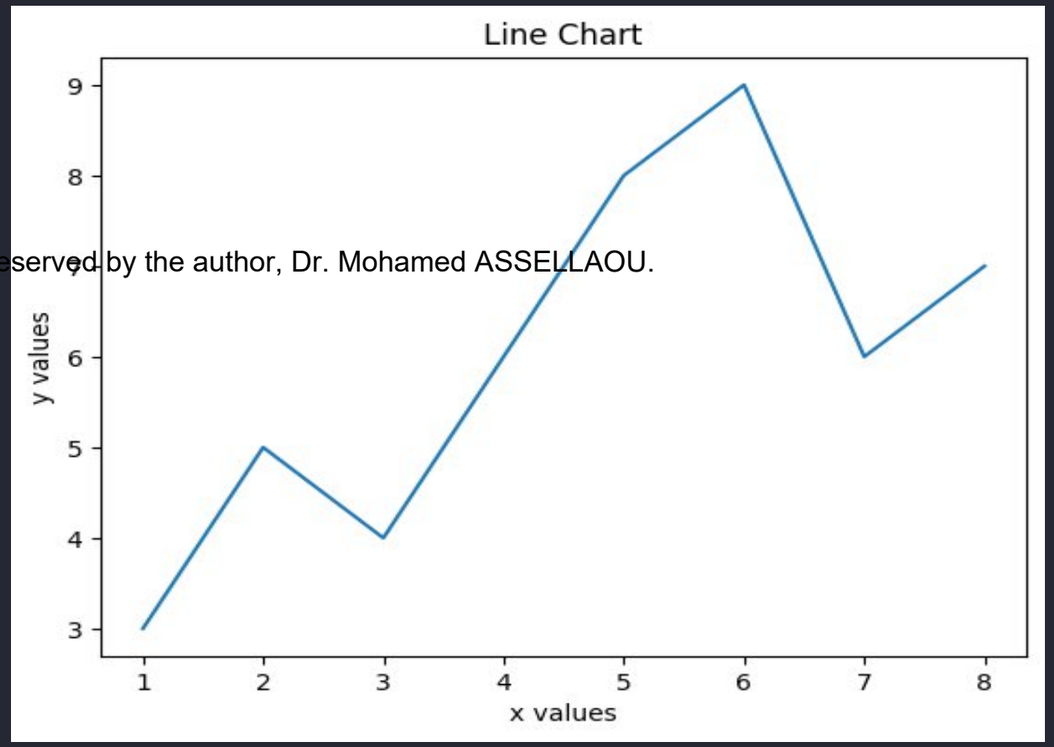
- Use clear titles
- Label axes
- Choose appropriate colors
- Keep it simple
- Show context

Don't

- Use 3D unnecessarily
- Overload with decoration
- Use too many colors
- Omit sources

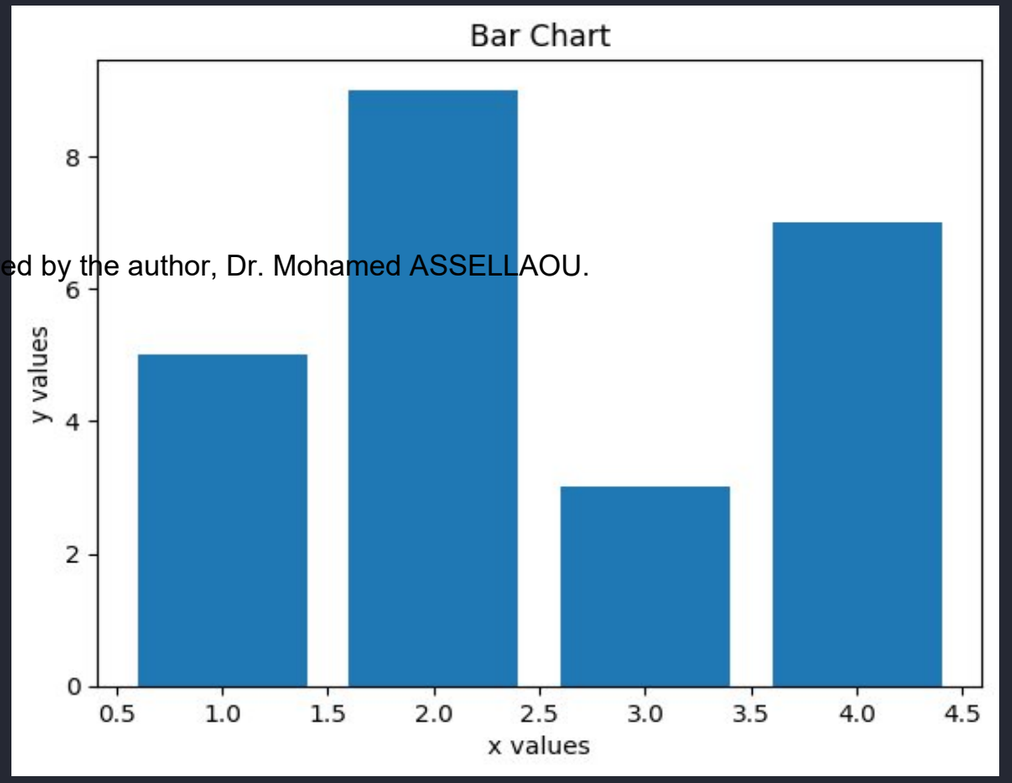
Line Chart

```
import matplotlib.pyplot as plt
import numpy as np

# line chart
x = [1,2,3,4,5,6,7,8]
y = [3,5,4,6,8,9,6,7]
plt.title("Line Chart")
plt.xlabel('x values')
plt.ylabel('y values')
plt.plot(x,y)
plt.show()
```

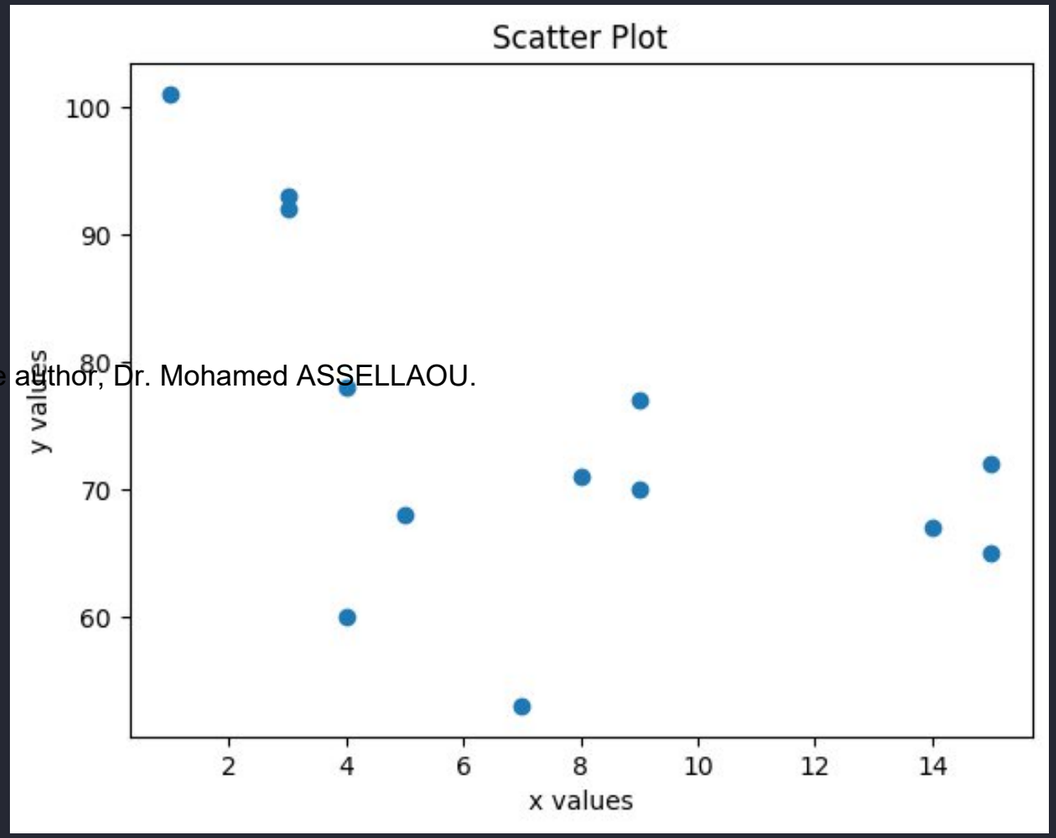
Bar Chart

```python
import matplotlib.pyplot as plt
import numpy as np

# bar chart
x = [1,2,3,4]
y = [5,9,3,7]
plt.title("Bar Chart")
plt.xlabel('x values')
plt.ylabel('y values')
plt.bar(x,y)
plt.show()
```

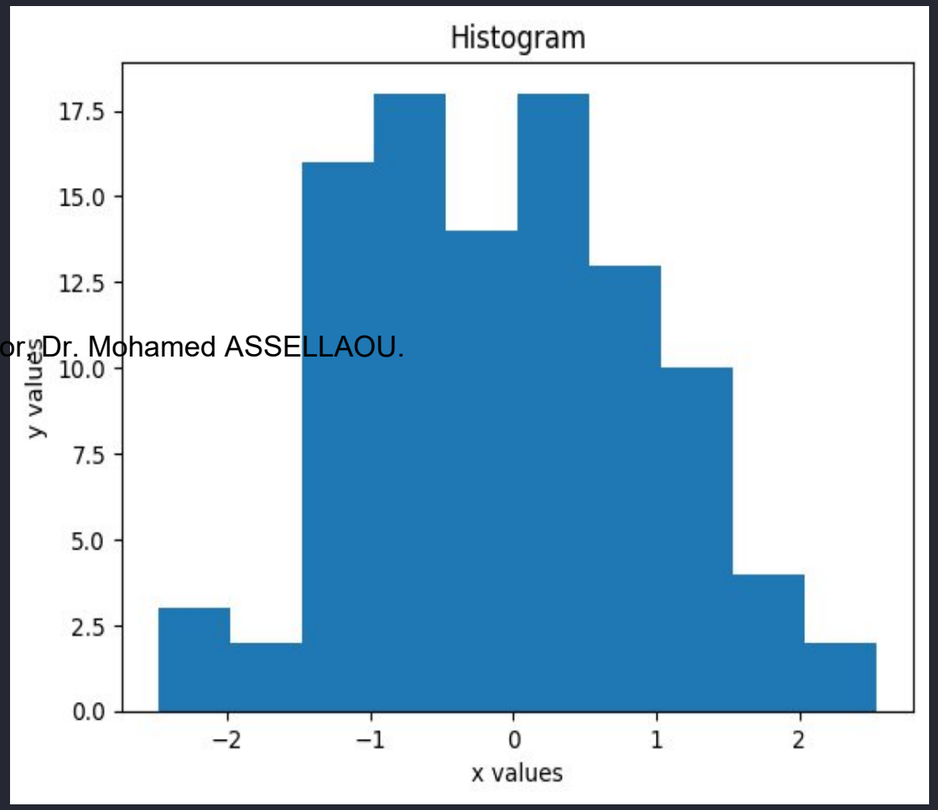
Scatter Chart

```python
import matplotlib.pyplot as plt
import numpy as np

# scatter plot
x = [4,5,7,9,3,15,1,9,3,14,15,8,4]
y = [78,68,53,77,92,72,101,70,93,67,65,71,60]
plt.title("Scatter Plot")
plt.xlabel('x values')
plt.ylabel('y values')
plt.scatter(x,y)
plt.show()
```

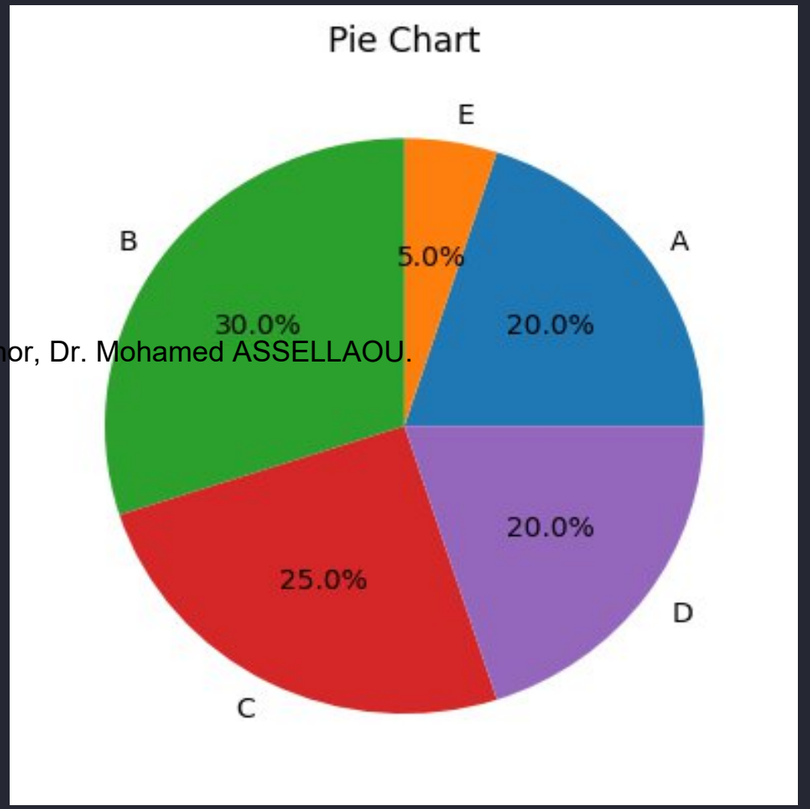
Histogram

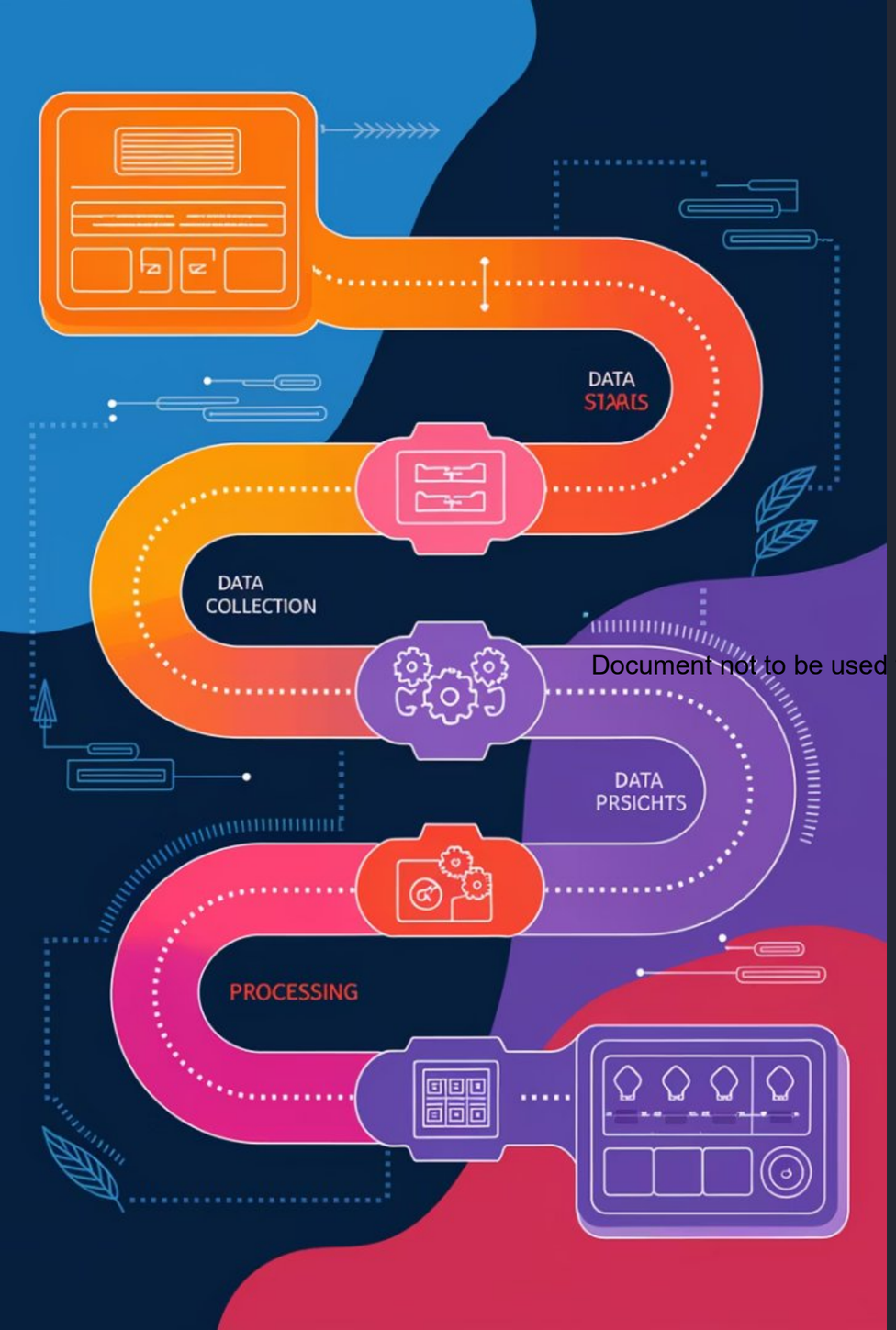
```python
import matplotlib.pyplot as plt
import numpy as np

# histogram
x = np.random.normal(0, 1, 100)
plt.hist(x)
plt.title("Histogram")
plt.xlabel('x values')
plt.ylabel('y values')
plt.show()
```

Pie Chart

```python
import matplotlib.pyplot as plt
import numpy as np

# pie chart
data = [20, 5, 30, 25, 20]
labels = ['A', 'E', 'B', 'C', 'D']
plt.pie(data, labels = labels, autopct='%1.1f%%')
plt.title("Pie Chart")
plt.show()
```



Connect It All

1 **Collect**

APIs, sensors, files

2 **Store**

Cloud, database

Clean

Fix, normalize, validate

Visualize

Charts, dashboards

Test Your Knowledge

What's typically stored in a data lake?

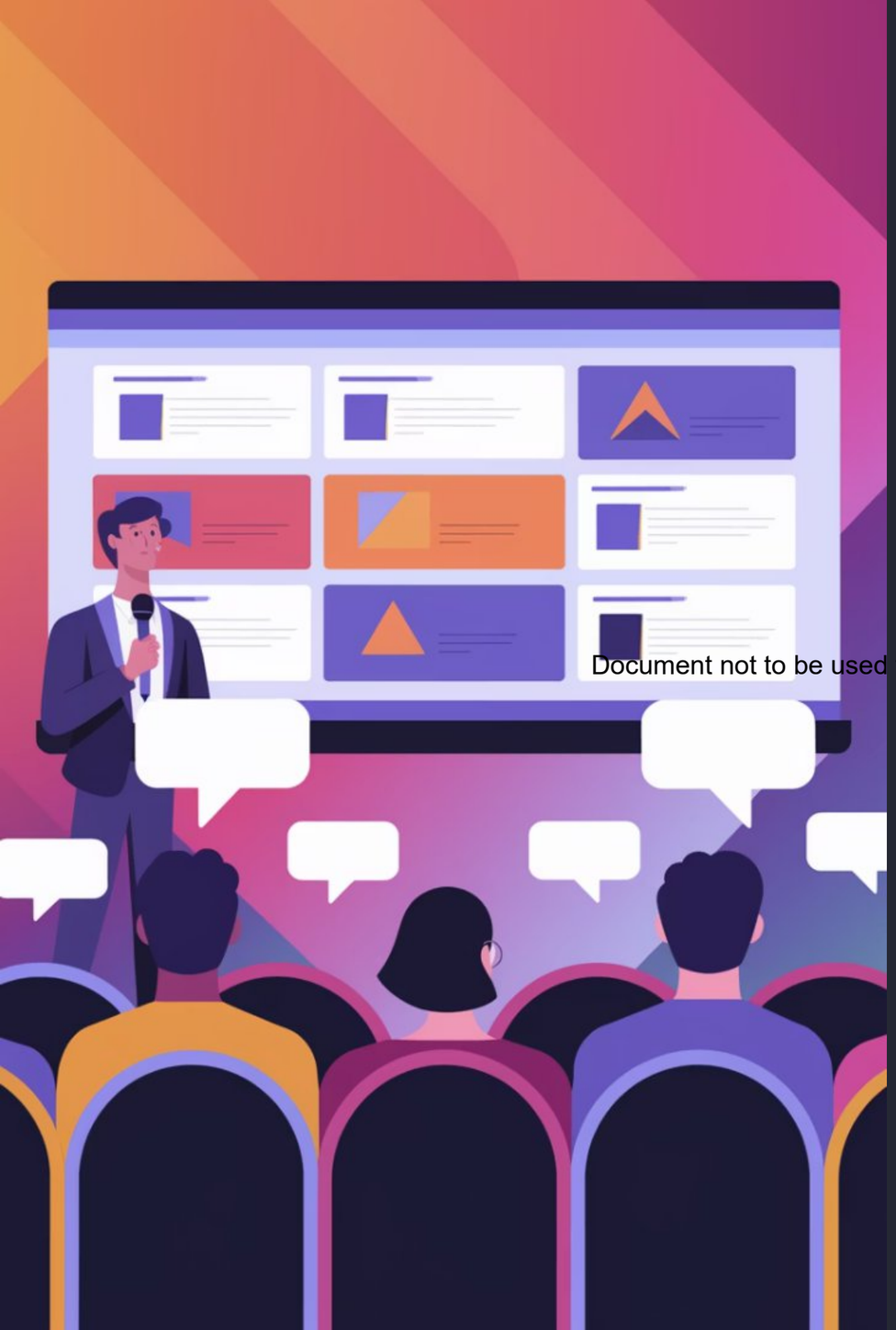
1. Only structured data
2. Raw, unprocessed data
3. Only clean data

Which is NOT a data cleaning step?

1. Removing duplicates
2. Creating dashboards
3. Standardizing formats

Best chart for parts of a whole?

1. Scatter plot
2. Line chart
3. Pie chart



Conclusion

Data is a powerful asset that requires careful handling to unlock its full potential.

Effective data management involves collection, storage, cleaning, visualization, and security.

Collaboration, continuous learning, and community support are key to success in data science.