Open in Colab

DTS 201 – Introduction to Data Science

Week 1 Practical (PH): Exploring Sample Datasets and Data Types

Instructor: Dr. Sakinat Folorunso

Title: Associate Professor of AI Systems and FAIR Data Department: Computer Sciences, Olabisi Onabanjo University, Ago-Iwoye, Ogun State, Nigeria

Course Code: DTS 201
Practical Week: 1 – Introduction to Data and File Types

🎯 Practical Learning Objectives

By the end of this practical session, you should be able to:

- Open and explore sample datasets stored in **CSV** and **Excel** formats.
- Inspect the **rows, columns, data types, and basic structure** of a dataset.
- Distinguish between **structured**, **semi-structured**, and **unstructured** data using real examples.
- Describe, in your own words, what type of data each sample file represents and why.

> ✅ This notebook is **student-centred**: you will run the code, modify it, and answer short reflection questions in the provided spaces.

1. Getting Started: Where Is Your Data?

You can run this notebook in one of two common ways:

1. **Google Colab**
   - Upload your dataset files (e.g., `.csv`, `.xlsx`) to your Google Drive or directly to Colab.
   - If your dataset is saved in Google Drive, you'll need to **mount Drive first.**
2. **Local Jupyter Notebook (on your laptop)**
   - Save the dataset files in a folder on your computer.
   - You will need to provide the **correct file path** to the dataset (e.g., `"data/students_scores.csv"`).

> 🖍️ You do **not** need to have a specific dataset yet. Your lecturer may provide sample files such as `students_scores.csv` or `sales_data.xlsx`. You can also practice with any simple CSV/Excel file you already have.

1.1 (Optional) Mounting Google Drive in Colab

In [ ]:
```
# NOTE: Run this cell ONLY if you are using Google Colab and your data is stored in

# Import the drive module from google.colab to allow access to Google Drive
from google.colab import drive  # Provides tools to connect Colab to your Google Dr

# Mount your Google Drive so that it appears as a folder within the Colab environme
drive.mount('/content/drive')  # After running, follow the on-screen instructions t

# After mounting, your files will be accessible under the path '/content/drive/MyDr
# Example: data_path = '/content/drive/MyDrive/Datasets/students_scores.csv'
```

2. Importing Required Libraries

In [1]:
```
# Import the pandas library for working with tabular data (rows and columns)
import pandas as pd

# Import numpy for numerical operations (we may need it later)
import numpy as np

# Import matplotlib for basic plotting (optional for this week but useful generally
import matplotlib.pyplot as plt

# Configure matplotlib so that any plots appear inside the notebook
%matplotlib inline
```

3. Exploring a Sample CSV Dataset

In this section, you will:

- Load a sample **CSV** file into a pandas DataFrame.
- Look at the first few rows.
- Inspect the structure of the data (columns, types).

> 🪛 **Instruction:**
>
> - Replace `"students_scores.csv"` with the actual name of your CSV file.
> - Make sure the file is in the same folder as this notebook, or provide the full path.

In [2]:
```
# Define the path or file name of your CSV dataset
# If your file is in the same folder as this notebook, you can just use the file na
# Example: data_path_csv = 'students_scores.csv'
data_path_csv = '../data/student_score.csv'
  # TODO: change this to your actual CSV file name

# Load the CSV file into a pandas DataFrame
df_csv = pd.read_csv(data_path_csv)  # Reads the CSV file and stores the data in th

# Display the first 5 rows of the dataset
df_csv.head()  # Helps you quickly see what the data looks like (columns and sample
```

Out[2]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30

> 📝 **Student Task 1:**
>
> - What does each **row** represent in this dataset (e.g., a student, a transaction, a product)?
> - Name at least **three columns** and say what kind of information they contain (e.g., score, age, grade).
> - Write your answers in a separate markdown cell or on your answer sheet.

1. What does each row represent? Each row represents a student's study session record showing the relationship between study hours per day and corresponding exam scores achieved.

2. Name at least three columns: Since this dataset contains only 2 columns:

- **Hours** - Study hours per day (numerical - float type)
- **Scores** - Exam scores achieved (numerical - integer type)

3.1 Understanding the Structure of the CSV Dataset

In [4]:
```
# Check the number of rows and columns in the CSV dataset
df_csv.shape  # Returns a tuple: (number_of_rows, number_of_columns)
```

Out[4]:
```
(25, 2)
```

In [5]:
```
# Display information about the CSV dataset
# This includes column names, data types, and how many non-missing values are in ea
```

```
df_csv.info()  # Helps you understand the structure and quality of the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Hours   25 non-null     float64
 1   Scores  25 non-null     int64
dtypes: float64(1), int64(1)
memory usage: 532.0 bytes
```

In [6]:
```
# Display the data types of each column explicitly
df_csv.dtypes  # Shows whether each column is integer, float, object (text), etc.
```

Out[6]:
```
Hours     float64
Scores      int64
dtype: object
```

> 🖊️ **Student Reflection 2:**
>
> - How many rows and columns does your CSV dataset have?
> - Which columns are **numerical** (e.g., integers, floats)?
> - Which columns are **categorical/text** (e.g., names, categories)?

1. How many rows and columns does your CSV dataset have?

- Rows: 25 rows
- Columns: 2 columns
- Shape: (25, 2)

2. Which columns are numerical?

- Hours: float64
- Scores: int64
- All columns are numerical

3. Which columns are categorical/text? None - This dataset contains no categorical or text columns.

4. Data Types: Structured, Semi-Structured, and Unstructured

Before we explore Excel files, let's recall three important categories of data:

1. **Structured Data**
   - Organized in a clear tabular format (rows and columns).
   - Each column has a defined data type (e.g., number, text, date).

- Examples: Excel tables, CSV files, relational databases.

2. **Semi-Structured Data**

   - Contains some organization but not strictly tabular.
   - Uses tags or markers (e.g., key–value pairs).
   - Examples: JSON, XML, some log files.

3. **Unstructured Data**

   - Has no fixed format or layout.
   - Examples: raw text documents, images, audio, video.

> 💡 In this practical, your CSV and Excel files are examples of **structured data** because they are arranged in rows and columns.

5. Exploring a Sample Excel Dataset

Now, we will repeat the process with an **Excel** file (`.xlsx`).

> 🔧 **Instruction:**
>
> - Replace `"sales_data.xlsx"` with your actual Excel file name.
> - If the file has multiple sheets, you can specify the sheet name with the `sheet_name` argument.

In [8]:
```
# Define the path or file name of your CSV dataset (changed from Excel)
# Example: data_path_csv = 'Sales_data.csv'
data_path_csv = '../data/Sales_data.csv'
# TODO: change this to your actual CSV file name

# Load the CSV file into a pandas DataFrame
# Using read_csv instead of read_excel since we're using CSV format
df_excel = pd.read_csv(data_path_csv)  # Reads the CSV file and stores the data in

# Display the first 5 rows of the CSV dataset
df_excel.head()  # Helps you check the structure and contents of the CSV data
```

Out[8]:

	product_id	product_name	category	discounted_
0	B07JW9H4J1	Wayona Nylon Braided USB to Lightning Fast Cha...	Computers&Accessories|Accessories&Peripherals|...	
1	B098NS6PVG	Ambrane Unbreakable 60W / 3A Fast Charging 1.5...	Computers&Accessories|Accessories&Peripherals|...	
2	B096MSW6CT	Sounce Fast Phone Charging Cable & Data Sync U...	Computers&Accessories|Accessories&Peripherals|...	
3	B08HDJ86NZ	boAt Deuce USB 300 2 in 1 Type-C & Micro USB S...	Computers&Accessories|Accessories&Peripherals|...	
4	B08CF3B7N1	Portronics Konnect L 1.2M Fast Charging 3A 8 P...	Computers&Accessories|Accessories&Peripherals|...	

> 🖊 **Student Task 3:**
>
> - Compare the Excel dataset with the CSV dataset you loaded earlier.
> - How are they similar? How are they different (e.g., columns, data types, context)?

Similarities:

- Both are structured data with rows and columns
- Both loaded successfully into pandas DataFrames
- Both have defined column headers

Differences:

- **Size**: CSV has 25 rows, 2 columns vs Excel has 1,465 rows, 16 columns
- **Data Types**: CSV is purely numerical (float64, int64) vs Excel has mixed types (text, numerical)
- **Context**: CSV shows study hours vs scores vs Excel contains e-commerce product data
- **Complexity**: CSV is simple relationship data vs Excel has complex business data with reviews, prices, ratings

5.1 Inspecting the Structure of the Excel Dataset

In [9]:
```
# Check the number of rows and columns in the Excel dataset
df_excel.shape  # Returns (number_of_rows, number_of_columns) for the Excel data
```

Out[9]:
```
(1465, 16)
```

In [10]:
```
# Display detailed information about the Excel dataset
df_excel.info()  # Shows column names, data types, and non-null counts for each col
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1465 entries, 0 to 1464
Data columns (total 16 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   product_id           1465 non-null   object
 1   product_name         1465 non-null   object
 2   category             1465 non-null   object
 3   discounted_price     1465 non-null   object
 4   actual_price         1465 non-null   object
 5   discount_percentage  1465 non-null   object
 6   rating               1465 non-null   object
 7   rating_count         1463 non-null   object
 8   about_product        1465 non-null   object
 9   user_id              1465 non-null   object
 10  user_name            1465 non-null   object
 11  review_id            1465 non-null   object
 12  review_title         1465 non-null   object
 13  review_content       1465 non-null   object
 14  img_link             1465 non-null   object
 15  product_link         1465 non-null   object
dtypes: object(16)
memory usage: 183.3+ KB
```

In [11]:
```
# Display the data types of each column in the Excel dataset
df_excel.dtypes  # Shows whether each column is integer, float, object (text), etc.
```

Out[11]:
```
product_id             object
product_name           object
category               object
discounted_price       object
actual_price           object
discount_percentage    object
rating                 object
rating_count           object
about_product          object
user_id                object
user_name              object
review_id              object
review_title           object
review_content         object
img_link               object
product_link           object
dtype: object
```

> 🖊️ **Student Reflection 4:**
>
> - Are the data types in the Excel dataset mostly similar to those in the CSV file?
> - Identify one column that you think is **categorical** and one that is **numerical** in the Excel dataset.

Are the data types in the Excel dataset mostly similar to those in the CSV file? No, they are quite different. The CSV dataset has only numerical data types (float64 and int64), while the Excel dataset has mostly object/text data types with some numerical columns.

Identify one categorical and one numerical column in the Excel dataset:

- **Categorical**: product_name (contains text descriptions of products)
- **Numerical**: rating (contains numerical rating values like 4.2, 4.0, 3.9)

6. Classifying Datasets by Type (Activity)

Using the definitions from Section 4, classify the following as **structured**, **semi-structured**, or **unstructured**:

1. The CSV dataset you loaded (`df_csv`).
2. The Excel dataset you loaded (`df_excel`).
3. A folder of .jpg images.
4. A collection of WhatsApp chat exports in `.txt` format.
5. A JSON file containing product data (e.g., from an API).

> 🖊️ **Student Task 5:**
>
> - For each of the five items above, write:
>   - The **type of data** (structured / semi-structured / unstructured).
>   - A **one-line justification** of your choice.

Student Task 5: Classifying datasets by type

1. The CSV dataset you loaded (df_csv):

- **Type:** Structured
- **Justification:** Organized in clear rows and columns with defined data types (Hours, Scores)

2. The Excel dataset you loaded (df_excel):

- **Type:** Structured

- **Justification:** Tabular format with consistent columns and defined schema for product data

3. A folder of .jpg images:

- **Type:** Unstructured
- **Justification:** No fixed format or layout, just binary image data without organized structure

4. A collection of WhatsApp chat exports in .txt format:

- **Type:** Unstructured
- **Justification:** Free-form text without consistent structure or predefined fields

5. A JSON file containing product data (e.g., from an API):

- **Type:** Semi-structured
- **Justification:** Uses key-value pairs and nested objects but lacks strict tabular organization

7. Summary and Reflection (Week 1 PH)

In this practical, you:

- Loaded **CSV** and **Excel** files into pandas DataFrames.
- Used `.head()`, `.info()`, `.shape`, and `.dtypes` to understand the structure of your data.
- Reviewed the concepts of **structured**, **semi-structured**, and **unstructured** data.
- Practised classifying different real-world data examples by type.

> 🧠 **Final Reflection (Write in your own words):**
>
> - What was the most **surprising** thing you learned about data formats today?
> - Which method or function (e.g., `.head()`, `.info()`) do you think you will use the most in future practicals, and why?

What was the most surprising thing you learned about data formats today? The most surprising thing was how different datasets can have vastly different structures and complexity levels - from simple 2-column numerical data to complex 16-column business data with mixed types, yet both are considered structured data.

Which method or function do you think you will use the most in future practicals, and why? I think I will use `.info()` the most because it provides a comprehensive overview of the dataset including shape, column names, data types, and missing values all in one command, making it essential for initial data exploration.