# DATA Preprocessing The way to Machine Learning

Abdenour Bouziane
Baaziz Mohammed Chaker

# CONTENTS OF THIS SESSION

Here's what you'll find in this **slides** :

1. Introduction to Python
2. Introduction to Data Preprocessing :
   a. Data cleaning
   b. Data transformation
   c. Data reduction
   d. Data discretization
3. Hands-on Examples
   - Importing libraries: pandas, numpy
   - Loading data: CSV, Excel, etc.
   - Handling missing values:
   - Data transformation:
   - Data reduction:
   - Data discretization:
4. Conclusion and Next Steps

# Why Python for Data Preprocessing?

- Python has a rich ecosystem of libraries specifically designed for data manipulation and analysis, such as pandas, NumPy, and scikit-learn.
- It offers easy-to-understand syntax, making it accessible for beginners without sacrificing power and flexibility.
- Python's popularity in the data science community means there is extensive documentation, tutorials, and community support available.

# Basic Python Syntax

1. Variables and Data Types:

main.py

```python
# Variables
x = 10
name = "Abdou"
is_student = True

# Data types
integer_num = 10
float_num = 3.14
string_text = "Hello, world!"
boolean_value = True
```

# Basic Python Syntax

2. Basic Arithmetic Operations:

main.py

```python
# Arithmetic operations
a = 10
b = 5
# Addition
sum_result = a + b
# Subtraction
difference_result = a - b
# Multiplication
product_result = a * b
# Division
division_result = a / b
# Modulus (remainder)
modulus_result = a % b
# Exponentiation
exponentiation_result = a ** b
```

# Basic Python Syntax

3. Basic Data Structures:

main.py

```python
# Lists
my_list = [1, 2, 3, 4, 5]

# Tuples
my_tuple = (1, 2, 3)

# Dictionaries
my_dict = {"name": "John", "age": 30, "city": "New York"}

```

# Basic Python Syntax

4. Functions:

main.py

```python
# Function definition
def greet(name):
    return "Hello, " + name + "!"

# Function call
greeting = greet("Alice")
print(greeting)
```

# Basic Python Syntax

5. Control Flow Statements:

main.py

```python
# If-else statement
x = 10
if x > 5:
    print("x is greater than 5")
else:
    print("x is not greater than 5")

# For loop
for i in range(5):
    print(i)

# While loop
counter = 0
while counter < 5:
    print(counter)
    counter += 1
```

# Introduction to Data Preprocessing

# What is Data Preprocessing ?

Data preprocessing is a crucial step in the data analysis pipeline that involves cleaning, transforming, and preparing raw data into a format suitable for further analysis. It aims to improve the quality of data, enhance its interpretability, and facilitate the performance of machine learning algorithms.

# Why is Data Preprocessing important ?

- **Improves Data Quality**: Raw data often contains errors, outliers, and missing values. Preprocessing helps identify and rectify these issues, ensuring that the data is accurate and reliable.
- **Enhances Interpretability**: Preprocessed data is easier to understand and interpret, making it more accessible for analysis and decision-making.
- **Prepares Data for Analysis**: Many machine learning algorithms require data to be in a specific format. Preprocessing ensures that data meets these requirements, enabling the application of various analytical techniques.

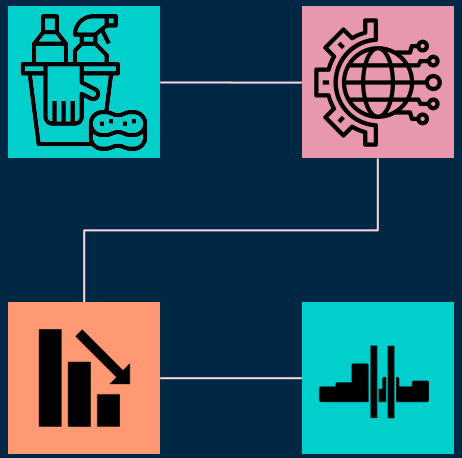
# Steps involved in Data Preprocessing

Data Cleaning

Data Transformation

Data Exploration

Data Discretization



# 1. Data Exploration

Data exploration is the process of examining and understanding the structure, patterns, and relationships within a dataset before performing further analysis. It involves summarizing the data using descriptive statistics and visualizing it through charts and graphs.

```python
import pandas as pd

# Load data
data = pd.read_csv('data.csv')

# Data Exploration
summary_stats = data.describe()
data.head()
data.tail()
data.colum
data.dtypes
print(data['data column'].describe())
print(data['column'])
data.shape()
```

# 2. Data Cleaning

Identifying and handling missing values, outliers, and inconsistencies in the data.

main.py

```python
import pandas as pd

# Load data
data = pd.read_csv('data.csv')

# Check for missing values
missing_values = data.isnull().sum()

# Handle missing values (e.g., fill with mean)
data.fillna(data.mean(), inplace=True)

# Remove duplicate records
data.drop_duplicates(inplace=True)
```

# Binning Technique

Data binning, bucketing is a data pre-processing method used to minimize the effects of small observation errors. The original data values are divided into small intervals known as bins and then they are replaced by a general value calculated for that bin. This has a smoothing effect on the input data and may also reduce the chances of overfitting in the case of small datasets

There are 2 methods of dividing data into bins:

1. Equal Frequency Binning: bins have an equal frequency.
2. Equal Width Binning : bins have equal width with a range of each bin are defined as [min + w], [min + 2w] .... [min + nw] where w = (max – min) / (no of bins).

# Binning Technique

1. Equal frequency

Suppose we have a dataset with the following values:

Values=[5,8,12,15,18,20,22,25,28,30]

We want to perform equal frequency binning with 3 bins.

# Binning Technique

1. Equal frequency

    1. **Sort the Data**:

       Sorted Values=[5,8,12,15,18,20,22,25,28,30]

    2. **Determine Number of Observations per Bin**:
    3. Total Observations=10 | Target Frequency per Bin=310≈3.33
    4. **Identify Bin Boundaries**:
        - Bin 1: [5,12)
        - Bin 2: [12,22)
        - Bin 3: [22,30]

# Binning Technique

1. Equal frequency

    5. Assign Values to Bins:

        - Values 5,8,12 belong to Bin 1.
        - Values 15,18,20 belong to Bin 2.
        - Values 22,25,28,30 belong to Bin 3.

    6. Handle Ties:

        - Ties can be handled by either assigning them to the same bin or distributing them evenly across adjacent bins, depending on the implementation.

# Binning Technique

1. Equal Width

    1. Calculate Range of Values:

       $\text{Range} = \text{Max}(\text{Values}) - \text{Min}(\text{Values}) = 30 - 5 = 25$

    2. Calculate Bin Width: $\text{Bin Width} = 25 \div 3 \approx 8.33$
    3. Define Bin Edges:
        - Bin 1: [5,13.33)
        - Bin 3: [21.67,30]

# Binning Technique

1. Equal Width

    4. Assign Values to Bins:

        i. Values 5,8,12 belong to Bin 1.
        ii. Values 15,18,20 belong to Bin 2.
        iii. Values 22,25,28,30  belong to Bin 3.

    5. Handle Edge Cases:

        iv. Values equal to the minimum or maximum may be included in the bin adjacent to them or assigned to a special bin, depending on the implementation.

# 3. Data Transformation

Converting and standardizing data into a suitable format for analysis. This may involve encoding categorical variables, scaling numerical features, and normalizing data distributions.

main.py

```python
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Encode categorical variables
label_encoder = LabelEncoder()
data['category_encoded'] = label_encoder.fit_transform(data['category'])

# Scale numerical features
scaler = StandardScaler()
data[['numerical_feature1', 'numerical_feature2']] = scaler.fit_transform(data[['numerical_feature1', 'numerical_feature2']])
```

# 4. Data Discretization

Grouping continuous data into discrete intervals or categories.

main.py

```python
# Bin numerical data
data['binned_feature'] = pd.cut(data['numerical_feature'], bins=3, labels=['Low', 'Medium', 'High'])
```

Do you have any questions?

abdenour.bouziane@ensia.edu.dz

mohammed.baaziz@ensia.edu.dz

# THANKS

# Resources

Python : https://github.com/AbderrahimRezki/PythonWorkshop.git

More about Data preprocessing :
https://www.geeksforgeeks.org/data-preprocessing-in-data-mining/

Github repository to work on :
https://github.com/AbdenourBouziane/ETC-Data-Preprocessing-Workshop