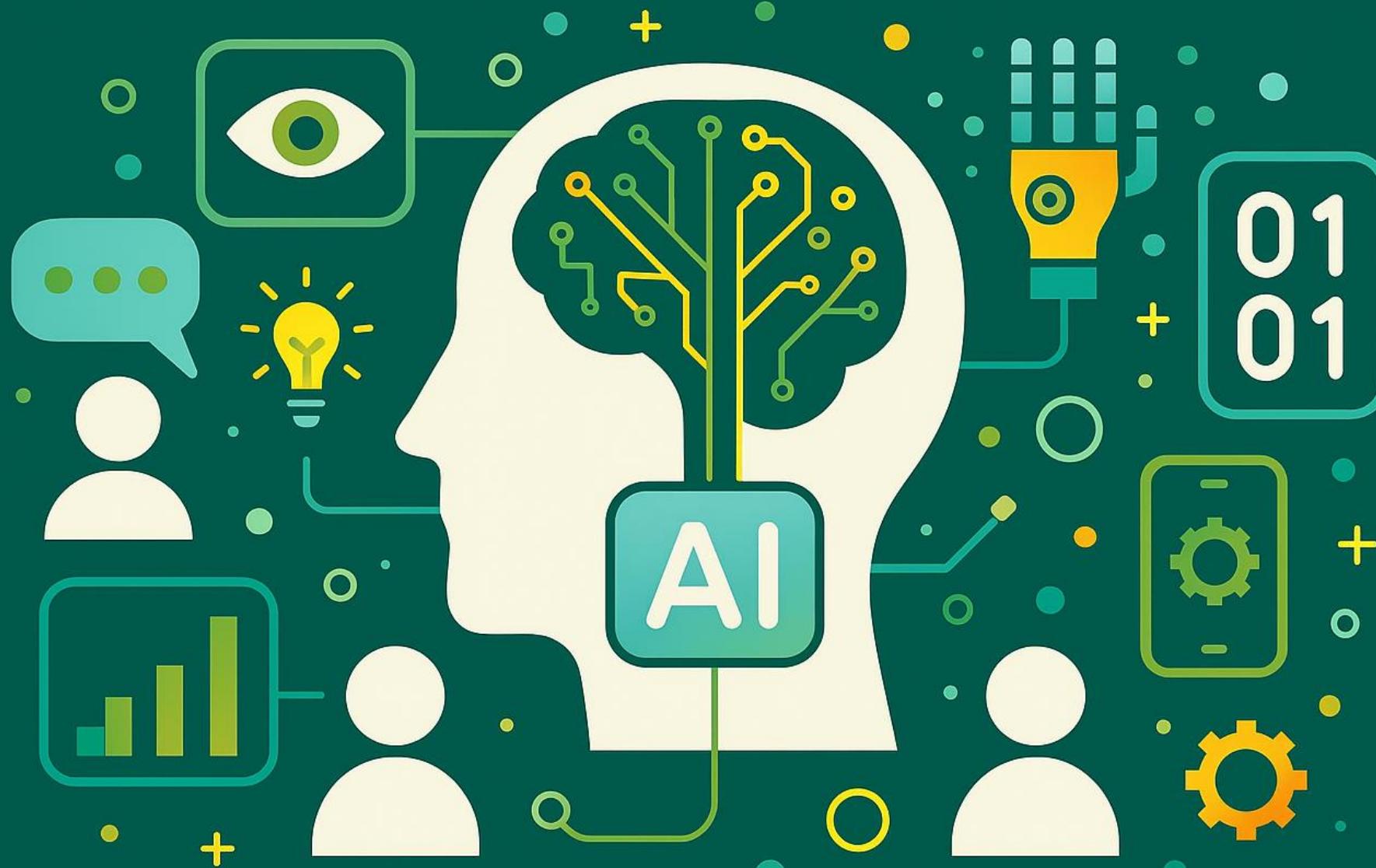


# Data preprocessing



## **Licence Disclaimer**

This document is distributed under the terms of the GNU General Public Licence, Version 3, dated 29 June 2007. It is intended to promote freedom to use, study, modify, and share the content herein, in accordance with the principles of free and open-source documentation. By accessing, reproducing, or modifying this document, you agree to comply with the conditions set forth in the GNU GPL v3. A full copy of the licence is available at <https://www.gnu.org/licenses/gpl-3.0.html>.

This licence applies to the document as a whole, including any derivative works, unless otherwise stated. No warranties are provided, and the document is offered “as-is” without liability for its use or interpretation.

## **Attribution Requirement**

When using, sharing, or adapting this document for any individual, group, or organisation, proper citation of the original author is required. Please cite as follows:

Alsaggaf, I. (2025) *Introduction to Artificial Intelligence*. Available at:

<https://github.com/ibrahimsaggaf/Introduction-to-Artificial-Intelligence> (Accessed: [insert date]).

# Content

- Data cleaning
- Data scaling
- Encoding
- Feature selection
- Q&A

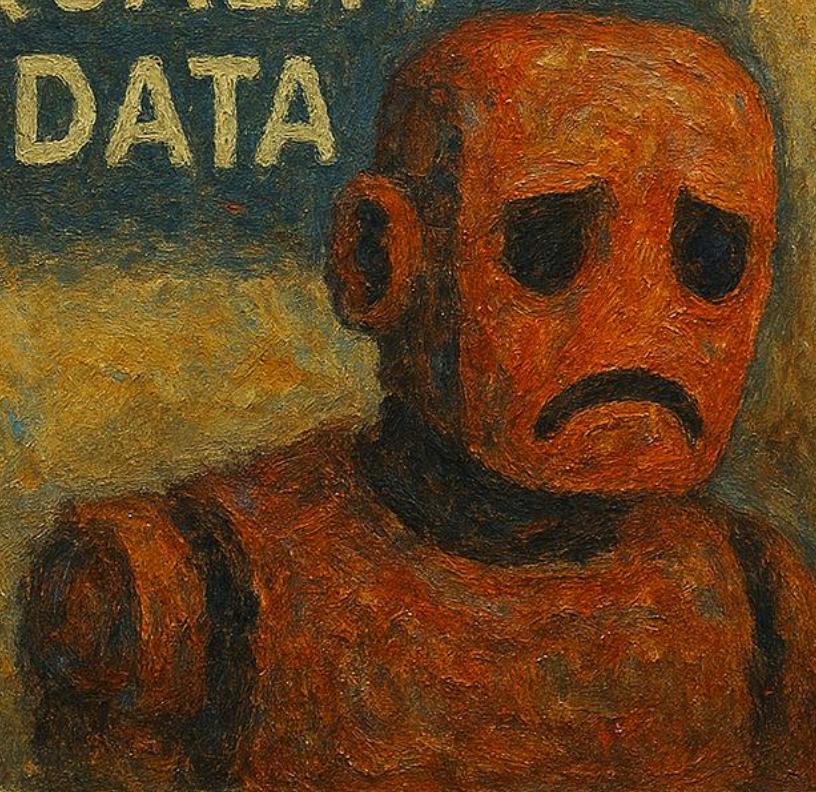
Lab session: Implementing a data preprocessing pipeline

# Data

1. How important is data in the success of AI projects?
2. To what extent does data quality influence the outcomes of AI initiatives?
3. What role does data play in determining the effectiveness of AI models?
4. Why does data matter so much in AI development?
5. Can AI succeed without high-quality data?

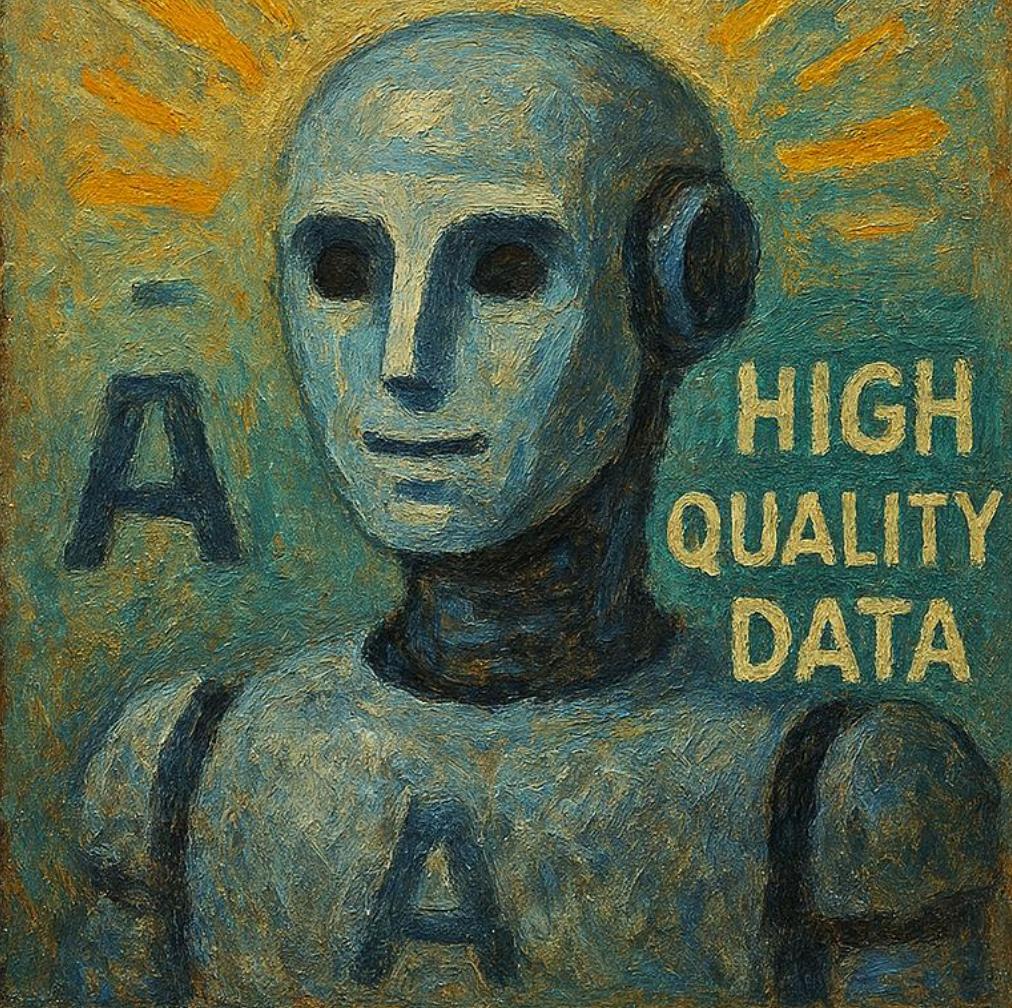
PAST

POOR  
QUALITY  
DATA ?



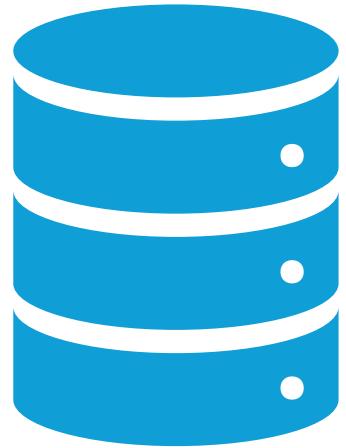
PRESENT

A  
HIGH  
QUALITY  
DATA



# Data cleaning

Data

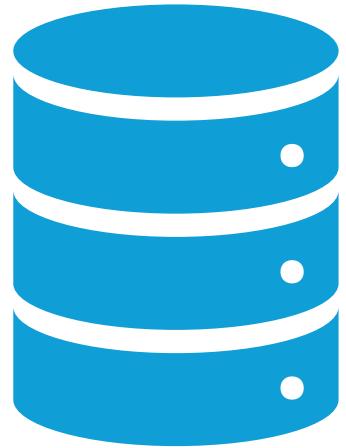


- Availability
- Correctness
- Completeness
- Consistency
- Freshness

✓ Gather data

# Data cleaning

Data

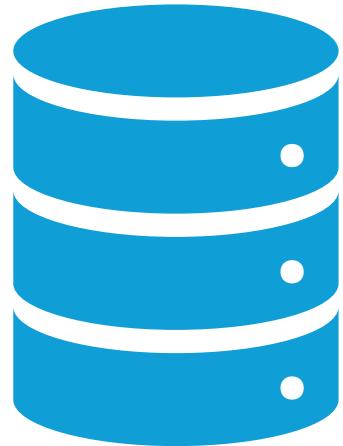


- Availability
- Correctness
- Completeness
- Consistency
- Freshness

- ✓ Fix errors
- ✓ Remove outliers

# Data cleaning

Data

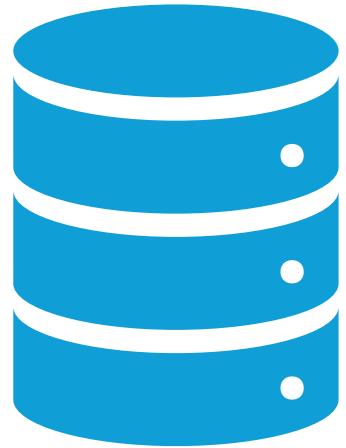


- Availability
- Correctness
- Completeness
- Consistency
- Freshness

✓ Impute nulls

# Data cleaning

Data

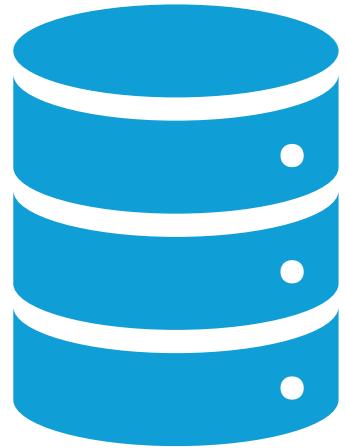


- Availability
- Correctness
- Completeness
- Consistency
- Freshness

✓ Enforce standards

# Data cleaning

Data

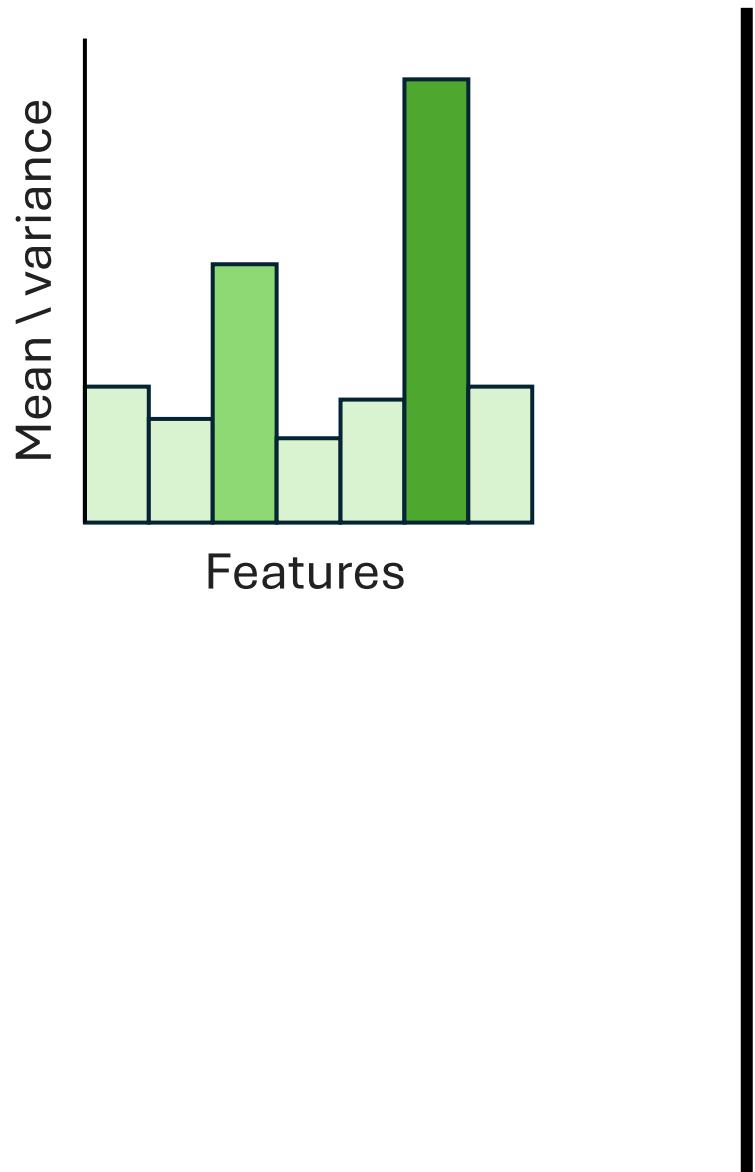


- Availability
- Correctness
- Completeness
- Consistency
- Freshness

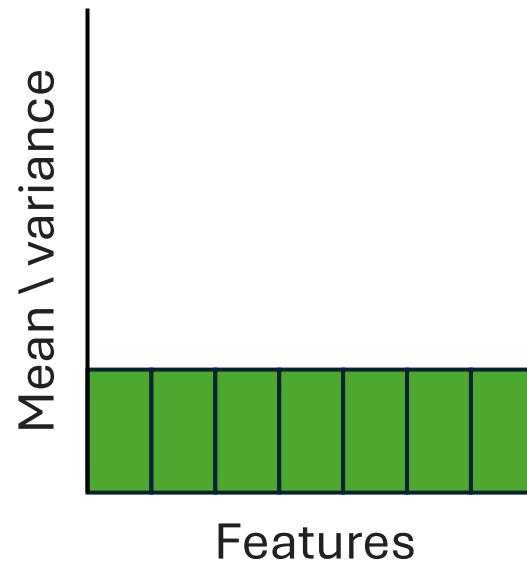
- ✓ Update frequency
- ✓ Streaming

# Data scaling

Unscaled data



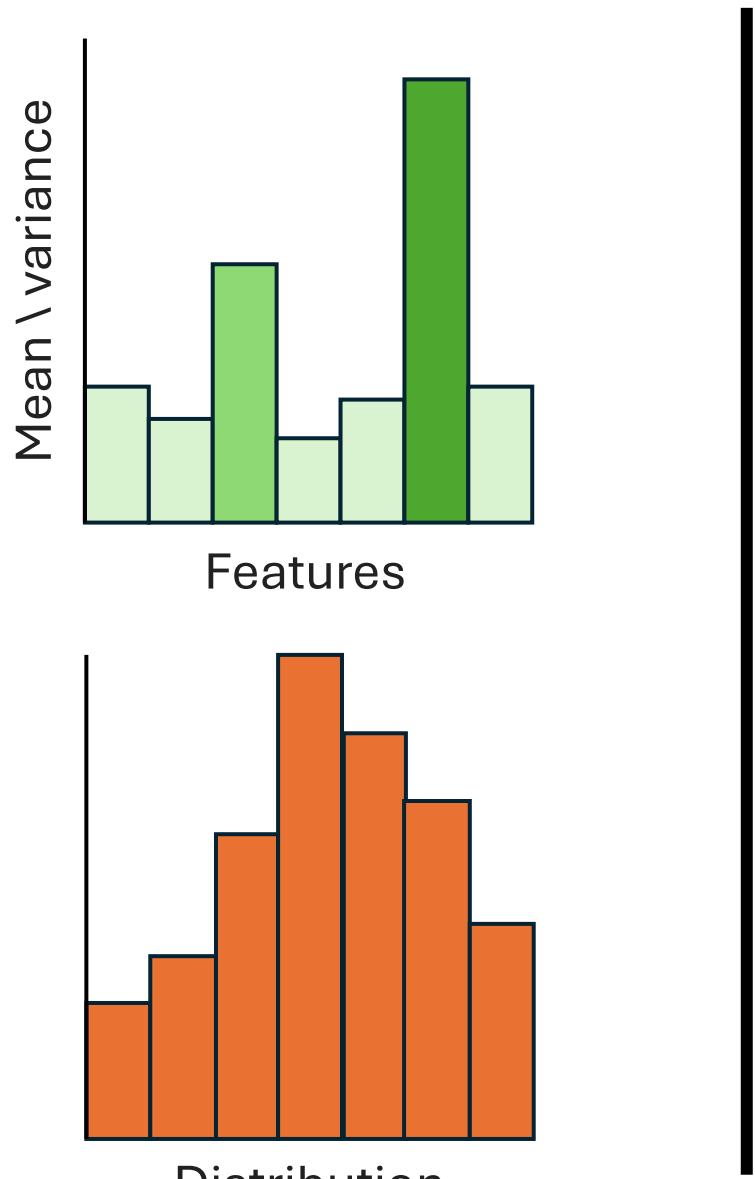
Scaled data



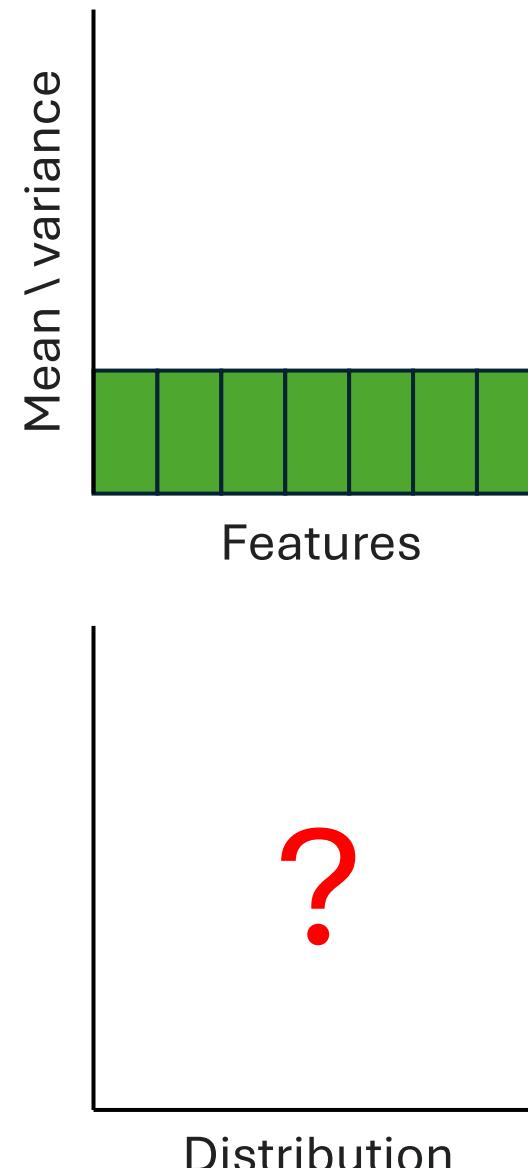
- ✓ Remove differences in statistical properties
- ✓ All features equally contribute to learning

# Data scaling

Unscaled data



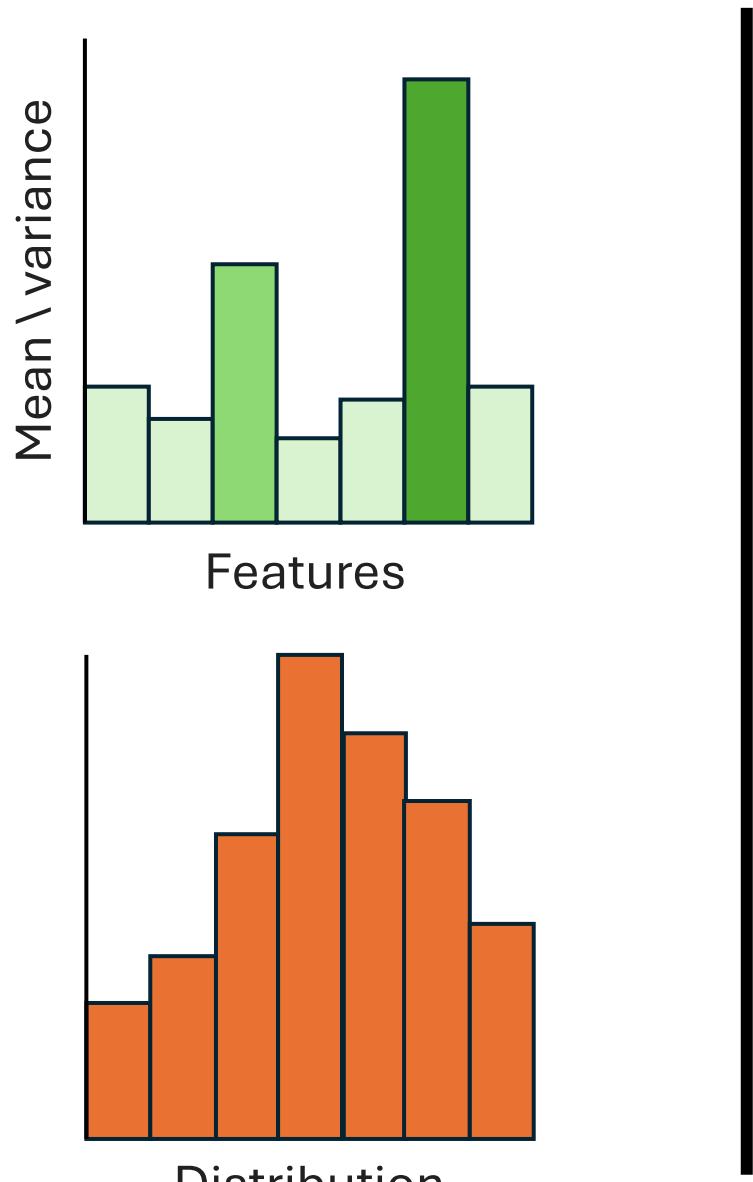
Scaled data



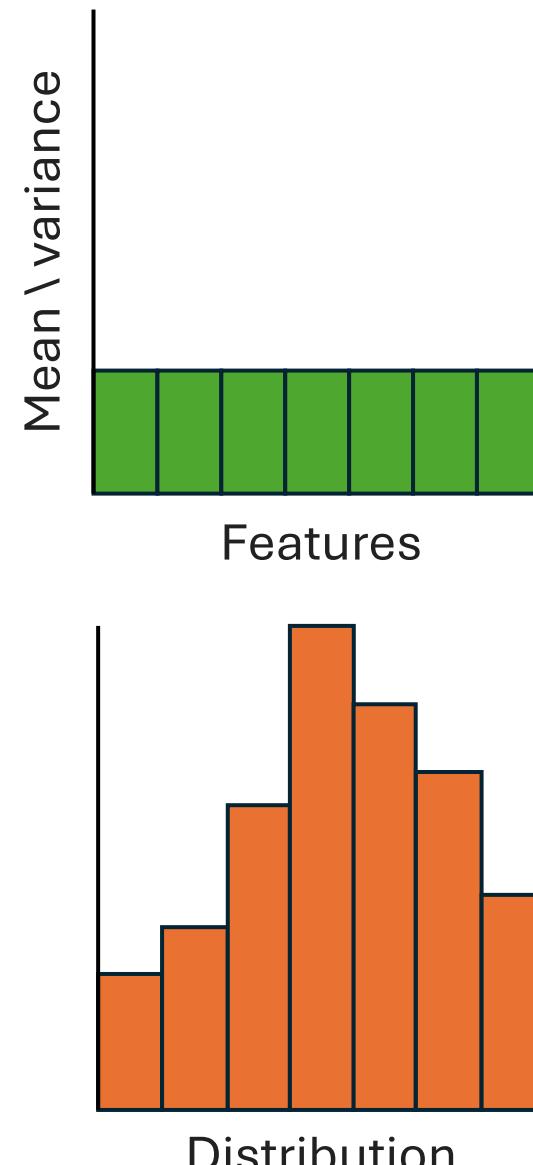
- ✓ Remove differences in statistical properties
- ✓ All features equally contribute to learning

# Data scaling

Unscaled data



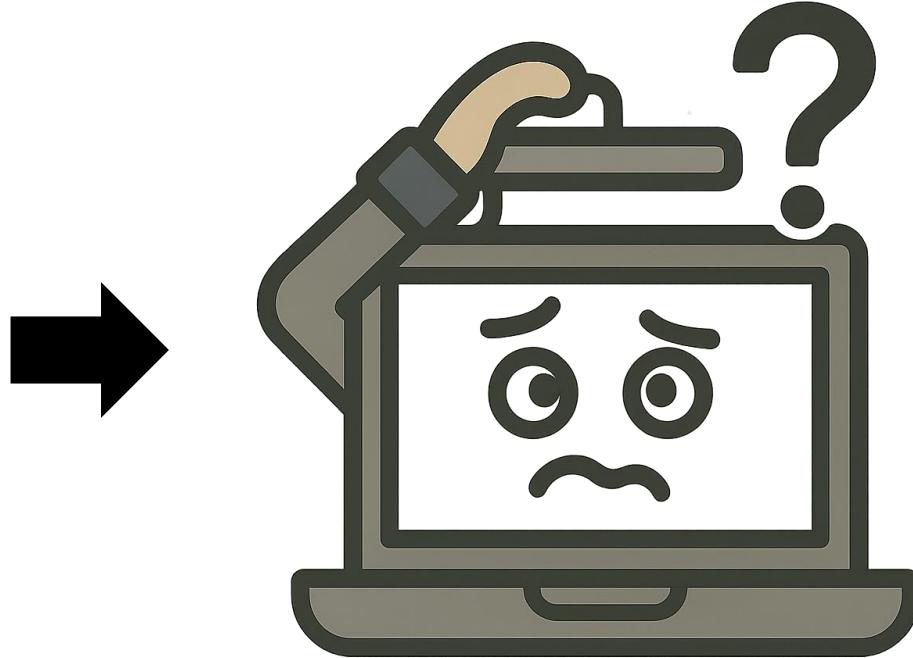
Scaled data



- ✓ Remove differences in statistical properties
- ✓ All features equally contribute to learning
- ✓ Maintain feature distribution

# Encoding

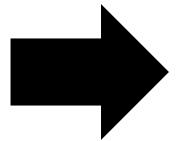
y
Green
Red
Blue
Orange
Blue
Red
Green
Green
Orange



Machines cannot  
understand natural language

# Encoding

y
Green
Red
Blue
Orange
Blue
Red
Green
Green
Orange

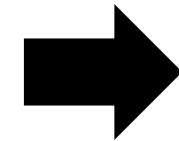


Label encoding

{

Blue: 0,  
Green: 1,  
Orange: 2,  
Red: 3

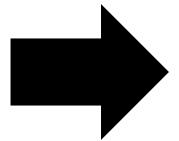
}



y
1
3
0
2
0
3
1
1
2

# Encoding

y
Green
Red
Blue
Orange
Blue
Red
Green
Green
Orange

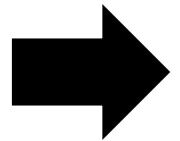


One-hot encoding

{

Blue: 0,  
Green: 1,  
Orange: 2,  
Red: 3

}



Blue	Green	Orange	Red	
y	0	1	0	0
0	0	0	1	
1	0	0	0	
0	0	1	0	
1	0	0	0	
0	0	0	1	
0	1	0	0	
0	1	0	0	
0	0	1	0	

# Are all features informative?

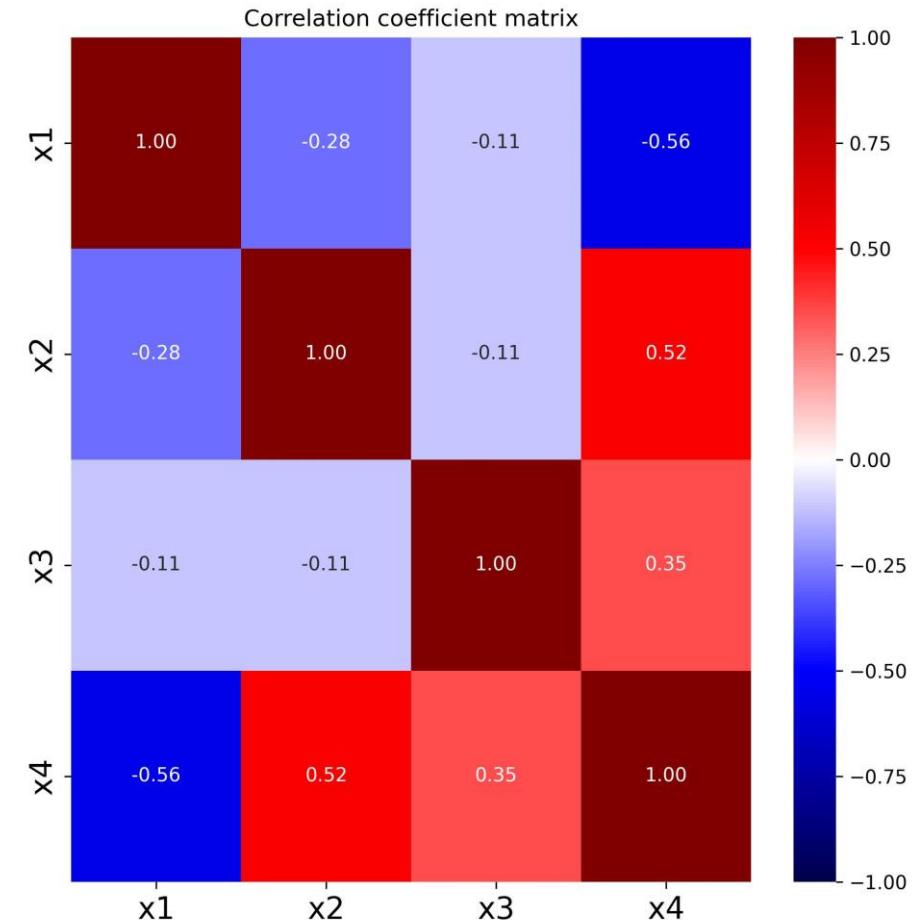
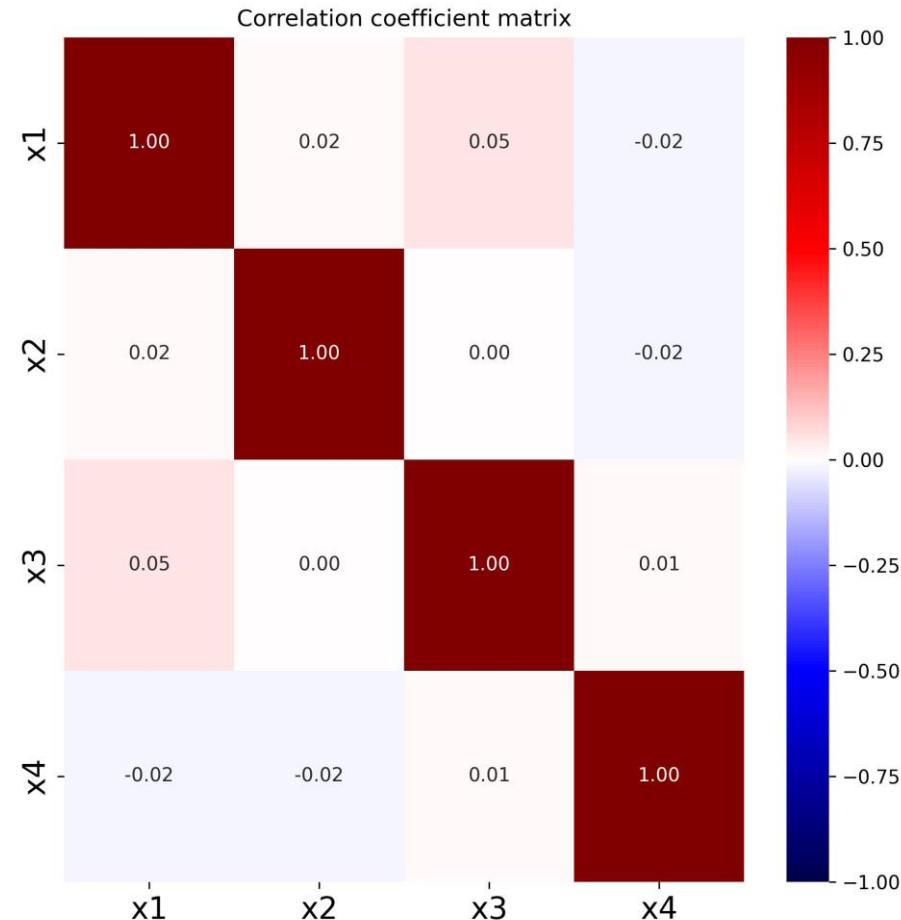
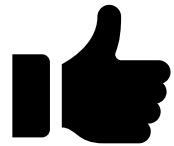
# Feature selection

## A sample of feature selection methods

Method	Type	Requires training	Expensive
Pearson's correlation coefficient	Statistical-based		
Analysis of Variance (ANOVA)	Statistical-based		
Recursive Feature Elimination (RFE)	Model-based	✓	✓
Tree-based models (e.g. Random Forest)	Model-based	✓	
...	...		

# Pearson's correlation coefficient

# Feature selection



# Tree-based model (Random Forest)

## Feature selection



Iris

Donated on 6/30/1988

A small classic dataset from Fisher, 1936. One of the earliest known datasets used for evaluating classification methods.

### Dataset Characteristics

Tabular

### Subject Area

Biology

### Associated Tasks

Classification

### Feature Type

Real

### # Instances

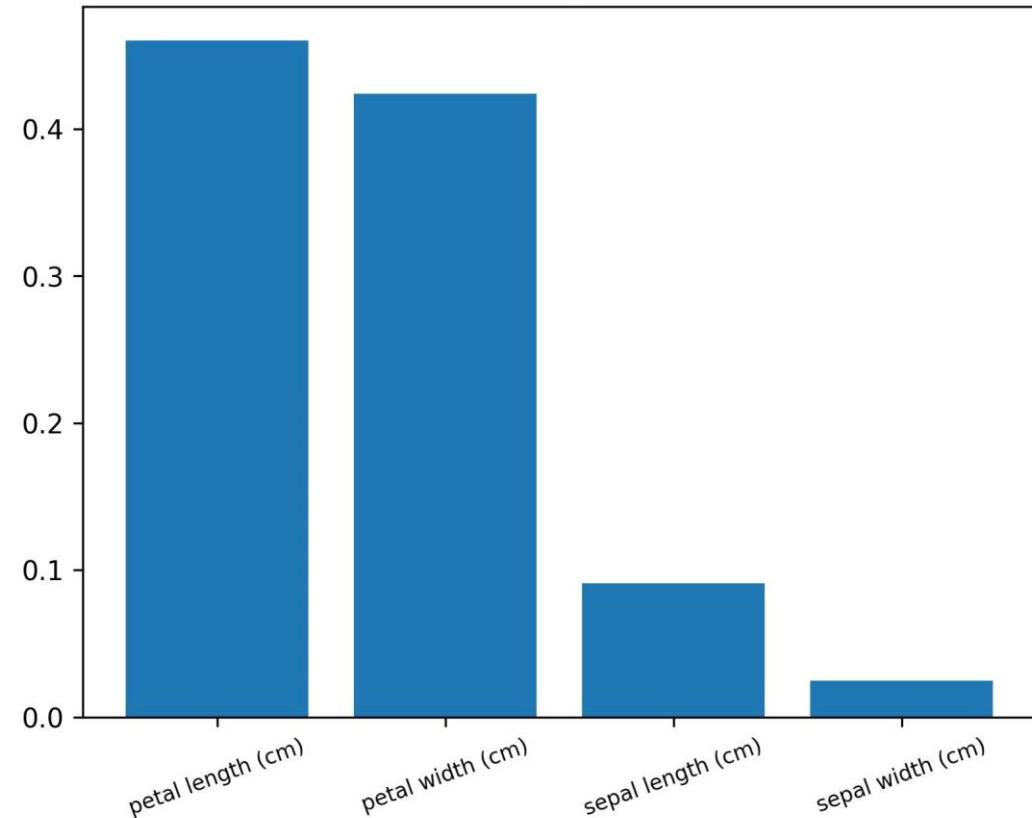
150

### # Features

4

[Source] UCI Machine Learning Repository

Feature Importance



# Q&A



# Lab Time

Lab 2: Implementing a data preprocessing pipeline

# From Lab1 we know how to:

[✓] Install Python 3.14.2

[✓] Install Visual Studio Code

[✓] Verify the installation

[✓] Create a virtual environment

# Lab 2: Dataset

## 8.1.2. Diabetes dataset

Ten baseline variables, age, sex, body mass index, average blood pressure, and six blood serum measurements were obtained for each of  $n = 442$  diabetes patients, as well as the response of interest, a quantitative measure of disease progression one year after baseline.

### Data Set Characteristics:

**Number of Instances:** 442

**Number of Attributes:** First 10 columns are numeric predictive values

**Target:** Column 11 is a quantitative measure of disease progression one year after baseline

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	target
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641	135.0

# Step 1

- Download the Lab2 directory from the GitHub repository  
<https://github.com/ibrahimsaggaf/Introduction-to-Artificial-Intelligence>
- Open the Lab2 directory in Visual Studio Code.
- The Lab2 directory contains 3 python files:
  - without\_pipeline.py
  - with\_pipeline.py
  - utils.py

Take your time examining the code in these files.

## Step 2

- Create and activate a virtual environment under the name “lab2\_env” (see Lab 1)
- Install the scikit-learn library inside the virtual environment by running the command:  
*pip install -U scikit-learn*

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File Edit Selection View Go Run ...
- Search Bar:** Untitled (Workspace)
- Sidebar (Left):**
  - EXPLORER
  - UNTITLED (WORKSPACE)
    - lab2
      - lab2\_env
    - utils.py
    - with\_pipeline.py (selected)
    - without\_pipeline.py
- Terminal:** The terminal window is active and displays the following command-line session:

```
PS C:\Users\sagga\Downloads\lab2> python -m venv lab2_env
PS C:\Users\sagga\Downloads\lab2> lab2_env\scripts\activate
(lab2_env) PS C:\Users\sagga\Downloads\lab2> pip install -U scikit-learn
Collecting scikit-learn
  Using cached scikit_learn-1.8.0-cp314-cp314-win_amd64.whl.metadata (11 kB)
Collecting numpy>=1.24.1 (from scikit-learn)
  Using cached numpy-2.4.1-cp314-cp314-win_amd64.whl.metadata (6.6 kB)
Collecting scipy>=1.10.0 (from scikit-learn)
  Using cached scipy-1.17.0-cp314-cp314-win_amd64.whl.metadata (60 kB)
Collecting joblib>=1.3.0 (from scikit-learn)
  Using cached joblib-1.5.3-py3-none-any.whl.metadata (5.5 kB)
Collecting threadpoolctl>=3.2.0 (from scikit-learn)
  Using cached threadpoolctl-3.6.0-py3-none-any.whl.metadata (13 kB)
Using cached scikit_learn-1.8.0-cp314-cp314-win_amd64.whl (8.1 MB)
Using cached joblib-1.5.3-py3-none-any.whl (309 kB)
Using cached numpy-2.4.1-cp314-cp314-win_amd64.whl (12.4 MB)
Using cached scipy-1.17.0-cp314-cp314-win_amd64.whl (37.1 MB)
Using cached threadpoolctl-3.6.0-py3-none-any.whl (18 kB)
Installing collected packages: threadpoolctl, numpy, joblib, scipy, scikit-learn
Successfully installed joblib-1.5.3 numpy-2.4.1 scikit-learn-1.8.0 scipy-1.17.0 threadpoolctl-3.6.0
(lab2_env) PS C:\Users\sagga\Downloads\lab2>
```
- Status Bar:** Shows icons for file status (0 changes), search (0 results), and timeline (0 changes).
- Bottom Right:** Signed out button.

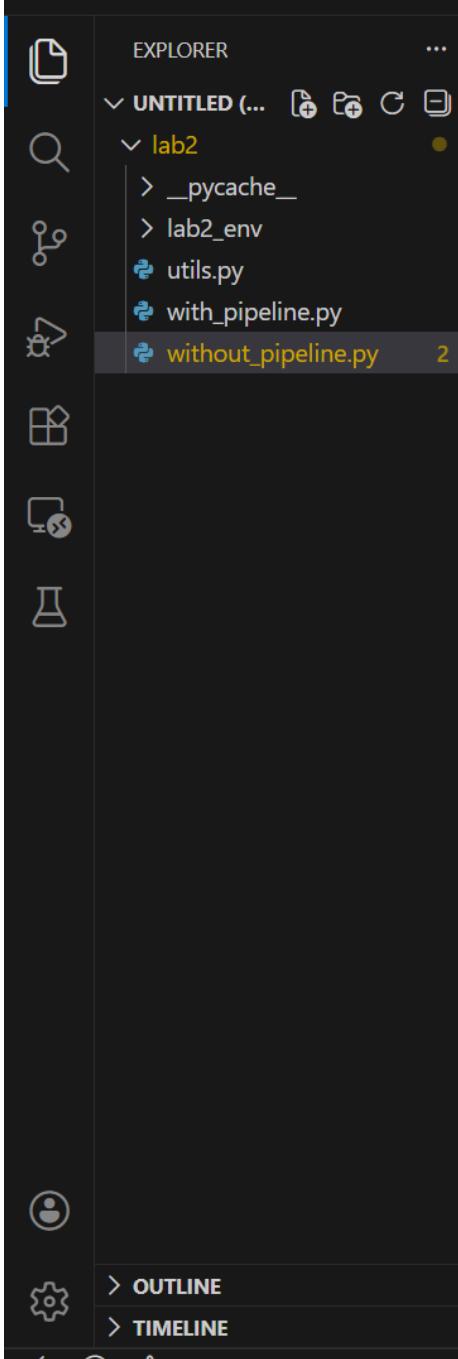
## Step 3

- Run the command:

*python without\_pipeline.py*

This command evaluates a model's performance without applying data preprocessing:

1. Split the data into training and testing sets.
2. Train a Support Vector Machine (SVM) on the training set to solve a regression task – predicting disease progression one year after baseline.
3. Measure the model's performance on the testing set using Mean Squared Error (MSE).



without pipeline.py 2 >

lab2 > 🐍 without\_pipeline.py > .

```
1 ...
2 Author: Dr Ibrahim Alsaggaf
3 Learning type: Supervised learning
4 Task: Regression without data preprocessing
5 Dataset: Diabetes dataset [1]
6 Library: Scikit-learn [2]
7 Model: Support Vector Machine (SVM)
8
9 [1] Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression.
10 |   https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html
11
12 [2] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
13 ...
14
15
16 from sklearn.svm import SVR
17 from sklearn.datasets import load_diabetes
18
19 from utils import split, performance
20
21 # Load the dataset from the Scikit-learn library
22 dataset = load_diabetes()
23 X, y = dataset.data, dataset.target
```

PROBLEMS 2 OUTPUT TERMINAL PORTS DEBUG CONSOLE

P

OUTP

TERMINAL

PORTS

DEBUG CONSOLE

powershell + ⌂ ⌂ ... | ⌂ ×

```
(lab2_env) PS C:\Users\sagga\Downloads\lab2> python without_pipeline.py  
MSE: 5706.1262  
(lab2_env) PS C:\Users\sagga\Downloads\lab2> █
```

## Step 4

- Run the command:  
*python with\_pipeline.py*

This command evaluates a model's performance with applying data preprocessing :

1. Scale the features.
2. Select the top 2 important features using a tree-based model.
3. Split the data into training and testing sets.
4. Train a Support Vector Machine (SVM) on the training set to solve a regression task – predicting disease progression one year after baseline.
5. Measure the model's performance on the testing set using Mean Squared Error (MSE).

The screenshot shows a dark-themed code editor interface, likely Visual Studio Code (VS Code). The top bar includes standard menu items: File, Edit, Selection, View, Go, Run, etc. The title bar displays "Untitled (Workspace)". The left sidebar has icons for Explorer, Search, Open, Find, and others. The Explorer view shows a workspace folder named "UNTITLED (WORKSPACE)" containing a "lab2" folder with files: \_\_pycache\_\_, lab2\_env, utils.py, with\_pipeline.py (selected), and without\_pipeline.py. The main editor area displays the content of "with\_pipeline.py". The code is a Python script for regression analysis using Scikit-learn:

```
1  ''
2  Author: Dr Ibrahim Alsaggaf
3  Learning type: Supervised learning
4  Task: Regression with data preprocessing
5  Dataset: Diabetes dataset [1]
6  Library: Scikit-learn [2]
7  Model: Support Vector Machine (SVM)
8
9  [1] Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression.
10 |   https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html
11
12 [2] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
13 ''
14
15
16 from sklearn.svm import SVR
17 from sklearn.datasets import load_diabetes
18
19 from utils import split, scale, feature_selection, performance
20
21 # Load the dataset from the Scikit-learn library
22 dataset = load_diabetes()
23 X, y = dataset.data, dataset.target
```

The bottom navigation bar includes PROBLEMS (2), OUTPUT, TERMINAL (selected), PORTS, and DEBUG CONSOLE. The terminal window shows the execution of the script in a PowerShell environment:

```
(lab2_env) PS C:\Users\sagga\Downloads\lab2> python with_pipeline.py
The number of features before selection is 10
The number of features after selection is 2
MSE: 3089.1350
(lab2_env) PS C:\Users\sagga\Downloads\lab2>
```

The status bar at the bottom indicates: Ln 28, Col 63 (60 selected) Spaces: 4 UTF-8 CRLF { } Python. The bottom right corner shows the version 3.14.2 and a signed-out user icon.

# Lab 2: Data preprocessing pipeline

Congrats! 

[] Install the scikit-learn library

[] Implement a data preprocessing pipeline

# Quiz 1

Q1: Which statement correctly defines outliers?

- A) Outliers are data errors resulting from invalid inputs or irrelevant observations
- B) Outliers are genuine data points that represent extreme or unusual cases and may offer valuable insights
- C) Either A or B
- D) Both A and B

Q2: In this lab, what conclusion can be drawn about the impact of applying data preprocessing?

- A) Improved performance but increased runtime
- B) Improved performance and decreased runtime
- C) Worse performance but decreased runtime
- D) Worse performance and increased runtime

# Reading list

What is Data Imputation

<https://www.geeksforgeeks.org/machine-learning/what-is-data-imputation>

Recursive Feature Elimination (RFE) for Feature Selection in Python

<https://machinelearningmastery.com/rfe-feature-selection-in-python>