

# TP 4 Lab: NumPy Data Analysis — Load a Numeric CSV, Describe, Standardize, Detect Outliers

---

## Goal

Using NumPy (no pandas), you will:

- load a numeric dataset from CSV (including a “dirty” version),
  - compute column-wise statistics (NaN-aware),
  - standardize features with z-scores,
  - detect outliers,
  - export a normalized CSV.
- 

## Constraints

- Use a **virtual environment (venv)**.
  - Use **NumPy** for computations (no pandas).
  - Standard library allowed for I/O (**csv**).
  - For computations (stats/standardization/outliers), avoid Python loops when possible (use NumPy ops).
  - Missing values must be represented as **np.nan**.
- 

## Required Project Structure

```
session4/
  main.py
  requirements.txt
  README.md
  src/
    np_io.py
    np_stats.py
  data/
    measures_clean.csv
    measures_dirty.csv
  out/
```

---

## Data Files (copy/paste)

### **data/measures\_clean.csv**

```
id,age,income,spend,score
1,23,2100,1200,0.62
2,45,4200,2300,0.71
3,31,3200,1800,0.55
4,29,2900,1600,0.60
5,52,5100,2600,0.78
6,41,3900,2100,0.66
7,36,3600,1950,0.58
8,27,2500,1400,0.63
```

### **data/measures\_dirty.csv**

```
id,age,income,spend,score
1,23,2100,1200,0.62
2,45,4200,2300,0.71
3,31,,1800,0.55
4, 29 ,2900,1600,0.60
5,52,5100,2600,0.78
6,41,3900,2100,NA
7,36,3600,1950,0.58
8,27,2500,1400,0.63
```

`9,not_an_int,9999,9999,0.99`

---

## Tasks

### Task A — Robust CSV Loader to NumPy (25 min)

In `src/np_io.py`, implement:

- `load_numeric_csv(path: str) -> tuple[np.ndarray, list[str]]`

### Requirements

- Read the CSV with `csv.DictReader`.
- Return:
  - `X`: a float NumPy array of shape (`n_rows, n_features`) using columns `[age, income, spend, score]`
  - `cols`: the list `["age", "income", "spend", "score"]`
- Convert values:
  - `" "` or `"NA"` → `np.nan`
  - extra spaces should be handled (`" 29 "` → `29`)
- If a row contains a non-numeric value that cannot be converted (e.g., `not_an_int`), **skip the row** and print a warning:
  - `WARNING line X skipped: ...`

---

### Task B — Column-wise Describe (NaN-aware) (25 min)

In `src/np_stats.py`, implement:

- `describe(X: np.ndarray, cols: list[str]) -> dict`

For each column, compute:

- `mean (np.nanmean)`
- `std (np.nanstd)`
- `min (np.nanmin)`
- `max (np.nanmax)`
- number of missing values (`np.isnan(...).sum()`)

Return a dictionary like:

```
{  
    "age": {"mean": ..., "std": ..., "min": ..., "max": ..., "n_nan": ...},  
    ...  
}
```

---

### Task C — Z-score Standardization (30 min)

In `src/np_stats.py`, implement:

- `zscore(X: np.ndarray) -> np.ndarray`

Rules:

- Standardize each column:  $(X - \text{mean}) / \text{std}$
  - Use NaN-aware mean/std (`nanmean, nanstd`)
  - Avoid division by zero: if `std == 0`, use `1.0` instead
-

### Task D — Outlier Detection (15 min)

In `src/np_stats.py`, implement:

- `outlier_mask(Z: np.ndarray, threshold: float = 3.0) -> np.ndarray`

Definition:

- outlier if `abs(z) > threshold`

You should print, for each column:

- number of outliers detected
- 

### Task E — Export Normalized Data (15 min)

In `src/np_stats.py`, implement:

- `export_csv(path: str, cols: list[str], Z: np.ndarray) -> None`

Rules:

- Write a CSV with header `age, income, spend, score`
- For values:
  - finite numbers → write with 6 decimals
  - `Nan` or infinite → write as empty string

Export path:

- `out/normalized.csv`
- 

### Task F — Main Program (CLI) (10 min)

In `main.py`:

- parse `--input` (required)
- run: load → describe → zscore → outlier\_mask → export
- print:
  - basic stats per column
  - outlier counts per column
  - export confirmation

Example runs:

```
python main.py --input data/measures_clean.csv  
python main.py --input data/measures_dirty.csv
```

---

## Deliverables

- Working `session4/` project with the required structure
- `README.md` filled (template below)
- Exported file: `out/normalized.csv`