EE456 Computer Assignment 1

Due: 2025-Sep-30 at 23:59

Overview

In this assignment, you will implement functions in a provided Python framework to process the UCI Iris dataset. This assignment emphasizes correct handling of data pipelines: loading, reporting, splitting, basic statistics, and visualization.

Learning Objectives

- Practice writing and organizing functions inside Python classes.
- Load data from the UCI repository.
- Generate class balance reports and numeric statistics.
- Perform reproducible train/validation splits.
- Create common plots with `matplotlib`.

Dataset

You must load the Iris dataset directly from the UCI repository: `https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data`

Columns (no header provided in file):

```
["sepal_length", "sepal_width", "petal_length", "petal_width", "species"]
```

Constraints (Important)

- Do not download or use a local copy; load directly from the URL in your code.
- Allowed libraries: `pandas`, `numpy`, `matplotlib`, `json`, `os`, `time`.
- Call `ensure_dir("./outputs")` exactly once in `main()`.
- All log messages must include your **initials** and your **9-digit student ID**.

Required Logging Format

All operations must log using the provided helper function:

```
[HH:MM:SS] [XX-123456789] Your message
```

where `XX` are your initials and the number is your PSU 9-digit ID.

Listing 1: Required log format

```python
def log(msg: str, initials: str, student_id: str) -> None:
    import time
    print(f"[{time.strftime('%H:%M:%S')}] [{initials}-{student_id}] {msg}")
```

Tasks

You will complete all methods marked with `TODO` in the framework. The work is divided into two classes:

A. Class `IrisLoaderUCI`

A.1 `load()`: Read the CSV from UCI, drop fully empty rows, reset index.

A.2 `check_class_balance()`: Report counts and proportions of each species.

A.3 `save_head(k)`: Shuffle data using the last two digits of your student ID as the seed, then save the first k rows.

A.4 `save_class_balance()`: Save the balance report as `class_balance.json`.

B. Class `Processor`

B.1 `add_numeric_label()`: Map species names (alphabetical) to integers and save `label_map.json`.

B.2 `stats()`: For each numeric column, compute `min, max, mean, median, std`.

B.3 `train_val_split()`: Split *in order* (no shuffle). Validation size = round($n\times$ ratio), clamped so both sets have ≥ 1 row.

B.4 **Plots:**

- Histogram of one numeric feature.
- Bar chart of species distribution.
- Scatter plot of two numeric features colored by species.

B.5 `save_processed()`: Save the final processed DataFrame.

Outputs (must exist in `./outputs/`)

- `class_balance.json`, `head.csv`, `label_map.json`
- `train.csv`, `val.csv`, `processed.csv`
- `stats.json`
- Plots: `hist_petal_length.png`, `label_bar.png`, `scatter_petal.png`

What to Submit

Submit a single `hw1_submission.zip` archive that contains:

- **Source code:** your completed `hw1_iris.py`.
- **Outputs folder:** the entire `./outputs/` directory with all generated CSV/JSON/PNG files.
- **Console screenshot:** showing program run with correct logging format and messages.

Rubric (100 pts)

- Logging(10)
- IrisLoaderUCI (40): load, balance, head, save
- Processor (40): labels, stats, split, plots, save
- Reproducibility & determinism (10)