Esercizio

Compendio: Framework di Analisi Dati Personalizzato

Obiettivo generale

L'esercizio chiede di costruire una mini-libreria in Python per gestire dataset rappresentati come **liste di dizionari**. L'idea è simulare una piccola versione semplificata di strumenti come *pandas*, ma limitandoci a concetti di **programmazione a oggetti**, **funzioni higher-order** e **metodi speciali**.

Il lavoro è suddiviso in **milestone (M1–M8)**: ogni milestone introduce un concetto chiave di OOP o programmazione funzionale, consolidando man mano le competenze acquisite.

Struttura dell'esercizio

La classe centrale è `DataSet`, che rappresenta una collezione di record (dizionari con chiave→valore). Su questa classe si costruiscono:

- **Metodi magici** per trattarla come una collezione Python (len, iterazione, indicizzazione).
- **Metodi di trasformazione** in stile funzionale (filter, map, reduce).
- **Metodi di analisi** per raggruppare e calcolare statistiche.
- **Mixin di esportazione** per salvare i dati in JSON o CSV.

Alla fine, la classe `ExportableDataSet` eredita sia da `DataSet` che dai due mixin, diventando la versione completa e usabile nella pratica.

Le milestone spiegate

M1 – Validazione dei dati

- Implementazione del costruttore `__init__`.
- Lo studente deve garantire che i dati siano una lista di dizionari.
- In caso contrario, sollevare `ValueError`.

- Importante: memorizzare una **copia** dei dati per evitare modifiche esterne.

Concetti chiave: incapsulamento, validazione input.

M2 – Metodi magici

- Implementare `__len__`, `__iter__`, `__getitem__`.
- In questo modo `DataSet` si comporta come una sequenza Python:
  - `len(ds)` restituisce il numero di record.
  - `for r in ds` permette l'iterazione.
  - `ds[0]` accede a un record specifico.

Concetti chiave: metodi speciali, integrazione con le API built-in di Python.

M3 – Filter

- Metodo `filter(predicate)` che restituisce un nuovo `DataSet` contenente solo i record che soddisfano la condizione.
- Deve mantenere immutabile l'oggetto originale.

Concetti chiave: programmazione funzionale, immutabilità, higher-order functions.

M4 – Map

- Metodo `map(transform)` che applica una trasformazione a ogni record.
- Deve validare che il risultato di `transform` sia ancora un dizionario, altrimenti lanciare `TypeError`.

Concetti chiave: trasformazioni funzionali, robustezza tramite validazioni.

M5 – Reduce

- Metodo `reduce(func, initial)` che riduce il dataset a un unico valore (ad esempio somma, concatenazione, conteggio).
- Utilizza lo schema `acc = func(acc, record)` partendo da `initial`.

Concetti chiave: aggregazione, uso di `functools.reduce`.

M6 – Group By

- Metodo `group_by(key_or_fn)` che raggruppa i record:
  - Se si passa una stringa, viene usata come chiave di campo.
  - Se si passa una funzione, viene usato il risultato della funzione come chiave.
- Deve restituire un dizionario che mappa ogni gruppo a un nuovo `DataSet`.

Concetti chiave: polimorfismo (stringa o funzione), creazione di sotto-collezioni.

M7 – Statistiche numeriche

- Implementare metodi: `sum(field)`, `mean(field)`, `min(field)`, `max(field)`.
- Devono estrarre i valori numerici di un campo e calcolare le statistiche.
- Se un campo non esiste → `ValueError`.
- Se un valore non è numerico → `TypeError`.
- Se il dataset è vuoto → `ValueError`.

Concetti chiave: gestione errori, data validation, funzioni di aggregazione.

M8 – Mixin di esportazione

- Creare due mixin:
  - `ExportJSONMixin` → metodo `export_json(path)` che salva i record in formato JSON.
  - `ExportCSVMixin` → metodo `export_csv(path)` che salva i record in formato CSV, con colonne ordinate alfabeticamente.
- Infine, creare `ExportableDataSet` che eredita da `DataSet`, `ExportJSONMixin` e `ExportCSVMixin`.

Concetti chiave: ereditarietà multipla, riuso tramite mixin, I/O su file.