Euromed Analytics Dashboard

Technical Report

Aya Hassan , Yassir Tagmouati

June 2025

Contents

1 Introduction and Project Overview

The **Euromed Analytics** project is a web-based application for analyzing and visualizing student data at the University Euromed of Fès. It provides statistical and predictive tools to help university administrators and decision-makers understand the student population, academic performance, and other key indicators.

The main goal was to build an interactive, high-performance, and intuitive platform capable of handling large datasets while providing advanced statistical insights and predictions.

2 System Architecture

2.1 Technical Architecture

- **Backend:** REST API built with Python (Flask)
- **Frontend:** Single Page Application (React + Material-UI)
- **Communication:** JSON over HTTP
- **Persistence:** CSV files for data storage

2.2 Code Structure

```
LO2/
 backend/
    app.py
    services/
    data/
    guaranteed_start.py
    utils/
 frontend/
    src/
       components/
       views/
       utils/
       config.js
    public/
 run.sh
```

3 Backend Development

3.1 REST API Design

Key endpoints implemented include:

- `/api/data/summary` - Data summary
- `/api/statistics/...` - Gender, nationality, city stats
- `/api/predictions/...` - Predictions (graduation, income, major)

- `/api/upload` - CSV data import
- `/api/schema` - Schema information

3.2 Data Management

Optimized CSV Parsing:

- Custom parser in `csv_parser.py`
- Chunk-based memory management
- Automatic type detection

Schema Analysis:

- `schema_analyzer.py` to detect column structure
- Dynamic adaptation of features to input columns

3.3 Large Dataset Optimizations

- **Chunk processing:** via `data_chunker.py`
- **Memory release:** periodic GC cleanup
- **Caching:** LRU with TTL (`optimize_server.py`)
- **Sampling:** Smart sampling and approximation for large files

4 Frontend Development

4.1 User Interface

Developed using React + Material-UI for a modern, responsive experience.

Main Sections:

- Dashboard: KPIs and overview
- Statistics: Distribution charts by gender, city, etc.
- Predictions: Academic success, major, financial
- Data Import: CSV upload with validation

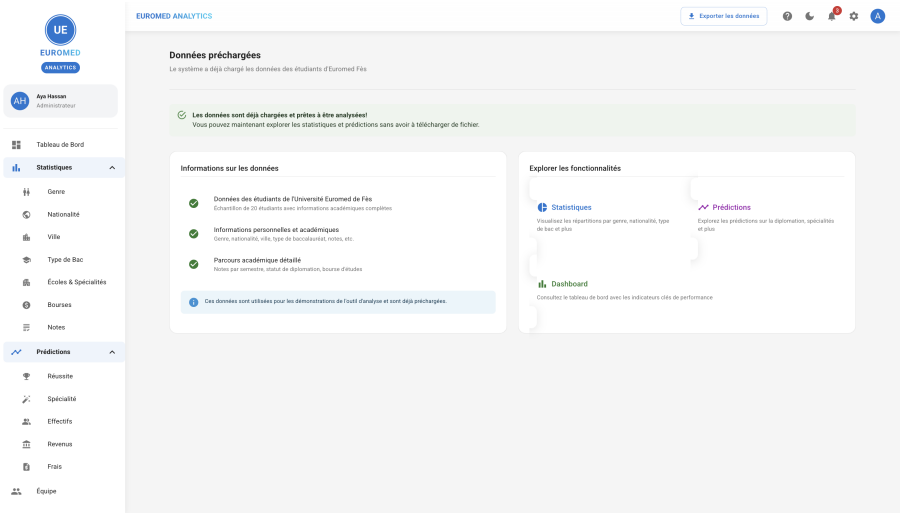

Figure 1: Dashboard overview screen

4.2 Data Visualizations

- **Pie Charts:** Custom `PieChart` component with fallback for missing data
- **Advanced Tables:** `DataTablePaginated` with server-side pagination and filtering
- **Schema-Aware UI:** Smart feature disabling via `SchemaChecker`

4.3 Issues and Fixes

Chart display bug: "No data available" resolved by:

- Creating `chartDataHelper.js` to normalize data format
- Improving compatibility and debug logging in PieChart

5 Performance Optimization

5.1 Backend

- Async processing via workers
- Memory management and GC control
- Paginated and filtered APIs
- `guaranteed_start.py` for fallback execution handling

5.2 Frontend

- Progressive loading (skeletons)
- Virtualized long lists
- Lazy-loaded sections
- Performance logging and diagnostics

6 Deployment and Installation Tools

- `run.sh`: One-click launcher
- `adapt_my_data.py`: Converts external datasets
- `demo_100k.sh`: Simulates a dataset with 100K students

7 Testing and Validation

- Functional testing for all views
- Performance benchmarks on datasets from 10 to 100K rows
- User testing for clarity and responsiveness

8 Conclusion and Future Work

8.1 Key Achievements

- Modular, scalable architecture
- Informative and interactive dashboards
- Large-scale data support
- Adaptable to varied input formats

8.2 Next Steps

- Integrate more advanced AI models
- Add comparative analytics between cohorts
- Export APIs for third-party systems
- Customizable user dashboards

This project lays the foundation for a robust educational data analytics system and can be expanded to meet further institutional needs.