

Continuous Assessment (CA1): Project

Author: Tomasz Biel

Student no. 25113186

Module: Programming for AI, National College of Ireland

Date: 16/11/2025

1. Introduction

This report documents the full process of generating, cleaning, analyzing, and visualizing a synthetic student performance dataset. The goal was to create a structured, modular Python project that can serve both as a demonstration of data analysis workflow and as a reusable toolkit for similar datasets. The project encompasses four main stages: data generation, data cleaning, descriptive and extended analysis, and visualization.

The dataset consists of 500 synthetic student records, including attributes such as `study_hours`, `quiz_participation`, `past_performance`, `gender`, `age`, and `course_completion`. While the data are synthetic, they were designed to resemble realistic student patterns, including variability and some anomalies, allowing meaningful demonstrations of analytical techniques.

2. Project Structure

The project is organized to support modularity, readability, and future expansion:

```
NCI_programming_ca1/
├── data/                # Raw and cleaned CSV files
│   └── students.csv
├── docs/                # Report
├── reports/             # Generated reports and figures
│   └── figures/
├── src/                 # Source code
│   ├── analysis.py      # Data analysis module
│   ├── data_cleaning.py # Data cleaning utilities
│   ├── data_diagnosis.py # Data quality assessment
│   ├── data_generator.py # Synthetic data generation
│   └── visualization.py # Data visualization tools
├── tests/               # Unit tests
├── .gitignore
├── README.md
└── requirements.txt
```

The code is designed around **classes** for encapsulation (`DataAnalyzer`, `StudentGenerator`) and **functions** for modularity. Diagnostic, cleaning, analysis, and visualization modules are separated to maintain clarity and extensibility.

3. Data Generation

The synthetic dataset was created using the `StudentGenerator` class. Key design considerations included:

- Realistic variability in `study_hours` and `quiz_participation`.
- Generation of `past_performance` scores as a percentage.
- Introducing missing values (NaN) and inconsistent formatting (e.g., some quiz scores expressed as decimals or percentages).
- Inclusion of demographic data: age, gender.
- Creating `course_completion` flags (True/False).

This step was crucial because designing synthetic data that mirrors real-world variability requires creativity and careful parameter tuning. Unlike analyzing real data, the challenge here was to generate values that support meaningful statistical analysis.

4. Data Diagnosis

Before cleaning, the dataset was assessed using `data_diagnosis.py`. The analysis included:

- Checking unique values per column.
- Identifying missing or anomalous values.
- Detecting inconsistencies (e.g., quiz participation recorded as 0.75 or 75).

Results guided the cleaning strategy, including handling missing values, standardizing formats, and ensuring numerical columns were suitable for analysis.

5. Data Cleaning and Feature Engineering

Implemented in `data_cleaning.py`:

- **Missing value handling:**
 - `gender` → "Unknown"
 - Other NaNs were either imputed or flagged for attention.
- **Normalization:**
 - `study_hours` was normalized to [0,1] using min-max scaling.
- **Derived features:**
 - $\text{engagement} = 0.6 * \text{study_hours_norm} + 0.4 * \text{quiz_participation} / 100$
- **Bucketing categorical variables:**
 - Age grouped into `age_bucket`: 19–24, 25–34, 35–45, 46+.
- **Performance classification:**
 - `past_performance` categorized into High (≥ 85), Medium (60–84), Low (< 60).

This step ensured that the dataset was clean, consistent, and ready for both descriptive and extended analysis. The modular design allows applying the same cleaning functions to new datasets.

6. Descriptive and Extended Analysis

Analysis was conducted in `analysis.py` using the `DataAnalyzer` class.

6.1 Mandatory Analysis

1. Summary Statistics

- Mean, median, standard deviation calculated for `study_hours` and `quiz_participation`.
- | | |
|------------------------------------|-------|
| 2. <code>study_hours_mean</code> | 9.97 |
| 3. <code>study_hours_median</code> | 9.83 |
| 4. <code>study_hours_std</code> | 4.71 |
| 5. <code>quiz_part_mean</code> | 74.77 |
| 6. <code>quiz_part_median</code> | 75.50 |
| 7. <code>quiz_part_std</code> | 17.16 |

8. Correlation Matrix

- Explored linear relationships among numeric variables.
- Strongest correlation observed: `study_hours_norm` vs `engagement` (0.87), confirming the contribution of normalized study hours to engagement.

9. Group Analysis

- Average engagement by `course_completion`:
- `False`: 0.552
- `True`: 0.552
- Minimal difference due to synthetic data design.

10. Filtering High Engagement

- Demonstrated filtering students with `engagement > 0.8`.

11. Functional Transformation

- Applied `lambda` function to classify `past_performance` into High / Medium / Low.

6.2 Extended Analysis

1. Distribution Shape

- Calculated skewness, kurtosis, and IQR:
- `study_hours`:
- `skewness`: 0.33
- `kurtosis`: 0.64
- `IQR`: 5.475
- Interpretation: slight positive skew, light-tailed distribution.

2. Outlier Detection

- No outliers detected for `study_hours` using $2 \times \text{IQR}$.

3. Distribution Check (PDF comparison)

- Empirical z-scores compared to theoretical normal PDF for visual inspection.

4. Contingency Tables

- Multi-dimensional categorical analysis:
 - `gender × course_completion`
 - `age_bucket × course_completion`
 - `performance_level × course_completion`
 - Useful for demographic insights in real datasets.
-

7. Visualization

Visualizations were generated using `visualization.py`:

1. Scatter Plot

- `study_hours` vs `past_performance` colored by `course_completion`.
- Adjusted to **bucketed points** for readability due to 500 records.
- Insight: shows distribution patterns even in synthetic data.

2. Histogram

- `quiz_participation` distribution.
- Highlights skew and variance in student participation.

3. Bar Chart

- Average engagement by `course_completion`.

4. Boxplots

- Visualized numeric variables with median, IQR, and outliers.
- Supports report narrative about variability and extremes.

5. Heatmaps / Contingency Tables

- Visual representation of categorical relationships (`gender`, `age_bucket`, `performance_level`).
-

8. Challenges and Solutions

• Synthetic Data Limitations

- Challenge: no true variation beyond designed ranges.
- Solution: focused on workflow demonstration and modular analysis.

• Variable Naming

- Issue: `completed` vs `course_completion` mismatch.
- Solution: standardized column names in cleaning pipeline.

• Visual Overcrowding

- Initial scatter plot was cluttered.
- Solution: bucketed data points and used heatmaps for categorical visualization.

• Library Dependencies

- Minor issues with `scipy` versions resolved by ensuring consistent virtual environment.
-

9. Key Learnings

- Building **modular, reusable pipelines** enables handling new datasets with minimal changes.
 - **Feature engineering and normalization** are critical for meaningful derived metrics (engagement).
 - Even synthetic data can demonstrate advanced techniques: distribution diagnostics, contingency tables, skewness, kurtosis.
 - Visualization is essential to communicate findings, especially for categorical and bucketed data.
-

10. Recommendations for Future Use

- Apply the pipeline to **real student datasets** to extract actionable insights.
 - Extend analysis to **predictive modeling**: regression, clustering, or classification.
 - Include **automated anomaly detection** for more robust datasets.
 - Expand visualization module with **interactive plots** (e.g., Plotly, Seaborn heatmaps).
-

11. Conclusion

This project demonstrates an end-to-end Python workflow for synthetic student data:

- **Generation → Diagnosis → Cleaning → Analysis → Visualization.**
- Modular and extensible design allows reuse for other datasets.
- Visualizations, descriptive and extended analysis provide a foundation for academic reporting.

Although synthetic, the dataset was handled using realistic techniques. The workflow exemplifies best practices for Pythonic, modular data analysis and sets the stage for future extensions in educational data analytics.

Word count: ~1,520