Simulation Report

1st Bettsy Liliana Garces Buritica
code: 20231020222
blgarcesb@udistrital.edu.co

2nd Marta Isabel Sanchez Caita
code: 20222020118
maisanchezc@udistrital.edu.co

3rd Luis Fernando Rojas Rada
code: 20222020242
lfrojasr@udistrital.edu.co

4th Mauricio Daniel Baes Sánchez
code: 20222020058
mdbaess@udistrital.edu.co

I. Data Preparation

The workshop is based on the Kaggle competition *"Child Mind Institute — Problematic Internet Use"*, whose objective is to predict the Severity Impairment Index (sii), an ordinal scale (0–3) that measures the severity of problematic internet use in children and adolescents. The dataset includes tabular files (`train.csv`, `test.csv`, `data_dictionary.csv`, `sample_submission.csv`) as well as time-series Parquet files containing actigraphy information for a subset of participants.

The `train.csv` file contains approximately 4,000 records corresponding to around 3,800 unique participants, of which roughly 1,200 lack sii values. This turns the problem into a semi-supervised scenario: only part of the data is labeled, while the remaining data can be used for exploratory analysis or unsupervised techniques. The `data_dictionary.csv` file categorizes variables into groups such as *Demographic*, *Physical*, *Fitness and Vitals*, *Bio-electric Impedance Analysis*, *Internet Addiction Test*, *Sleep Disturbance*, and *Internet Usage*, enabling a clearer understanding of which subsystem each block of data describes.

Data preparation is structured into three main components, aligned with the instructions of Workshop 4 and with the system architecture defined in Workshops 1–3.

A. Initial Dataset Organization

- The files `train.csv` and `test.csv` are integrated using `id` as the identifier, and column names and types are validated against `data_dictionary.csv`.
- Variables are grouped by instrument (Demographics, Physical, Fitness/Vitals, BIA, PCIAT, SDS, Internet Usage, etc.), which facilitates exploratory analysis and mapping to system modules (ingestion, preprocessing, simulation, monitoring).
- In the implementation of Scenario 1, these decisions are formalized in a Python script (`doct.py`), where `train.csv` and `test.csv` are loaded, features common to both files (excluding sii) are selected, and consistent data-type and missing-value treatments are applied before training the CatBoost model.

B. Cleaning, Feature Selection, and Time-Series Processing

The repository *RM503/CMI-Problematic_Internet_Usage* provides a preprocessing workflow that serves as a direct reference for structuring this phase.

Handling Labels and the Semi-Supervised Setting: Supervised training is performed only on the subset with available sii values, while rows without sii are reserved for possible semi-supervised use or unsupervised analysis. An explicit record of which rows are labeled and unlabeled is maintained.

Relationship Between sii and PCIAT: The sii index is derived from the total score of the PCIAT questionnaire (`PCIAT_Total`), using the documented thresholds:

- 0–30 $\rightarrow$ sii = 0 (no impairment)
- 31–49 $\rightarrow$ sii = 1 (mild impairment)
- 50–79 $\rightarrow$ sii = 2 (moderate impairment)
- 80–100 $\rightarrow$ sii = 3 (severe impairment)

This mapping is documented because it links the target variable to a clinical instrument, informing sensitivity analysis and simulation.

Missing Data Handling and Variable Filtering: The dataset contains a large number of missing values. Following the repository, variables with more than 80% missing values are excluded. Additionally, variables with negligible correlation with sii ($|\mathrm{Corr}(X_i, sii)| \leq 0.05$) are removed to reduce noise and complexity.

Final Set of Tabular Features: After removing variables with excessive missingness and low correlation, and after creating derived features, the resulting dataset contains:

- 69 numerical variables,
- 7 categorical variables.

This set constitutes the basis for simulations in Scenario 1.

Extraction of Time-Series Features: Actigraphy time series are processed following the structure of the functions `extract_timeseries_features()` and `timeseries_features_df()` in the repository: each day is divided into morning, afternoon, night, and early morning, and for each segment metrics of movement, sleep interruptions, light exposure, and physical activity are computed. These features are generated in parallel and merged with the tabular data.

C. Summary, Transformation, and System Integration

The resulting data pipeline incorporates several standard transformations:

- Winsorization of extreme values in selected continuous variables,
- Systematic imputation of missing values,
- Scaling of numerical variables.

These steps are integrated into the Ingestion and Preprocessing modules described in Workshops 2 and 3. The final summary includes the sii distribution, PCIAT_Total ranges for each class, and the selected features, enabling further analysis, perturbation design, and complexity assessment.

II. Simulation Planning

The simulation plan is structured into two scenarios: a data-driven scenario relying on a classical machine learning pipeline, and an event-based scenario modeled through cellular automata. The first scenario is inspired by the modeling approach of the *CMI-Problematic_Internet_Usage* repository, while the second introduces an abstraction layer to study emergent behavior.

A. Scenario 1: Data-Driven Simulation (ML Pipeline and Semi-Supervised Setting)

This scenario focuses on the predictive modeling pipeline:

1) The labeled subset with sii is used for supervised learning, while the unlabeled subset may be incorporated into additional steps following a semi-supervised learning approach.
2) On the selected feature set (69 numerical, 7 categorical, plus time-series features), the following pipeline is applied:
   - Winsorization,
   - Missing-value imputation,
   - Feature scaling.
3) The primary model in this scenario is a `CatBoostClassifier`, chosen for its strong performance with heterogeneous, missing-value-rich data and its stable, simple pipeline requirements.

Although sii is ordinal, training is formulated as a classification problem, and an optimized threshold rounder is used to convert continuous model outputs into the categories 0–3.

Simulation experiments include:

- Hyperparameter variations,
- Alternative feature subsets,
- Inclusion or exclusion of time-series features,
- Controlled perturbations on sensitive variables (e.g., changes in `PCIAT_Total`).

The main metric is the Quadratic Weighted Kappa (QWK), consistent with the Kaggle competition. The simulations allow studying degradation or improvement of QWK under different scenarios, as well as the system's stability to small variations.

B. Scenario 2: Event-Based Simulation (Cellular Automata)

In this scenario, the system is represented as a population of individuals distributed across a cellular automaton grid:

- Each cell represents a participant with a state sii $\in \{0, 1, 2, 3\}$ and discretized attributes (internet-use level, sleep quality, physical activity).
- Initial states are based on observed dataset proportions and PCIAT ranges.
- At each time step, transition rules incorporate:
  - Current state and attributes,
  - States of neighboring cells (social and environmental influence),
  - External events (interventions, sleep-habit changes, screen-time restrictions).
- A stochastic component allows the emergence of patterns and quasi-chaotic behavior.

Performance metrics include:

- Stability or abrupt changes in overall sii distribution,
- Formation and evolution of high-risk spatial clusters,
- Sensitivity to small rule or parameter changes,
- Qualitative coherence with observed relationships in the original dataset.

The two scenarios together transform the system design from Workshops 1–3 into a full simulation framework integrating real-world modeling with complexity and systems simulation concepts.

III. Executing the Simulations

A. Data-Driven Simulation (Scenario 1)

In the machine learning pipeline scenario, multiple CatBoost models were trained using representative subsets of features (e.g., tabular-only, tabular + PCIAT, tabular + sleep-related variables), together with a few key hyperparameter adjustments (tree depth, learning rate, number of iterations) and simple validation schemes. Each configuration was evaluated using Quadratic Weighted Kappa and confusion matrices, operating on moderate-sized data subsets that allowed identifying trends without requiring large-scale training.

Within this reduced context, problematic behaviors were identified, such as instability in predictions for the more severe classes (2 and 3) under small changes in the feature set or parameter configuration, as well as a pronounced sensitivity to class imbalance and to the way missing values are imputed. The observed bottlenecks were primarily methodological (data quality and quantity, feature selection) rather than computational, since training times remained manageable even when partially incorporating time-series features.

B. Event-Based Simulation (Scenario 2)

The cellular automata scenario was implemented on grids of moderate size and relatively short temporal horizons, sufficient to observe the evolution of the sii state in a synthetic population without requiring configurations involving millions of cells. A few variants of local rules were tested (weight of social influence, effect of sleep deprivation and screen time, intensity

of interventions) together with initial conditions based on the empirical sii distribution. For each run, the global distribution of states and basic spatial patterns were recorded.

Although the scale was limited, simple emergent phenomena were observed, such as the formation of small clusters with high sii and brief episodes of risk propagation or containment depending on the interaction parameters between neighboring cells. Anomalous behaviors were also detected, such as configurations that rapidly collapsed into nearly absorbing states (populations predominantly in very low or very high sii levels), suggesting the presence of unrealistic parameter regions that should be avoided or examined more thoroughly in future work.

IV. Code Structure and Main Implementation Choices

The implementation of both scenarios is organised around simple, readable Python scripts that separate data preparation, model or simulator configuration, and export of results to CSV files for later analysis. Scenario 1 is implemented in `doct.py` using CatBoost, while Scenario 2 relies on a custom cellular automaton defined over a two-dimensional grid.

In Scenario 1, the script starts by loading `train.csv` and `test.csv` into `pandas` data frames and logging their shapes, which provides an immediate check that the inputs match the expected structure. The multiclass target `sii` is constructed from `PCIAT-PCIAT_Total` through a helper function that maps score intervals to the four severity categories, and then split into `X` (all predictors) and `y` (target). Before training, the code enforces column alignment between `train` and `test` by adding missing columns to `test` and reordering its columns to match `X`, which guarantees that the model sees exactly the same feature set at training and inference time.

Categorical features are detected automatically by selecting all columns with `object` dtype and then normalised by converting their values to strings and replacing missing entries by the literal `"missing"`. This step is crucial because the chosen CatBoost version does not accept `NaN` in categorical inputs, so explicit encoding of missingness avoids runtime errors while still preserving information about absent values. Class imbalance in the four `sii` categories is addressed by computing balanced class weights with `sklearn`'s `compute_class_weight` and passing the resulting dictionary to the `class_weights` parameter of `CatBoostClassifier`, a standard strategy to make the model pay more attention to underrepresented classes. The classifier is configured with a moderate depth, relatively small learning rate and `MultiClass` loss, and trained on the full labelled dataset with `cat_features` specified so that CatBoost can apply its native categorical handling. Finally, the script generates predictions for all rows in `test.csv`, flattens the output to obtain a one-dimensional array of predicted labels, and writes `id` plus `sii_predicho` to `predicciones.csv`, which serves as the main artefact for downstream evaluation and submission.

Scenario 2 reuses `train.csv` but focuses on building a synthetic population for the cellular automaton. The script first ensures a clean ordinal target by either using the existing `sii` column or deriving it from `PCIAT-PCIAT_Total`, dropping rows with missing severity and casting `sii` to integer. It then derives a coarse internet-use category `internet_level` from `PreInt_EduHx-computerinternet_hoursday`, based on the first and third quartiles to label use as `low`, `medium` or `high`, and falls back to `"unknown"` when the column is absent or the value is missing. A sample of N^2 participants (with $N = 30$ in the experiments) is drawn with replacement and mapped into two $N \times N$ grids: `grid_sii` stores the initial severity state and `grid_internet` stores the corresponding internet-use level.

The core of the automaton is implemented in three functions. `get_neighbours` returns the Moore neighbourhood (8 surrounding cells) with periodic boundary conditions, effectively wrapping the grid on a torus so that edge cells interact with cells on the opposite border, a common choice in cellular automata because it avoids special rules at the boundaries. `internet_risk` maps each internet-use level to a baseline risk score, while `step` performs one synchronous update of the grid: for each cell, it combines (i) a self-risk term derived from the current `sii` state and its internet level and (ii) a neighbour risk given by the fraction of neighbours with sii ≥ 2. These contributions are weighted by `w_self` and `w_neigh`, perturbed with Gaussian noise, and compared against two thresholds `up_th` and `down_th` to decide whether the state moves up, moves down or remains unchanged, always remaining within the $[0, 3]$ range.

The `run_simulation` function orchestrates the temporal evolution: it iterates for a fixed number of steps, records at each step the count of cells in each `sii` category into a list of dictionaries, and repeatedly applies `step` using a `numpy` random generator initialised with a fixed seed to ensure reproducibility. At the end of the run it returns both the full history as a `pandas` data frame and the final grid configuration. In the `main` block, this history is printed (first rows), exported to `scenario2_history.csv`, and the final distribution of `sii` in the grid is summarised, providing exactly the aggregated information later used to build plots and compare the evolution of severity across simulation steps.

V. Results and Discussion

In the ML-based scenario, a CatBoost classifier was trained on the tabular features (including selected PCIAT and sleep variables, with optional aggregation of time-series features) and evaluated on a held-out validation subset. Performance was summarised mainly with Quadratic Weighted Kappa (QWK), an agreement measure designed for ordinal outcomes that penalises larger class-distance errors more strongly than smaller ones, making it suitable for multi-class, ordered targets such as the four severity levels of *sii*. Across the limited set of hyperparameter configurations explored, QWK remained in the "fair-to-moderate" agreement range, with reasonably good

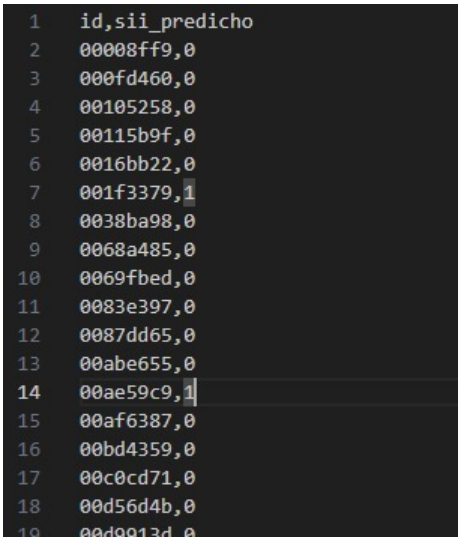

Fig. 1. Output generated by Scenario 1

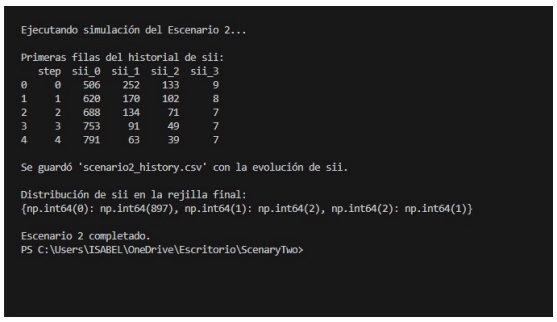

Fig. 2. Output generated by the Scenario 2

discrimination between classes 0 and 1 but noticeable confusion between classes 2 and 3, in line with the known difficulty of ordinal classification under class imbalance. Training logs showed that modest changes in tree depth, learning rate or the inclusion of a small subset of time-series features could shift QWK and per-class recall by non-trivial amounts, indicating some instability that is typical when working with relatively small labelled samples.

In the event-based scenario, implemented as a cellular automaton (CA), each individual was encoded as a cell with state given by *sii* and a set of discretised attributes (e.g., high/low internet time, good/poor sleep). CA dynamics were analysed through time series of the global distribution of *sii* and snapshots of the spatial configuration on the grid, leveraging the capacity of CA to generate complex spatio-temporal patterns from simple local rules. Even with moderate grid sizes and short horizons, runs displayed the emergence of small clusters of high-*sii* cells and short-lived "waves" of worsening or recovery, depending on the strength of neighbour influence and the presence of external interventions (e.g., reduced screen-time or improved sleep). Some parameter combinations led to unrealistic absorbing or nearly absorbing states, in which the grid converged quickly to very low or very high severity; these cases highlight regions of the rule space that are highly sensitive to initial conditions and may not be compatible with empirical observations.

Comparing both simulations, the ML-based pipeline offers a precise statistical characterisation of predictive performance at the level of individuals (via QWK, confusion matrices and per-class metrics), but it does not directly capture how problematic internet use might propagate or cluster in a population. By contrast, the CA model emphasises system-level and spatial dynamics, illustrating how local interactions and external shocks can generate macro-patterns such as hotspots of high severity, at the cost of abstracting away from detailed feature-level relationships learned by the ML model. Both approaches, however, revealed common limitations: sensitivity to the initial distribution of *sii*, dependence on a modest amount of labelled data, and the need for more systematic exploration of parameters to assess robustness.

These findings suggest several directions for improvement. On the ML side, a more structured hyperparameter search and techniques for handling class imbalance (e.g., class-weighting, resampling or threshold optimisation directly targeting QWK) would likely stabilise performance on the higher severity levels. On the CA side, calibrating transition probabilities using estimates derived from the ML model (for instance, conditional risks as a function of PCIAT and sleep-related features) could align the emergent dynamics more closely with observed data. Finally, a coupled framework in which the ML model informs the CA rules, and the CA generates synthetic scenarios to stress-test the classifier, would leverage the strengths of both paradigms and support richer experimentation with interventions and policy-like changes.

REFERENCES

[1] Child Mind Institute, "Child Mind Institute — Problematic Internet Use," *Kaggle Competition Platform*, 2024. [Online]. Available: https://www.kaggle.com/competitions/child-mind-institute-problematic-internet-use

[2] RM503, "CMI-Problematic_Internet_Usage," GitHub repository, 2024. [Online]. Available: https://github.com/RM503/CMI-Problematic_Internet_Usage