Workshop 2

Henry Ricaurte Mora 20221020084
Germán Darío Aya Fuentes 20232020091
Javier Alejandro Penagos Hernández 20221020028

1 Review Workshop #1 Findings

In Workshop #1, a detailed analysis was conducted on the system behind a Kaggle competition focused on predicting student performance in a game-based learning environment. The primary objective of this system is to determine, based on behavioral data collected during interaction, whether a student will answer a question correctly. This prediction is directly related to the level the user reaches in the game, which reflects both skill and progress.

Key Findings

System Components

The analysis identified an architecture composed of multiple interconnected elements forming a complex cyber-physical system:

- **Inputs**: User interaction begins with the *mouse*, whose main events are *hover* and *click*. These are recorded with Cartesian coordinates (x, y) and routed through the system.
- **Processing**: Events are handled by modules such as *screen*, *room*, and *recept*, which interact with the *event handler*. The *event handler* assigns a *name* and *text* to each event and forwards them to the *event registry*.
- **Identification and Context**: Each user is associated with a *user id*, which relates to parameters such as *level*, *configuration* (fullscreen, music option, visual quality), and *session id*.
- **Analytics**: All events are registered and processed by the *data analytic* module, responsible for generating a prediction of the student's *performance*, i.e., their ability to correctly answer questions.

Functional Relationships

The system's structure reveals highly integrated relationships between inputs, user context, and internal processing:

- User actions are progressively transformed and fed into the analytics module.
- System configuration introduces contextual elements that modify user experience and potentially influence behavior.
- The *event handler* and *event registry* serve as central bridges between input and output, consolidating interactions into structured data for prediction.

System Sensitivity

Several variables were identified as highly sensitive, meaning that their variation can significantly affect system behavior:

- **Mouse Events**: These micro-decisions reflect internal processes like attention, emotion, and strategy. While game actions are predefined, their execution varies per user.

- **Response Time**: May represent either reflection or confusion. Its interpretation requires context and is non-linear.
- **User Level**: Indicates progress but results from a sequence of unique decisions, differing across users.
- **Session Duration and Frequency**: Influenced by personal factors like motivation or time availability; affects data volume and quality.
- **System Configuration**: Audio, fullscreen mode, and graphic quality introduce perceptual differences that affect performance in a non-linear way.

Chaos and System Dynamics

The system was characterized as a **complex adaptive system** with **deterministic chaotic dynamics**. This behavior is evident in:

- **Sensitivity to Initial Conditions**: Small differences in students' initial states (prior knowledge, motivation, emotional context) lead to divergent interaction trajectories.
- **Nonlinearity**: There is no direct, proportional relationship between input factors and performance. The same action can yield different outcomes.
- **Emergence and Strange Attractors**: Over time, unique behavioral patterns emerge per student, difficult to predict and highly individualized.

These elements suggest that the system cannot be modeled effectively using simple or linear relationships, and that any design must incorporate strategies to manage its inherent complexity.

2 Define System Requirements

2.1 Functional Requirements

2.1.1 Data Capture and Storage

RF-001 Capture all user interactions including hovers, clicks, and drags with their respective `x,y` coordinates and timestamp

RF-002 Assign a unique `session_id` per game session.

RF-003 Link all interactions to a specific `user_id`.

RF-004 Store user configuration settings including `full_screen`, `hq`, and `music_volume`.

RF-005 Save the `level_group` and question progress.

2.1.2 Data Processing and Normalization

RF-006 Remove erroneous or duplicate clicks (condition: ¡90ms between clicks).

RF-007 Normalize `x,y` coordinates by standardizing to a key resolution.

RF-008 Extract temporal features: time between events, response speed.

RF-009 Extract spatial features: movement patterns like trajectories.

RF-010 Extract contextual features: difficulty level and number of retries.

2.1.3 Prediction Model

RF-011 `screen_coor_x/y`: Mouse position at critical questions.

RF-012 `event_name`: Actions like `cutscene_click` or `map_click`.

RF-013 `elapsed_time`: Cumulative time in session.

RF-014 `hover_duration`: Time spent on interactive elements.

2.2 Non-Functional Requirements

2.2.1 Performance

RNF-001 The system must capture and store user interactions in real time without affecting the user experience.

RNF-002 Data preprocessing and normalization must not exceed 500 ms per batch of captured events.

2.2.2 Reliability

RNF-003 The system must guarantee 99.9% availability during gameplay sessions.

RNF-004 An integrity check must be implemented for each processed data block.

2.2.3 Security

RNF-005 All sensitive data (`user_id`, `session_id`) must be stored and transmitted using AES-256 encryption.

RNF-006 The system must implement role-based access control for data management and visualization.

2.2.4 Ease of Use

RNF-007 The user configuration interface must be accessible and intuitive, allowing easy adjustments to parameters such as volume, resolution, and fullscreen mode.

RNF-008 Key metrics and model outputs must be visualized through interactive dashboards supported by tools such as **Grafana**, facilitating interpretation by non-technical users.

2.2.5 Interoperability

RNF-009 The system must integrate seamlessly with analysis platforms such as **Jupyter**, **Apache Kafka**, **Grafana**, and relational or NoSQL databases.

RNF-010 The predictive model must be exportable in `ONNX` format or similar, ensuring portability across languages and frameworks.

3 High-Level Architecture

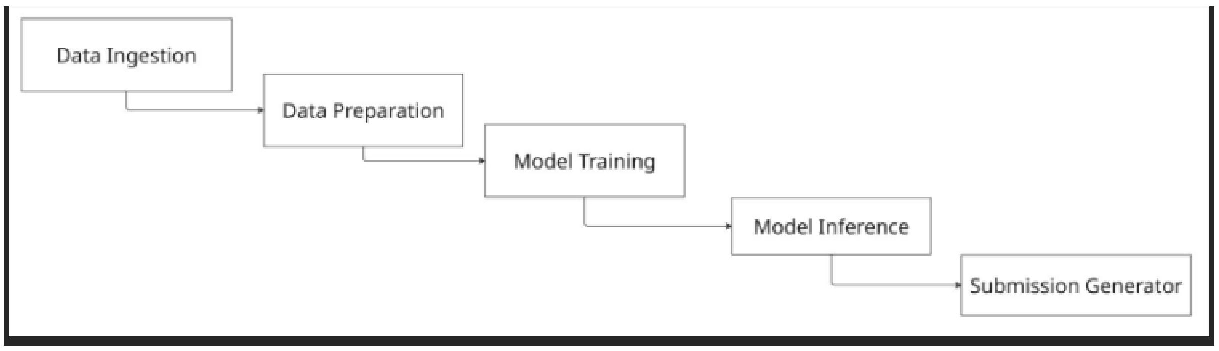

Figure 1: Pipeline for Predictive Model Development

Data Ingestion

First, raw data is loaded. The competition provides four essential files to develop the predictive model:

Train.csv Training dataset containing the features necessary for the model to learn the underlying patterns.

Test.csv Evaluation dataset where you will apply your trained model to generate predictions. Similar to the training set but without including the correct answers.

sample_submission.csv Template that specifies the exact format required to submit your final predictions to the competition platform, ensuring compatibility with the evaluation system.

train_labels.csv Contains the target answers that the model must learn to predict. This file includes the session identifier (session_id), the corresponding question, and a binary indicator that shows whether the question was answered correctly or not.

This data structure, which separates features from labels, is designed to facilitate the development of a binary classification model that can effectively predict whether an answer will be correct or incorrect.

Data Preparation

Once the raw data is loaded, the first step is to clean it to retain only the elements relevant for modeling, in addition to extracting relevant information that is not explicitly available. The goal is to extract features that are likely to correlate with student performance and reflect behavioral patterns in the game data.

Model Training

Once the data is prepared, a machine learning model is selected for training. For this competition, a tree-based model is used, such as RandomForest, GradientBoostedTrees, CART, or DistributedGradientBoostedTrees, as they work well with structured data and effectively manage numerical and categorical variables. Additionally, they are ideal for capturing non-linear relationships in the dataset.

Model Inference

Once the model is trained, inference is performed on the test set to evaluate its predictive accuracy. Multiple models can be tested to compare their performance and determine which offers the most accurate results.

Submission Generator

Formatting predictions in the required format and deploying to LLM Kaggle.

Systems Engineering Principles

Modularity: The system was divided into independent components, each with a specific responsibility. In our case, modules were separated for data ingestion, cleaning and processing, feature engineering, model training, evaluation, and prediction.

Reusability: Thanks to modularity, components can be reused without the need to modify the entire system. For example, if you want to train the model with another algorithm, it is not necessary to change the data ingestion module, and adjustments in processing would be minimal or none, depending on the requirements of the new model.

Scalability: The system was designed to scale horizontally, allowing it to process larger volumes of data without affecting its structure. If more event records or new sessions are received in the future, the current components can adapt without redesigning the entire system.

4 Addressing Sensitivity and Chaos

In this section, we address the sensitive variables and chaotic factors identified during the system analysis in Workshop 1. The student's performance in this context is modeled based on their interactions with the educational game. Multiple sources of variability were identified, including unpredictable user behavior, variable initial conditions, and random in-game events.

In terms of user behavior, we observed fast or erratic clicks, irregular mouse movements, and non-linear navigation paths. Regarding initial conditions, factors such as emotional state, motivation level, and familiarity with the game can significantly affect user interaction. Random events may include variations in question difficulty or game elements that introduce noise into the dataset.

The mitigation strategies implemented to address these challenges include:

1. **Data Normalization and Preprocessing:**
    - Elimination of erroneous and duplicate clicks.
    - Standardization of `x,y` coordinate space.
    - Extraction of temporal features.
2. **Use of TensorFlow Decision Forests (TF-DF):**
   TF-DF is robust to noisy data and can handle both categorical and numerical features without requiring additional encoding. The ensemble nature of decision forests enables the model to capture non-linear and complex relationships among variables, thus improving generalization capabilities.
3. **Monitoring of Unanticipated Conditions:**
   Implementation of observation routines to detect unusual patterns or anomalies in the interaction data.

5 Technical Stack and Implementation Sketch

The following tools will be used to build and train the model:

Programming Language

- **Python:** Main language due to its ease of use and its broad ecosystem for data science and machine learning.

Libraries and Frameworks

- **NumPy and Pandas:** For data manipulation, cleaning, and processing.
- **LightGBM or XGBoost:** For creating tree-based models, highly efficient for structured data.
- **Matplotlib and Seaborn:** For visualization and exploratory data analysis.
- **TensorFlow Decision Forests (optional):** For experimenting with advanced tree models within the TensorFlow environment.

Environment and Tools

- **Jupyter Notebooks:** For implementation, experimentation, and model documentation.
- **Git and GitHub:** For version control and collaborative work.
- **Kaggle Notebooks:** For testing and final submission of the model on the official competition platform.