Workshop No.2 — Systems Engineering Analysis

COVID-19 Global Forecasting
Kaggle Competition

Santiago Vargas Gomez - 20242020139
David Esteban Sanchez Torres - 20221020093
Dilan Guisseppe Triana Jimenez - 20221020100
Daniel Alejandro Castro - 20242020271
Course: Systems Analysis & Design
Professor: Carlos A. Sierra

October 18, 2025

1 Introduction

The COVID-19 Global Forecasting system is designed as part of a data-driven initiative to predict the progression of the pandemic using publicly available datasets. This workshop focuses on analyzing the system through systems engineering principles, emphasizing how each component contributes to the overall functionality and reliability of the predictive process.

The objective of this analysis is to understand how data, models, and feedback mechanisms interact within a dynamic and uncertain environment. Through this examination, the workshop seeks to identify the core elements of the system, define their interrelationships, and establish the operational boundaries that ensure consistency and reproducibility.

Additionally, this document integrates technical and methodological aspects of system design, including architecture, performance requirements, sensitivity factors, and implementation details. By applying a structured analytical approach, it aims to build a comprehensive understanding of the system's behavior, its limitations, and its potential improvements in forecasting accuracy and stability.

2 Review Workshop #1 Findings

Workshop #1 analysis showed that the COVID-19 Global Forecasting system works as a structured pipeline that helps monitor daily epidemiological data by producing forecasts of confirmed cases and deaths. The main insights can be grouped into the following categories:

2.1 Primary System Components

The system has several key parts that make it function properly:

Inputs: Historical datasets (`train.csv`, `test.csv`) that include cumulative counts of confirmed cases and deaths, divided by region and date.

Processes: Steps related to handling data, creating features, training and testing models, and evaluating results through time-series and machine learning methods.

Outputs: The `submission.csv` file, which contains the predicted cumulative cases and deaths, following the competition's official format.

Actors: The participants who design and train the models, and the Kaggle platform that checks submissions and updates the rankings.

All these parts are connected in a cyclical data flow, where the processed data go into the models, predictions are submitted, and the feedback from scores helps improve the next iterations.

2.2 Primary Limitations

The analysis revealed a number of structural limitations:

- Predictions are cumulative and must not decrease over time.
- All outputs must strictly follow the structure defined in `submission.csv`.
- Predictions are limited to the dates specified in the `test.csv` file.
- Only publicly available data can be appended externally.
- Each submission must contain all unique entries for every `ForecastId`, with no entries omitted or duplicated.

Even so, these limits enable consistency, comparability, and fairness across submissions, at the expense of model flexibility and the use of data augmentation strategies.

2.3 Sensitivity and Chaotic Elements

The system was found to be highly sensitive to variations in the data, or chaotic, due to several factors:

- **Incomplete or inconsistent data:** For example, many observations (around 54% null values) lacked province information.
- **Delayed or batch updates:** Reported cases may experience sudden jumps when data are updated in batches, affecting time series stability.
- **External interventions:** Government actions such as lockdowns or reopenings cause abrupt shifts in the pandemic trajectory.
- **Random human behavior:** Small variations in social interaction (e.g., mobility or gatherings) can lead to large cascading epidemiological effects.

These sources of uncertainty suggest that small perturbations in the input can produce large changes in the output. Therefore, validation, feedback, and control mechanisms should be integrated into the system's methodology.

3 Establish System Requirements

Based on the analysis above, the following system requirements are defined to ensure robustness, reliability, and interpretability.

3.1 Performance Requirements

- The system should efficiently manage large-scale time-series datasets (in the order of hundreds of thousands of records).
- Each model training and processing cycle should complete within a reasonable time frame (typically several minutes per iteration).
- The forecasting model, evaluated through RMSLE (Root Mean Squared Logarithmic Error), should achieve low error and competitive accuracy relative to baseline models.

3.2 Reliability Requirements

- The preprocessing process should detect and automatically correct missing or inconsistent entries.
- Data integrity must be preserved during ingestion and transformation.
- Each execution should produce reproducible and consistent results when provided with the same input set.

3.3 Interpretability Requirements

- The model should provide transparent decision logic and interpretable metrics such as feature importance.
- The system should include visualization tools for trends, prediction distributions, and residual errors.
- The workflow should be documented to enable external review and reproducibility.

3.4 Security & Compliance Requirements

- All data sources and integrations must come from publicly available datasets, in compliance with Kaggle competition rules.
- Data and submission deliverables must be transmitted without privacy alterations, adhering to the official submission format.
- Version control (e.g., GitHub) must be used for all code and configurations to ensure full traceability and transparency.

4 High-Level Architecture

The high-level architecture of the COVID-19 forecasting system provides an overview of how the main modules interact to transform raw data into validated predictions. The system follows fundamental systems engineering principles such as modularity, scalability, maintainability, and traceability. This architecture ensures that each component functions independently while remaining part of a coherent and adaptable structure.

4.1 General Architecture Description

The architecture is designed as a modular pipeline composed of five main stages: data ingestion, preprocessing, modeling, evaluation, and feedback. Each module performs a specific role in the data transformation process, allowing the system to operate efficiently and iteratively improve over time.

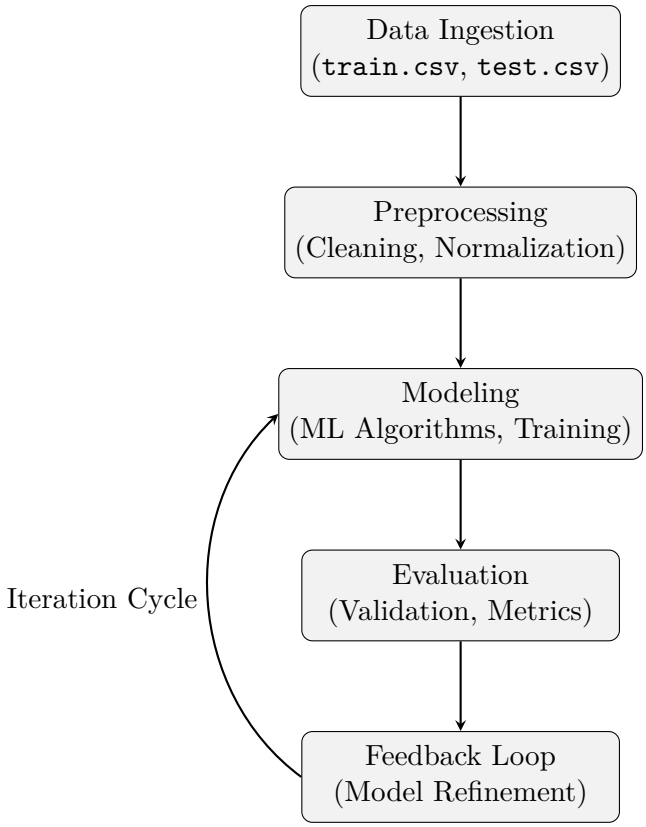

Figure 1: High-Level Architecture of the COVID-19 Forecasting System.

The diagram illustrates the modular structure of the system. Data flows sequentially through the modules, starting from ingestion and preprocessing, continuing to modeling and evaluation, and finally returning through the feedback loop for continuous improvement.

4.2 Main System Components

The system is divided into five core modules, each responsible for a distinct part of the data processing pipeline. This modular approach allows updates or improvements to individual components without affecting the rest of the architecture.

- **Data Ingestion:** This module manages the import of input datasets (`train.csv` and `test.csv`) from Kaggle or local sources. It verifies file structure, consistency, and accessibility, providing the foundation for the entire data pipeline.
- **Preprocessing:** Responsible for cleaning and transforming the data. It removes missing values, normalizes numeric variables, unifies date formats, and standardizes region identifiers. The output is a consistent dataset suitable for machine learning algorithms.
- **Modeling:** The analytical core of the system. In this stage, algorithms such as *Random Forest Regressor*, *Gradient Boosting*, or *XGBoost* are trained using preprocessed data. The models learn temporal and spatial patterns that enable prediction of confirmed cases and fatalities.
- **Evaluation:** The evaluation module validates predictions against known data and competition metrics. Submissions are generated in the `submission.csv` format and uploaded to Kaggle for automatic scoring, ensuring consistency and comparability.

- **Feedback:** This module collects the evaluation results and identifies potential improvements in data handling, feature engineering, or model configuration. It closes the learning cycle by sending refined parameters and insights back to earlier modules.

Together, these components form an integrated but flexible architecture capable of supporting iterative development, reproducibility, and scalability.

4.3 Interactions Between Modules

The interaction between modules defines the logical data flow from input to prediction output. The process begins with raw datasets entering the ingestion module, which ensures data readiness and sends it to preprocessing. After data cleaning and transformation, the processed dataset flows to the modeling module, where algorithms are trained and used to generate predictions.

The prediction results are then passed to the evaluation module, which compares the forecasts against real-world data and generates performance metrics. These metrics feed into the feedback module, where developers analyze results and decide on possible refinements, such as improving feature engineering or model tuning. This cyclical process ensures continuous improvement in model performance and system reliability.

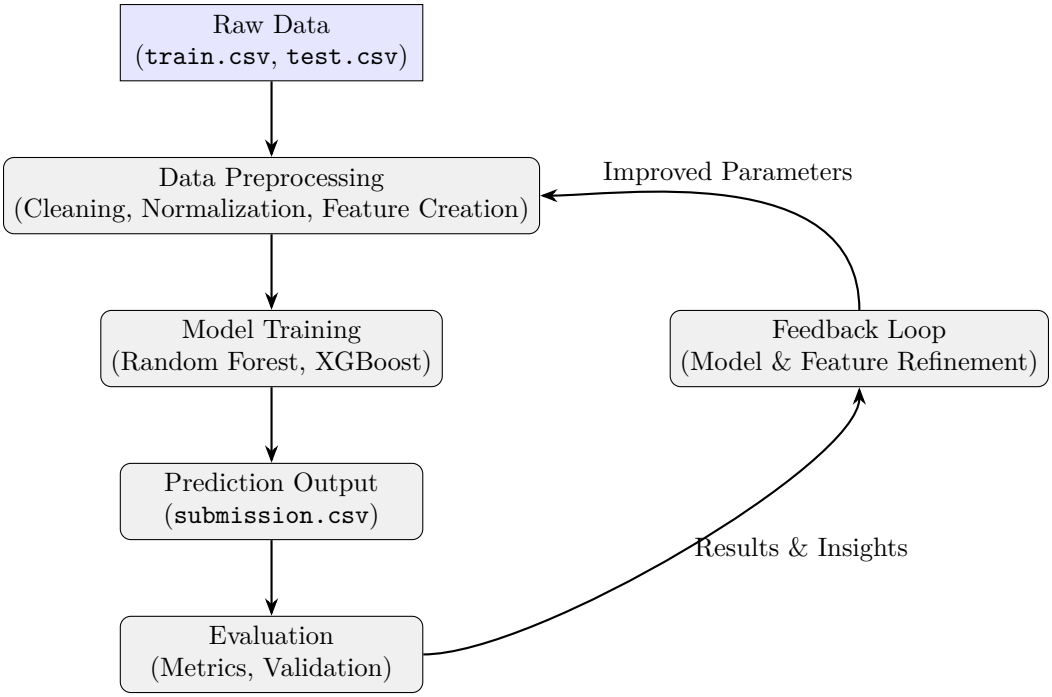

Figure 2: Data Flow Between Modules of the COVID-19 Forecasting System.

The data flow diagram emphasizes the sequential and cyclical nature of the system. Each module passes structured outputs to the next, ensuring traceability and consistency, while the feedback mechanism drives iterative refinement aligned with systems engineering principles.

5 Sensitivity and Chaos

5.1 Sensitivity Factors

As discussed in the previous workshop, the COVID-19 forecasting system exhibits multiple sensitivity points related to the data's nature and the external environment in which it operates:

- **Data variability and noise:** The analyzed datasets sometimes lead to inconsistencies because of their multiple sources of origin. This data contains inconsistent reporting, missing values, and sudden corrections, leading to what we can call "noise points" that affect model stability and evaluation metrics.
- **Temporal Dependence and Drift:** One of the main problems related to the datasets of this model is their high dependency on time and how they change suddenly. When infection trends change rapidly, the model's learned patterns become outdated, causing prediction drift.
- **Feedback Effects:** Forecast results indirectly influence human and policy behavior (such as lockdown decisions), creating feedback loops that alter the very data being modeled. Basically, data feedback can be intervened by external agents, leading to different resolution paths.
- **Hyperparameter and Model Sensitivity:** Small changes in hyperparameters (such as learning rate, max depth, or lag window size) can lead to large variations in predicted outcomes, revealing the sensitive nature of the data.

5.2 Control and Mitigation Mechanisms

Based on the proposed integrated architecture, we can handle the validation processes, error monitoring, and feedback-driven adjustments. Following the previous architecture, we start by the data ingestion, incorporating schema validation and completeness checks to prevent structural inconsistencies. After this, we continue with the preprocessing, applying outlier detection, smoothing, and normalization to minimize the impact of noisy or irregular records. In the modeling phase, systematic hyperparameter tuning ensure that the model generalize well across temporal segments, while increasing the model's robustness. At this point, we have to evaluate the performance through Root Mean Squared Logarithmic Error (RMSLE) and residual analysis to detect anomalies or drifts. As a final step, with the feedback module we implement an adaptive retraining process that updates the model when performance degradation is detected, ensuring the system remains responsive and stable

5.3 System Stability Strategies

Through the version control we ensure that every dataset, and its corresponding preprocessing steps, and model configuration are traceable and reproducible. By monitoring every continuous error,we can detect and allow the system to identify performance degradation or anomalous data behavior earlier. The feedback loop operates as a self-correcting mechanism that trigger parameter adjustments when the metrics variation exceed predefined limits. Additionally, gradual model updates, ensemble averaging, and comprehensive logging prevent abrupt performance shifts and maintain consistent forecasting accuracy over time.

6 Technical Stack and Implementation Sketch

Based on the previously defined system architecture, this section describes the technologies and workflow that underpin the implementation of the COVID-19 prediction system. The goal is to maintain a balance between performance, transparency, and reproducibility throughout the entire process, from data loading to the generation of final predictions.

6.1 Technical Stack

Python was used to develop the system due to its adaptability and the strength of its data science and machine learning ecosystem. This language enables effective data management, rapid prototyping, and seamless integration with analytical tools.

Large volumes of data are efficiently handled using **Pandas** and **NumPy** for data manipulation and numerical processing. The modeling process is carried out with **Scikit-learn**, a library that provides multiple regression algorithms and evaluation tools, making it ideal for time series forecasting tasks. Exploratory analysis and a deeper understanding of model performance are achieved through visualization libraries such as **Matplotlib** and **Seaborn**.

The development environment is **Jupyter Notebook**, which combines code, documentation, and visualizations in a single interface to facilitate collaboration and experiment tracking. Additionally, the **Kaggle API** is used to automate dataset retrieval and submission of predictions, thereby streamlining the workflow and ensuring compliance with competition requirements.

All notebooks and scripts are managed through **GitHub**, which enables version control, ensures code transparency, and supports effective teamwork.

Overall, this technology stack provides a strong foundation for model training, validation, and iteration, allowing flexibility and scalability in the overall system design.

6.2 Implementation Sketch

A structured and modular methodology is followed during implementation to ensure clarity and maintainability at every stage. The process begins with **data loading**, where `train.csv` and `test.csv` are read and validated using Pandas. Next, **data preprocessing** is performed to ensure consistency across datasets, including cleaning missing values, standardizing variables, and unifying region or country names. When necessary, **feature engineering** is applied to improve the model's predictive capacity by adding time-based or location-based variables.

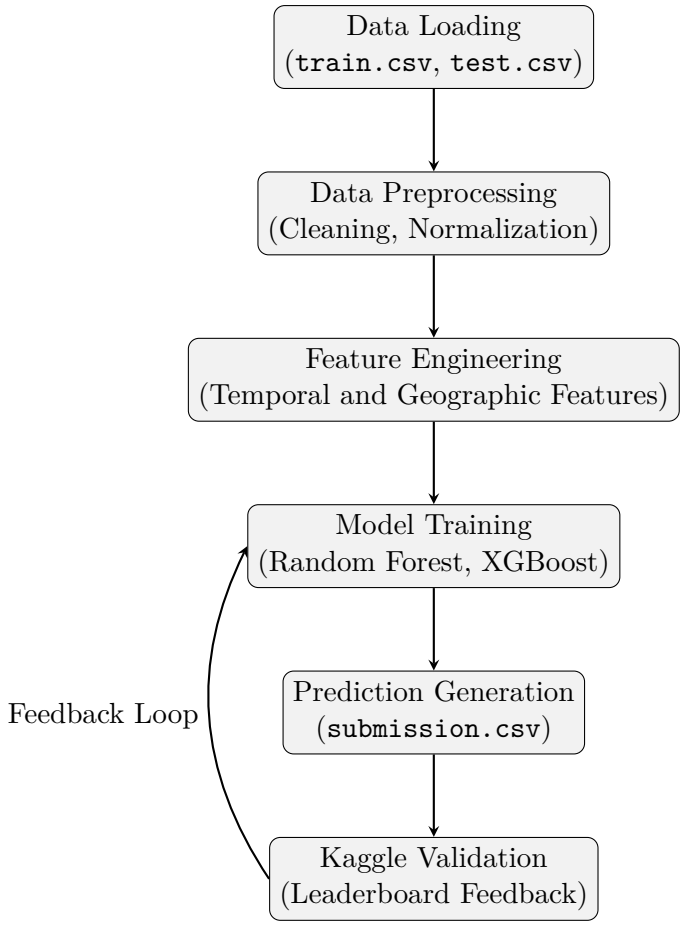

Figure 3: Implementation workflow of the COVID-19 forecasting system.

After the data is prepared, **machine learning algorithms** such as *Random Forest Regressor*, *Gradient Boosting*, or *XGBoost* are used to train the model. During this stage, hyperparameters are fine-tuned and cross-validation is applied to optimize accuracy and avoid overfitting.

Once trained, the system applies the model to the `test.csv` dataset to generate predictions for `ConfirmedCases` and `Fatalities`. The results are then exported to the `submission.csv` file, following the official format specified by the Kaggle competition.

The **Kaggle API** is used to upload the submission file, where it is automatically validated and the leaderboard score is updated. These results serve as a **feedback mechanism** that helps identify possible improvements in preprocessing or model configuration. Through this iterative cycle of analysis, prediction, and refinement, the system's accuracy and performance are progressively enhanced.

The modular design ensures that modifications in one part of the system (such as model selection or feature creation) do not affect other components. This structure promotes scalability, maintainability, and adaptability, in accordance with the fundamental principles of systems engineering and data-driven design.

7 Conclusions

The analysis of the COVID-19 Global Forecasting system demonstrates how systems engineering principles can be effectively applied to a real-world data modeling challenge. The modular architecture and iterative workflow allowed the system to manage large volumes of epidemiological data while maintaining reproducibility and adaptability.

The study highlighted the importance of integrating validation, feedback, and control mech-

anisms to mitigate sensitivity and chaotic behavior in the data. Small variations in inputs or external factors can cause significant changes in outputs, reinforcing the need for continuous monitoring and adaptive model retraining.

From a technical standpoint, the use of **Python**, **Jupyter Notebook**, and machine learning libraries such as **Scikit-learn** and **XGBoost** provided an efficient and transparent environment for experimentation and collaboration. Moreover, the adoption of version control and public datasets ensured transparency, compliance, and traceability throughout the development process.

In conclusion, this project not only fulfills the competition's requirements but also illustrates the relevance of systems analysis in designing robust, scalable, and interpretable forecasting models. The insights obtained here serve as a foundation for future work focused on improving data integration, enhancing predictive precision, and strengthening the stability of complex adaptive systems.