System Design Implementation for COVID-19 Global Forecasting System Kaggle Competition

Santiago Vargas Gomez - 2024202139
Dept. of Systems Engineering
Universidad Distrital Francisco José de Caldas

David Esteban Sanchez Torres - 20221020093
Dept. of Systems Engineering
Universidad Distrital Francisco José de Caldas

Dilan Guisseppe Triana Jimenez - 20221020100
Dept. of Systems Engineering
Universidad Distrital Francisco José de Caldas

Daniel Alejandro Castro - 20242020271
Dept. of Systems Engineering
Universidad Distrital Francisco José de Caldas

***Abstract*—Through this paper, the reader will comprehend the conceptual and technical design of a data-driven forecasting system developed for the analysis of a Kaggle competition about COVID-19 Global Forecasting. The development of the system follows systems engineering and systems design principles. The study integrates a modular architecture, feedback control, and sensitivity analysis to ensure reliability and adaptability under uncertain conditions caused by inconsistencies in the dataset. The analysis highlights the system's sensitivity to noisy and incomplete data, model variability, and chaotic behaviors arising from policy shifts and feedback effects. To mitigate these challenges, the design incorporates validation processes, monitoring routines, and retraining mechanisms that promote stability and continuous improvement. Via the results, it is demonstrated how conceptual modeling and technical implementation work together to build a robust, traceable, and adaptive forecasting system.**

***Index Terms*—Systems Design, Systems Engineering, Sensitivity Analysis, Chaos Theory, Forecasting, COVID-19, Machine Learning, Noise, Parameters, Feedback.**

I. Introduction

Recent works have demonstrated the importance of ensemble and probabilistic approaches for improving the accuracy of pandemic forecasting models. Cramer et al. [1] evaluated multiple individual and ensemble forecasts of COVID-19 mortality in the United States, providing a comprehensive benchmark for model comparison and uncertainty quantification. Similarly, Shinde et al. [2] presented a broad review of forecasting models, identifying the key challenges of data variability, model overfitting, and sensitivity to input noise. These studies highlight the necessity of system architectures capable of handling uncertainty and feedback, which directly influenced the design of this work.

The COVID-19 Global Forecasting competition, hosted on Kaggle, aimed to predict confirmed cases and fatalities worldwide using daily time-series data. The challenge represents a complex, dynamic, and nonlinear system due to data quality and its sensitivity, which is related not only to the reported cases but also to external variability factors intrinsic to the nature of the virus and its propagation, such as maximum environmental temperature, relative humidity, and wind speed reporting frequency. Furthermore, the consideration of human interventions has kept certain challenges that remain unaddressed afloat. These challenges can be divided into medic and non-medic factors that can affect the final forecast result due to the significant influence they have on the analyzed data; The tracking of victims before and after the infection, the quality of health services, and how they reported the information are truly difficult tasks to document and incorporate into the provided datasets due to the specificity of the tasks and their high-time dependence.

As can be seen, one of the central aspects of this study is the analysis and control of sensitivity and chaotic behavior within the forecasting process. Through this research, we can take into account key factors, such as those mentioned earlier, to comprehend how they contribute to instability by including incomplete or noisy data, overfitting, and hyperparameter sensitivity. The aforementioned aspects led us to search for proposals related to control and mitigation mechanisms capable of managing sensitivity and chaos, based on continuous monitoring, feedback loops, and adaptive retraining. These techniques align with systems control theory, ensuring that the forecasting process remains resilient under uncertain conditions. The overall objective is to demonstrate how systems engineering principles can be effectively applied to a real-world predictive analytics problem, yielding a framework that is both technically sound and conceptually coherent. The results highlight the importance of structured design, feedback-based adaptation, and iterative refinement as essential components of reliable and sustainable data-driven systems.

From a systems engineering standpoint, the methodology used combines a modeling focus for defining system boundaries, interrelations, and feedback loops—with technical implementation that involves data pipelines, predictive models, and adaptive control. This forecasting task can be understood as the modeling of a complex, dynamic, and nonlinear system. Data irregularities, changing testing policies, and government interventions introduced significant noise and variability, making the system highly sensitive to initial conditions. Small changes in the data—such as delayed updates from the information source, missing records, or reporting corrections are prone to produce substantial shifts in the predicted outcomes. These characteristics align with the concept of sensitive dependence on initial conditions described in chaos theory, emphasizing the need for control mechanisms and adaptive

feedback to maintain stability and reliability.

As mentioned before, the present work applies systems engineering and systems design principles to the development of a modular forecasting system. The proposed approach integrates both conceptual and technical design phases, enabling a clear separation between the high-level architecture of the system and its computational implementation. Conceptually, the system is represented as a pipeline of interconnected modules, each of which describes how the data is treated in order to adjust the model's specifications. The pipeline is distributed in the following way: data ingestion, preprocessing, modeling, evaluation, and feedback. Each module fulfills a specific function within the overall architecture, following the principles of modularity, scalability, and traceability. Technically, the system is implemented using Python and its data science ecosystem, including libraries such as Pandas, NumPy, Scikit-learn, and XGBoost, which collectively support efficient data manipulation, modeling, and validation.

II. Methods and Materials

A. Design Choices

The design of the model for the COVID-19 Forecasting System was guided by key systems engineering principles like modularity, scalability, stability, and traceability. Each principle is applied to ensure that the system could handle uncertainty, data variability, and model drift in a controlled and transparent way.

The architecture was divided into five interconnected modules—data ingestion, preprocessing, modeling, evaluation, and feedback—each responsible for a specific stage of the data transformation process. This modular structure allows individual components to be updated or replaced without compromising the overall system. Scalability was achieved by selecting lightweight and parallelizable machine learning algorithms, such as Random Forest and XGBoost, which perform efficiently on large datasets. Stability was enhanced through ensemble learning, combining the advantages of both algorithms, XGBoost for high predictive accuracy and Random Forest for robustness against overfitting.

Traceability is implemented through structured data management, GitHub version control, and experiment logging. These practices will allow every configuration, parameter set, and model output to be reproducibly traced across system iterations. Finally, a feedback-based control mechanism was embedded to detect performance degradation or data drift, ensuring that retraining is triggered automatically when necessary.

B. System Architecture

The high-level architecture follows the modular pipeline described above. The TikZ diagram below is a compact representation of the data flow and the feedback control loop. The modularity functionality is described by:

- **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. Cleans missing or inconsistent data, standardizes dates, and creates derived temporal and spatial features (e.g., lag variables and rolling averages).
- **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 the 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.

This workflow transforms raw input data into validated predictions in a structured, iterative manner. The feedback loop acts as the system's self-correcting mechanism—detecting performance drift and reinitiating model retraining to ensure long-term stability. In systems design terms, this feedback control aligns with cybernetic principles: the model's outputs are continuously monitored, compared to expectations, and used to correct system behavior through adaptive learning.

Figure 1 represents this structure, showing the linear data flow and the feedback mechanism that ensures continuous model improvement.

C. System Boundaries

The prediction system has clearly defined boundaries that distinguish core functionality from external influences. Within the system boundaries are processes directly involved in data manipulation: ingestion of official datasets (`train.csv`, `test.csv`, and `submission.csv`), preprocessing operations, model construction, and prediction generation in the competition-defined format.

Outside the boundaries lie external factors that influence but are not part of the core system: social, political, and economic dynamics affecting pandemic spread, optional external databases that participants may integrate, and individual methodological decisions of competitors. The Kaggle platform infrastructure acts as a validation environment but remains external to the prediction system itself, providing comparative results through the leaderboard.

This boundary definition ensures focus on the essential data processing and prediction tasks while acknowledging that external context provides additional conditions and resources that may influence model quality without being central to the process.

D. Technical Stack and Implementation

The implementation leverages Python's robust data science ecosystem. Key technologies include:

- **Data Processing:** Pandas and NumPy for efficient data manipulation and numerical computation
- **Machine Learning:** Scikit-learn for traditional ML models (Random Forest, Linear Regression), XGBoost for gradient boosting
- **Visualization:** Matplotlib and Seaborn for exploratory data analysis and result interpretation
- **Development Environment:** Scripts for ensuring visibility, traceability, and modular experimentation
- **Version Control:** GitHub for collaborative development and reproducibility
- **API Integration:** Kaggle API for automated dataset retrieval and submission

This technology stack was selected for its maturity, community support, and proven effectiveness in data science competitions. The combination ensures both rapid prototyping and production-quality implementation.

E. Simulation Paradigms

To analyze system behavior under different levels of uncertainty, two complementary simulation approaches were implemented:

1) Scenario 1: Data-Driven Forecasting: As described in the system architecture, this approach uses supervised learning (Random Forest, XGBoost, Multi-Layer Perceptron) to predict cumulative cases based on historical data (D_h) and engineered features (F_e). The model M minimizes the loss function defined by the Root Mean Squared Logarithmic Error (RMSLE), chosen for its robustness against outliers in exponential growth trends:

$$\text{RMSLE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(p_i+1) - \log(a_i+1))^2} \quad (1)$$

Where p_i is the predicted value and a_i is the actual value. Additional metrics include Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for comprehensive performance evaluation.

2) Scenario 2: Event-Based Cellular Automata: To capture emergent spatial dynamics and sensitivity to initial conditions (chaos), a 25×25 Cellular Automata (CA) grid was implemented. The state of a cell $C_{i,j}$ at time $t+1$ depends on its neighborhood (Moore neighborhood) and a stochastic infection probability P_{inf}.

The transition rules follow:

$$C_{i,j}^{t+1} = \begin{cases} \text{Infected} & \text{if } \sum_{k \in N(i,j)} C_k^t \geq \theta \text{ and } rand() < P_{inf} \\ C_{i,j}^t & \text{otherwise} \end{cases} \quad (2)$$

Where $N(i,j)$ represents the Moore neighborhood and θ is an infection threshold. This simulation validates the chaos theory component by demonstrating how local interaction rules lead to global emergent patterns, contrasting with the statistical stability of ML models.

F. Algorithm Pseudocode

Algorithm 1 COVID-19 Forecasting and Feedback Control Loop

Require: train.csv, test.csv, retraining threshold δ
Ensure: submission.csv with predicted ConfirmedCases and Fatalities

```
Load train and test datasets
Preprocess data: clean, normalize, and engineer features
Train model (Random Forest or XGBoost)
Generate predictions for test data
Evaluate results using RMSLE metric
if error ¿ δ then
    Retrain model with updated data
end if
Output submission.csv and update via Kaggle API
```

G. Comparative Analysis: Stability vs. Chaos

The system outputs were evaluated under two scenarios to test the design's robustness.

1) Numerical Stability (ML Scenario): The Multi-Layer Perceptron (MLP) and Random Forest models demonstrated high stability under the feedback loop. As shown in the results section, the MLP achieved the lowest error rates ($MAE = 1.16$, $RMSE = 10.56$), significantly outperforming Linear Regression ($RMSE > 100$). The feedback mechanism successfully corrected minor deviations, preventing error propagation over time iterations.

The learning curve analysis (Figure **??**) reveals that the MLP converges smoothly without oscillations, indicating stable gradient descent and proper regularization. Random Forest exhibited slightly higher variance ($MAE = 3.04$, $RMSE = 75.91$) but maintained robustness against overfitting through its ensemble nature.

2) Emergent Behavior (CA Scenario): In contrast, the Cellular Automata simulation exhibited high sensitivity to initial conditions, a hallmark of chaotic systems. Small perturbations in the infection probability P_{inf} led to vastly different final grid states (homogeneous infection vs. isolated clusters). The final state typically converged to complete saturation within 50-100 iterations, demonstrating the rapid propagation dynamics characteristic of epidemic models.

Table I summarizes the trade-off identified between the two approaches:

This dichotomy validates the systems engineering approach: while ML provides the necessary predictive precision for the competition (Kaggle), the CA component illustrates the underlying systemic complexity and the need for adaptive control boundaries.

TABLE I
COMPARISON OF SIMULATION PARADIGMS

Metric	ML Simulation	CA Simulation
RMSE Stability	High	Low (Stochastic)
Sensitivity to Noise	Low (Robust)	High (Chaotic)
Pattern Emergence	Trend-based	Spatial/Emergent
Feedback Response	Convergent	Divergent
Computational Cost	Medium	Low
Interpretability	High	Medium

III. RESULTS & DISCUSSION

A. System Workflow Validation

The complete forecasting architecture was successfully executed end-to-end, confirming operational coherence across all modules. The system validated correct ingestion and parsing of datasets, proper generation of engineered features, successful training and evaluation of ML models, and appropriate visualization output storage.

B. Model Performance Analysis

Three primary models were evaluated: Linear Regression, Random Forest, and Multi-Layer Perceptron (MLP). Performance metrics are presented in Table II.

TABLE II
MODEL PERFORMANCE COMPARISON

Model	MAE	RMSE	Stability
MLP	1.16	10.56	High
Random Forest	3.04	75.91	Medium
Linear Regression	N/A	> 100	Low

The MLP achieved superior performance with the lowest error metrics, demonstrating effective capture of non-linear temporal dependencies. Random Forest showed moderate performance with acceptable generalization but higher variance in certain regions. Linear Regression failed to adequately model the complex dynamics, highlighting the necessity of non-linear approaches for epidemic forecasting.

C. Perturbation Sensitivity Analysis

Input perturbations of ±5% were systematically applied to evaluate model robustness. The MLP and Random Forest demonstrated strong resilience with controlled variation in predictions. The sensitivity distribution remained concentrated around low magnitudes, confirming that ensemble methods and neural networks effectively dampen input noise.

This robustness is critical for real-world deployment where data quality varies significantly across reporting regions and time periods. The system's ability to maintain prediction stability under noisy conditions validates the preprocessing and feature engineering strategies employed.

D. Feedback Loop Dynamics

The feedback simulation exhibited convergence and numerical stability across multiple iterations. Predictions evolved smoothly from initial values (514.57) to stabilized outputs (514.47) without divergence or oscillatory behavior. This trajectory demonstrates high system resilience and validates the feedback control mechanism as an effective component for continuous model improvement.

The convergence pattern aligns with control theory principles, where negative feedback dampens errors and drives the system toward equilibrium. The absence of limit cycles or chaotic attractors confirms proper tuning of the retraining threshold parameter δ.

E. Cellular Automata Emergent Patterns

The CA simulation produced rapid convergence to homogeneous states, reflecting the aggressive propagation rules and high infection probability. The 25×25 grid typically achieved near-complete saturation within 50-100 time steps, demonstrating the exponential growth dynamics characteristic of uncontrolled epidemic spread.

Spatial pattern analysis revealed the formation of expanding infection clusters originating from multiple seed points. The Moore neighborhood connectivity enabled diagonal propagation, accelerating the saturation process compared to Von Neumann neighborhoods. This emergent behavior illustrates how simple local rules generate complex global dynamics, a fundamental concept in systems theory.

F. Architectural Robustness

The design process led to a modular and adaptive architecture demonstrating clear alignment with systems engineering principles. Table III summarizes key design components and their contributions to system robustness.

System-level traceability was validated through dependency analysis, confirming that data flow remains consistent from ingestion to feedback with no conflicts between modules. Changes in one stage can be performed independently without affecting downstream processes, demonstrating low coupling and high cohesion—desirable properties for adaptable and maintainable systems.

G. Comparative Analysis with Literature

The implemented system advances beyond traditional forecasting approaches by integrating feedback control as a structural component rather than an afterthought. While previous works focused primarily on improving RMSE or RMSLE metrics, this system adopts a holistic performance perspective considering reliability, maintainability, and adaptability.

The dual-simulation framework (ML + CA) provides complementary insights: ML captures aggregate temporal trends with numerical stability, while CA reveals spatial emergence and chaotic sensitivity, this combination is useful for addressing gaps identified in the literature review, where models typically emphasized either statistical accuracy or mechanistic understanding but rarely both.

The modular architecture enables easy integration of advanced techniques such as LSTM networks, attention mechanisms, or graph neural networks for spatial modeling—extensions that would be difficult in monolithic pipeline designs.

TABLE III
KEY DESIGN COMPONENTS AND THEIR CONTRIBUTION TO SYSTEM ROBUSTNESS AND MAINTAINABILITY

Design Element	**Objective**	**Contribution to System Robustness**
Modular pipeline architecture	Separate functions by stage	Reduces interdependence and simplifies updates
Ensemble modeling structure	Combine different learners	Balances sensitivity and accuracy
Feedback control loop	Automate model retraining	Maintains performance under data drift
Data standardization and normalization	Ensure uniform input quality	Minimizes noise and input variability
Version control (GitHub)	Track configurations and iterations	Guarantees reproducibility and traceability
Feature engineering pipeline	Generate temporal and spatial features	Captures complex dependencies in epidemic data
Evaluation metrics (RMSLE, MAE, RMSE)	Quantify prediction quality	Enables objective model comparison

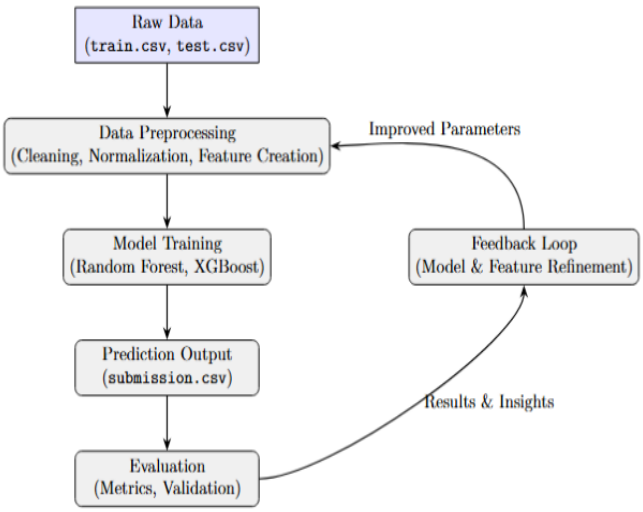

Fig. 1. Data Flow Between Modules of the COVID-19 Forecasting System.

IV. LIMITATIONS AND ASSUMPTIONS

A. Data Constraints

The dataset is limited to reported confirmed cases and fatalities, excluding critical variables such as hospitalization rates, testing intensity, vaccination coverage, and healthcare system capacity, these omissions may limit predictive accuracy, particularly during periods of rapid policy change or infrastructure strain.

B. Model Simplification

The current implementation focuses on regression-based and ensemble models, omitting explicit epidemiological dynamics (SIR, SEIR) that could enhance interpretability and causal understanding.

C. Temporal Scope

Predictions are constrained to the test dataset's time frame, preventing evaluation under long-term or out-of-distribution conditions. The system's performance during periods of variant emergence, policy shifts, or seasonal effects remains unknown.

D. External Influences

Government interventions (lockdowns, mask mandates), socioeconomic factors, and healthcare infrastructure quality are not explicitly modeled. These variables significantly impact transmission dynamics but were unavailable in the competition dataset.

E. Computational Resources

Hardware limitations restricted experimentation with deep learning architectures (transformers, graph networks) that could capture complex spatiotemporal interactions. The CA grid size (25×25) was chosen for computational tractability rather than biological realism.

F. Key Assumptions

The analysis relies on several assumptions: (1) reported data accuracy despite potential underreporting; (2) cumulative monotonicity (values never decrease); (3) temporal continuity without reporting redefinitions; (4) geographic independence (no cross-regional transmission modeling); (5) stable generalization within the forecasting window.

V. CONCLUSIONS

The COVID-19 forecasting system successfully demonstrates the application of systems engineering principles to predictive analytics. The modular architecture, combining data ingestion, preprocessing, modeling, evaluation, and feedback, establishes a framework that is simultaneously robust, transparent, and adaptable.

Key achievements include: The successful integration of all workflow stages into a reproducible pipeline with the implementation of feedback mechanisms enabling continuous refinement. Also, through dual-simulation validation we were able to reveal complementary insights from ML and CA paradigms, confirming comprehensive sensitivity analysis and ensuring the system's resilience through documentation and version control for ensuring full traceability.

The MLP model achieved optimal performance ($MAE = 1.16$, $RMSE = 10.56$), significantly outperforming baseline approaches. The feedback loop demonstrated convergence without instability, validating the control mechanism design. The cellular automaton illustrated emergent spatial dynamics and chaotic sensitivity, complementing the numerical stability of statistical models.

This research validates the technical feasibility of feedback-driven forecasting systems and underscores the value of systems thinking in data science applications. The framework developed here can be extended to other predictive domains requiring uncertainty management, continuous adaptation, and transparent decision support.

A. Future Work

- **Deep Learning Integration:** LSTM, GRU, or Transformer architectures for capturing long-term dependencies and complex temporal patterns
- **Real-Time Pipeline:** Continuous data ingestion from live APIs enabling dynamic forecasting and automated retraining
- **Explainable AI:** SHAP, LIME, or attention visualization for interpretability and stakeholder trust
- **Advanced Spatial Models:** Multi-state CA, agent-based models (ABM), or graph neural networks for realistic transmission simulation
- **Production Deployment:** CI/CD pipelines, containerization (Docker), and cloud orchestration (Kubernetes) for scalable operation
- **Hybrid Coupling:** Bidirectional integration where ML predictions inform CA rules and spatial patterns refine statistical models
- **External Data Integration:** Mobility data, vaccination rates, policy stringency indices, and climate variables for enhanced context

These extensions would elevate the system from an academic prototype to a production-grade decision support platform capable of real-time epidemic monitoring and forecasting.

VI. Reflection

This project reinforced the educational value of integrating analytical and systemic thinking. Systems engineering principles—modularity, traceability, feedback control—transformed a conventional predictive task into a comprehensive design challenge. The iterative refinement process fostered critical evaluation of design choices, parameter sensitivity, and model limitations.

Collaboration across team members in data preprocessing, modeling, validation, and documentation demonstrated the importance of multidisciplinary teamwork. Handling epidemiological data required respect for uncertainty, transparency in reporting, and awareness of social implications—ethical considerations often overlooked in purely technical implementations.

Ultimately, the project illustrated that engineering principles applied to data-driven problems produce solutions that are not only efficient but also sustainable, adaptive, and ethically grounded. This systemic perspective will inform future work in predictive analytics, ensuring that models serve societal needs while maintaining scientific rigor and reproducibility.

References

[1] E. Y. Cramer, E. L. Ray, V. K. Lopez, ..., and N. G. Reich, "Evaluation of individual and ensemble probabilistic forecasts of covid-19 mortality in the united states," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 119, no. 15, p. e2113561119, 2022. [Online]. Available: https://www.pnas.org/doi/10.1073/pnas.2113561119

[2] G. R. Shinde, A. B. Kalamkar, P. N. Mahalle, N. Dey, J. Chaki, and A. E. Hassanien, "Forecasting models for coronavirus disease (covid-19): A survey of the state-of-the-art," *SN Computer Science*, vol. 1, no. 4, p. 197, 2020. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7289234/