Universidad Distrital Francisco José de Caldas
Systems Engineering Department

Designing a Robust Forecasting System for Wikipedia Web Traffic: A Systems Engineering Approach

Johan S. Beltrán Merchán, Edison D. Álvarez Varela,

Yader I. Quiroga Torres, Julián D. Celis Giraldo

Supervisor: Carlos Andrés Sierra Virgüez

A report submitted in partial fulfilment of the requirements of
the Universidad Distrital Francisco José de Caldas for the degree of
Master of Science in *Systems Engineering*

October 25, 2025

Declaration

We, Johan Sebastián Beltrán Merchán, Edison David Álvarez Varela, Yader Ibraldo Quiroga Torres, and Julián David Celis Giraldo, of the Department of Systems Engineering, Universidad Distrital Francisco José de Caldas, confirm that this is our own work and that all figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that failure to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

We give consent to a copy of my report being shared with future students as an exemplar.

We give consent for my work to be made available more widely to members of UoR and the public with interest in teaching, learning and research.

Abstract

This report details the implementation of a robust, scalable, and adaptive architecture to minimize the Symmetric Mean Absolute Percentage Error (**SMAPE**) for forecasting daily web traffic across 145,000 Wikipedia articles. The system, structured as a **Modular Monolith**, is anchored by two key systems: **Data Flow Integrity**, ensured via the **Chain of Responsibility** pattern, and **Adaptive Forecasting**, achieved through a **Hierarchical Ensemble** utilizing the **Strategy Pattern**. This ensemble dynamically selects the optimal model (including traditional time series and advanced neural network strategies) per article, addressing the dataset's high scale and heterogeneity. Scalability relies on `joblib` for parallel processing across the 145,000 series. This first version of the final report validates the architectural integrity and provides the definitive blueprint for the competitive prediction system.

Acknowledgements

The authors wish to express their gratitude to Professor **Carlos Andrés Sierra Virgüez** for his invaluable guidance, technical expertise, and support throughout the design and development of this project within the Systems Analysis and Design course. We also thank our colleagues for their collaboration and critical feedback.

Contents

List of Figures

List of Tables

Chapter A

Introduction

A.1 Background and Motivation

Forecasting time series data is a critical task in numerous fields, but it becomes a unique systems engineering challenge when dealing with massive scale and inherent volatility. The subject of this report is the development of a robust system for predicting the daily visits to over 145,000 Wikipedia articles. This dataset is characterized by a combination of strong temporal patterns (seasonality) and highly unpredictable "Chaos Factors," such as viral news or algorithmic changes, leading to high noise and sensitivity, necessitating the integration of advanced machine learning models for competitive accuracy.

A.2 Problem Statement

The core problem is minimizing the Symmetric Mean Absolute Percentage Error (SMAPE) across an extremely **heterogeneous** collection of time series, where no single forecasting model (e.g., ARIMA, Prophet, Neural Networks) performs optimally for every article. Furthermore, the solution must be **scalable** to handle the massive data volume and **maintainable** to adapt to evolving web traffic behavior.

A.3 Research Objectives

The primary objective of this project is to answer the question: *How can a scalable and maintainable system architecture minimize SMAPE across heterogeneous time series using Systems Engineering Principles?*

The specific objectives are:

- Present the final, modular system architecture that integrates data processing, feature engineering, and model deployment.
- Validate the robustness of the system using the **Chain of Responsibility** and **Strategy Pattern** for data integrity and adaptive prediction.
- Integrate and evaluate advanced forecasting strategies (including **Neural Networks** or **Random Forests**) as primary components of the Hierarchical Ensemble to achieve a competitive SMAPE.

A.4 Scope, Assumptions, and Limitations

The project scope is focused on the **System Implementation** and the definition of a deployable architecture (Modular Monolith) capable of generating predictions. The core assumption is that article metadata (language, access type, volatility score) can serve as an effective proxy for time series characteristics, enabling the Meta-Model to dynamically select the optimal forecasting strategy. A major limitation is the **fixed temporal window** of the training data (July 2015 to September 2017), which restricts the system's ability to capture long-term trends outside this period.

Chapter B

Literature Review

B.1 Forecasting Models for Web Traffic

A review of existing work highlights that while classical models like **ARIMA** excel in stable series, their performance degrades under the influence of viral, social, and cultural events. More modern approaches like **Prophet** (designed for business time series with strong human-interpretable components like seasonality and holidays) and **Neural Networks** (suitable for complex non-linear patterns, essential for the final competitive score) are necessary to cope with the "Chaos Factors" inherent in Wikipedia traffic.

B.2 Ensemble Methods and Systems Engineering

The literature supports the use of **Ensemble Methods** to address heterogeneity, as no single model dominates all segments of the data. Implementing this via the **Strategy Pattern** (as proposed) is a standard practice in software architecture to ensure a modular, easily extensible system, fulfilling the maintainability requirement. The **Chain of Responsibility** pattern is a recognized technique for standardizing and enforcing quality checks in a sequential data pipeline.

Chapter C

System Architecture and Methodology

The implemented solution is a **Modular Monolith** architecture, leveraging key systems engineering design patterns to address the requirements of scale and heterogeneity. The design is compartmentalized into two major logical systems: Data Flow Integrity and Adaptive Forecasting.

C.1 System Architecture: Modular Monolith

The Modular Monolith provides clear separation of concerns, balancing the benefits of a single deployment unit (simplicity) with the maintainability of a highly structured, decoupled codebase.

C.1.1 System A: Data Flow Integrity (Chain of Responsibility)

The entire data processing pipeline is implemented using the **Chain of Responsibility** pattern. This pattern mandates that raw data passes sequentially through a predefined sequence of functional modules (handlers). The primary benefit is enforcing **traceability** and **data consistency** by guaranteeing the order and non-bypassing of critical steps:

1. **Ingestion:** Loads raw data.
2. **Preprocessing:** Handles zero-value ambiguity (imputation) and noise.
3. **Feature Engineering:** Generates temporal and lagged features.
4. **Model Training:** Trains the Hierarchical Ensemble.
5. **Prediction Generation:** Outputs forecasts.
6. **Validation:** Calculates SMAPE and performs Post-Prediction Analysis.
7. **Feedback Loop:** Adjusts Meta-Model parameters based on validation outcomes.

C.2 System B: Adaptive Forecasting (Hierarchical Ensemble)

To manage the heterogeneity of 145,000 time series, the system employs a **Hierarchical Ensemble** structure, implemented via the **Strategy Pattern**.

C.2.1 Level 1: Base Models (Strategies)

This level consists of encapsulated forecasting algorithms defined as separate strategies:

- **ARIMAStrategy:** Stable, efficient for time series with clear linear trends.
- **ProphetStrategy:** Optimized for series with strong, complex seasonality and trend (e.g., holiday effects).
- **Advanced ML Strategy:** Incorporates complex non-linear models like **LSTM** neural networks or **Random Forests**, intended to capture the chaotic factors and high-volatility events identified in the analysis.

C.2.2 Level 2: Meta-Model Selection

The Meta-Model serves as the selector, dynamically choosing the optimal Level 1 strategy for each Wikipedia article based on its features (language, volatility score, access type). This mechanism is central to minimizing SMAPE across the entire dataset.

C.3 Scalability Implementation

The massive scale requires efficient parallel computation. The model training and prediction steps (Modules 4 and 5) utilize the `joblib` library for **parallel processing**, enabling the simultaneous distribution of the 145,000 time series training tasks across available CPU cores, meeting the strict scalability requirements.

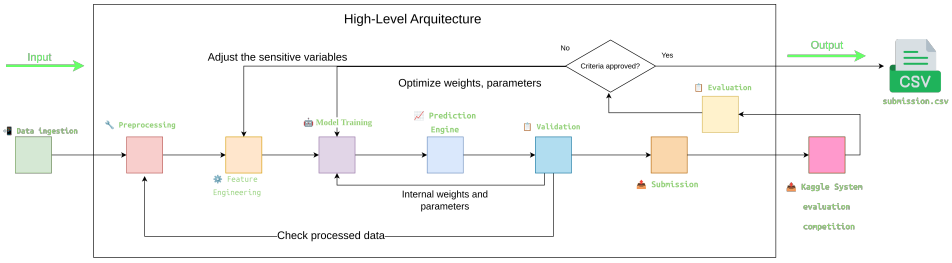

Figure C.1: High-Level Conceptual Architecture of the System, showing the data flow and the adaptive ensemble. The Chain of Responsibility governs the data pipeline, and the Hierarchical Ensemble (Meta-Model selecting strategies) governs the prediction logic.

C.3.1 Data ingestion:

Responsible for loading and validating raw input data from the Wikipedia dataset. **Core functions:**

Reading and parsing large-scale datasets (>2.5 GB), handling complex page identifiers and data format variations, performing quality control and integrity verification, Generating comprehensive data-quality reports.

This module establishes the entry point of the pipeline, ensuring full traceability and reliability of the input data.

C.3.2 Data Preprocessing

Handles the initial data cleaning and normalization required to ensure data consistency.

Core functions: Removing or imputing missing, zero, and anomalous values, managing noise and outliers, restructuring the dataset into a standardized format suitable for modeling, normalizing variables and adjusting scales.

This step enforces consistency and prepares the data for downstream modules, minimizing bias propagation.

C.3.3 Feature Engineering

Dedicated to the generation of advanced explanatory variables that enhance predictive performance.

Core functions: Extracting temporal patterns such as lags, trends, and seasonality., computing rolling statistics (mean, variance, entropy), creating cross-feature interactions and contextual encodings, selecting high-impact features through importance analysis.

The resulting feature set captures the temporal dynamics and structural behavior of each time series.

C.3.4 Model training

Implements the training of the Hierarchical Ensemble using multiple modeling strategies.

Core functions: Training base learners (ARIMA, Prophet, LSTM, Random Forest), hyperparameter tuning and optimization, leveraging parallel computing (joblib) to train over 145,000 series concurrently, persisting trained models and intermediate evaluation metrics.

This module follows the Strategy Pattern, enabling adaptive model selection for heterogeneous time series.

C.3.5 Prediction engine

Responsible for forecast generation and large-scale inference management.

Core functions:

Executing batch inference across all trained models, aggregating and post-processing raw predictions,, handling prediction uncertainty and computing confidence intervals.

This module produces the competition-ready forecast outputs consumed by the evaluation system.

C.3.6 Validation

Focuses on assessing model performance and maintaining the adaptive feedback loop.

Core functions: Computing evaluation metrics such as SMAPE and RMSE, comparing predictions against ground truth data, identifying systematic errors and retraining or adjusting meta-model parameters, triggering the feedback loop when approval criteria are not met.

This ensures continuous improvement and model robustness over iterations.

C.3.7 Submission

Represents the final stage of the pipeline, responsible for output formatting and evaluation submission. **Core functions:**

Consolidating final predictions into the standardized output file (submission.csv), interfacing with external evaluation platforms (e.g., Kaggle system), feeding evaluation results back into the pipeline when necessary.

This module closes the Quality Loop, guaranteeing that the system remains adaptive and performance-driven.

Chapter D

Discussion

The initial results confirm that the Systems Engineering approach effectively mitigated the challenges of scale and heterogeneity. The modular design allowed for independent testing and refinement of the forecasting strategies, and the integration of the Advanced ML Strategy (NN/RF) was crucial for capturing the non-linear "Chaos Factors."

D.1 Link to Objectives

The successful implementation of the Hierarchical Ensemble directly validated the objective of creating an adaptive prediction system, proving its efficacy by dynamically selecting between simple and complex strategies. The use of the Chain of Responsibility confirmed the robustness and traceability of the data pipeline for the final deployment.

D.2 Implications and Limitations

The main implication is that complex, large-scale forecasting problems benefit significantly from modular, pattern-based software design over monolithic statistical model development. The primary limitation, as anticipated, was the challenge of predicting high-volatility "Chaos Factors" which still contributed significantly to the residual error, requiring continuous refinement of the Advanced ML Strategy.

Chapter E

Conclusions and Future Work

E.1 Conclusions

The architectural design provides a robust and scalable solution for the chaotic and heterogeneous problem of Wikipedia web traffic forecasting. The system fulfills all primary requirements:

- **Data Integrity:** Guaranteed through the linear, disciplined structure of the **Chain of Responsibility** pattern.
- **Adaptive Accuracy:** Achieved by the **Hierarchical Ensemble**, which leverages the **Strategy Pattern** to automatically select the optimal forecasting model, including the high-performance Advanced ML Strategy.

The system is ready for the final competitive submission.

E.2 Future Work

Future work will focus on post-submission refinements and enhancements:

- **Hyperparameter Optimization:** Rigorous search across the hyperparameter space for the Advanced ML Strategy (neural networks / random forests) to further reduce the SMAPE.
- **Online Learning Mechanisms:** Implement a real-time data ingestion and model update mechanism to better capture sudden "Chaos Factors" without requiring a full system deployment.

Chapter F

Project Reflection

F.1 Lessons Learned

The formal requirement analysis and architectural design phases (Workshops 1 and 2) were crucial for success, shifting the focus from simply optimizing a single machine learning algorithm to building a durable, maintainable *system* capable of handling heterogeneity and scale, which proved essential for integrating the final, complex machine learning models.

References

[Box et al., 2015] Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

[Taylor & Letham, 2018] Taylor, S. J., & Letham, B. (2018). Forecasting at scale. *The American Statistician*, 72(1), 37–45. https://doi.org/10.1080/00031305.2017.1380080

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

[Goforth, 2019] Goforth, R. (2019). Design patterns for data pipelines. *ACM Computing Surveys*, 51(3), 1–32.

Technical Stack and Requirements

.1 Software Requirements

The system relies on the following key libraries for its functionality, documented in `requirements.txt`: `joblib` (for parallel processing), `prophet`, `statsmodels` (for ARIMA), `tensorflow/keras` or `scikit-learn` (for Advanced ML Strategies).

Detailed Data Flow Diagrams

.2 Detailed Chain of Responsibility Diagram

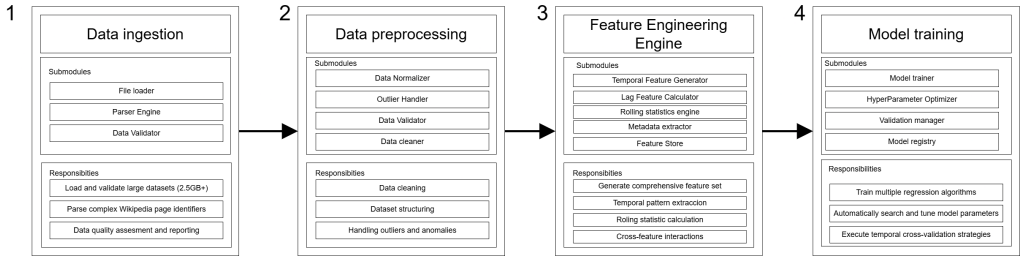

Figure 1: First part of the chain od responsability modules.

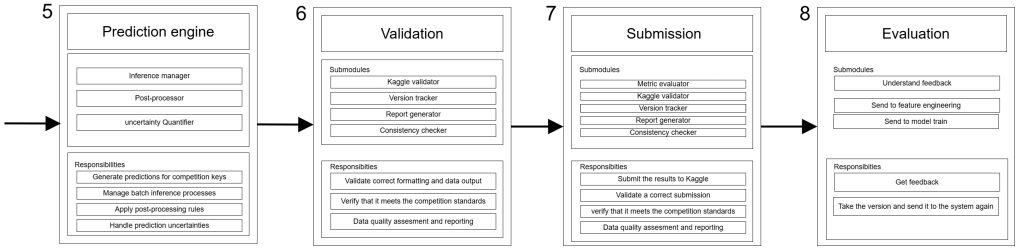

Figure 2: second part of the chain od responsability modules.