Workshop 2: Sweet Regression Competition Analysis

Samuel Aljure Bernal- 20202020111
Carlos Alberto Barriga Gámez-20222020179
David Santiago Aldana Gonzalez-20222020158
Juan Diego Alvarez Cristancho-20221020076

October 2025

1 Review of Workshop #1 Findings

1.1 Summary of Systemic Analysis

The systemic analysis conducted in Workshop #1 provided a detailed understanding of the operational context and analytical scope of the *Sweet Regression Competition*. The objective was to design a predictive framework capable of estimating chocolate sales for *Chocolates 4U* in new regions, based on marketing exposure, regional characteristics, and consumer indicators. The analysis established the boundaries, sensitivities, and potential sources of uncertainty that must guide the next stage of system design.

1.2 Critical Constraints

Several structural and procedural constraints were identified as determinants of system performance:

- **Technical constraint:** All model development must be implemented in `R`, which limits the available modeling libraries but ensures a common analytical standard.
- **Submission limit:** A maximum of ten submissions is permitted, with a single final evaluated file. This increases the risk associated with errors or instability in the final model.
- **Computational complexity:** The training dataset contains 27 variables and 750 observations, producing potentially expensive model-training cycles depending on the chosen algorithm.
- **Relative evaluation metric:** Performance is graded by comparison with peers rather than against an absolute standard, introducing external dependence and limiting the interpretability of scores.

These constraints inform the design priorities of the upcoming system: efficiency, reproducibility, and robustness against limited experimentation opportunities.

1.3 Data Characteristics

Preliminary data exploration revealed the following structural properties:

- The dataset combines quantitative and categorical predictors, including marketing exposure variables (`Web_GRP`, `TV_GRP`, `Facebook_GRP`), socio-economic indices, and environmental attributes (`Weather`, `Tone_of_Ad`, `Coffee_Consumption`).
- No missing values were detected, but feature scaling and categorical encoding will be required to ensure model compatibility.
- Several variables are potentially correlated, which suggests the need for feature selection or regularization to reduce multicollinearity.
- The target variable (`sales`) is continuous and will be evaluated through the Mean Absolute Error (MAE) metric.

These observations imply that the future design must include a structured preprocessing pipeline with correlation analysis, normalization, and encoding procedures defined prior to model training.

1.4 Chaos and Sensitivity Factors

Workshop #1 identified multiple sources of instability and sensitivity consistent with chaos-theory behavior:

- **Data sensitivity:** Small variations in input data (e.g., GRP values or socio-economic indices) can produce large differences in sales estimates.
- **Model sensitivity:** Regression models, especially those based on stochastic algorithms, may yield different outcomes with identical data if random seeds are not fixed.
- **Human variability:** Differences in analytical decisions—such as variable selection or normalization methods—introduce inconsistencies across development attempts.
- **Evaluation dependency:** Since grading is relative, identical models could score differently depending on the competitive environment.

The design phase must therefore include reproducibility mechanisms, fixed random seeds, and internal validation procedures to control for these sensitivities.

1.5 Design Implications

Based on these findings, the following principles will guide the system architecture and design specification for Workshop #2:

1. **Model experimentation framework:** Design the system to support multiple regression techniques (linear, regularized, and ensemble-based) for comparative evaluation.
2. **Overfitting control:** Integrate cross-validation routines and early-stopping mechanisms to maintain generalization between training and test datasets.
3. **Reproducibility and traceability:** Fix random seeds and record experiment configurations to ensure consistent results across runs.
4. **Internal performance benchmarking:** Include validation metrics (MAE) to evaluate performance before final submission.

These implications ensure that the upcoming system design will directly address the sensitivities and structural limitations identified during the first workshop, laying the groundwork for a stable and interpretable predictive modeling environment.

2 System Requirements Definition

2.1 Overview

Building upon the findings from Workshop #1, this section defines the technical and functional requirements that will guide the implementation of the predictive system for *Chocolates 4U*. The requirements aim to ensure that the designed system maintains a balance between predictive accuracy, computational efficiency, and user interpretability, while remaining robust to the constraints and sensitivities previously identified. All specifications are defined as design goals that will later be validated during the implementation phase.

2.2 Functional Requirements

The functional requirements define what the predictive system must accomplish in order to support reliable and reproducible model development.

- **FR1: Data ingestion and validation.** The system shall import structured data files (`.csv`) for both training and testing, verifying schema consistency, data types, and completeness prior to processing.
- **FR2: Preprocessing and feature engineering.** The system shall include routines for normalization, categorical encoding (for variables such as `Tone_of_Ad`, `Weather`, and `Coffee_Consumption`), outlier detection, and correlation analysis to reduce noise and multicollinearity.

- **FR3: Model experimentation.** The system shall support multiple regression modeling techniques (linear, regularized, and ensemble-based) and allow internal comparison through the Mean Absolute Error (MAE) metric on a validation subset.
- **FR4: Validation and overfitting control.** The system shall include early-stopping mechanisms to prevent overfitting and ensure model generalization to unseen data.
- **FR5: Performance tracking and reproducibility.** The system shall record model configurations, random seeds, and validation metrics to ensure reproducibility across experiments.
- **FR6: Prediction generation.** The system shall export predictions in a standardized `.csv` format that matches the submission structure required by the competition.

2.3 Non-Functional Requirements

The non-functional requirements describe the quality attributes of the system, ensuring efficiency, reliability, interpretability, and usability.

- **NFR1: Performance.** Each model training iteration shall complete within 5 minutes on a standard workstation (16 GB RAM, 4-core CPU).
- **NFR2: Reliability.** The pipeline shall execute without interruption in 99% of runs, with automatic handling of missing or unexpected data formats.
- **NFR3: Scalability.** The architecture shall support future datasets of at least twice the current size (1,500+ rows, 50+ variables) without requiring structural modification.
- **NFR4: Interpretability.** The system shall produce feature-importance visualizations and residual plots to assist analysts in understanding how key predictors (e.g., `GRP` variables or socio-economic indices) influence the model output.
- **NFR5: Usability.** All procedures shall be modular and clearly documented, allowing team members to execute the workflow without direct code modification.
- **NFR6: Security.** The system shall process all data locally; no dataset will be uploaded to external servers to maintain confidentiality.

2.4 Measurable Requirements Table

Table 1: Summary of Measurable System Requirements

ID	Requirement Description	Metric / Target	Priority
FR1	Data ingestion and validation	100% schema match, 0 missing values	High
FR2	Preprocessing and feature encoding	Execution time $<$ 60 s per dataset	High
FR3	Model experimentation with MAE metric	At least 3 regression models compared	High
FR4	Early stopping	The model stops if the error (MAE) on the validation set does not improve.	High
FR5	Experiment logging and reproducibility	100% of runs recorded with seed and MAE	High
FR6	Prediction export (CSV)	Conformity with competition format	Medium
NFR1	Performance	Training time $<$ 300 s per iteration	High
NFR2	Reliability	Execution success rate $>$99%	High
NFR4	Interpretability	Feature importance and residual plots included	Medium
NFR5	Usability	90% documentation coverage of modules	Medium

2.5 User-Centric Considerations

The system is designed not only for computational efficiency but also for usability by analysts and students participating in the competition.

- **Ease of Use:** Provide modular scripts and an execution guide outlining each stage of the pipeline.
- **Transparency:** Produce interpretable outputs such as scatter plots and feature-importance graphs to justify predictions.
- **Error Feedback:** Integrate clear error messages and validation checks for incorrect data formats or missing variables.

3 High-Level Architecture

3.1 Overview

The predictive system for *Chocolates 4U* will be designed following a modular architecture that organizes the entire analytical workflow into independent yet connected components. This architecture ensures data integrity, reproducibility, and interpretability throughout the process, from data ingestion to prediction generation. Each module is designed to align with systems engineering principles such as modularity, traceability, and feedback control, which are essential to mitigate the sensitivities and chaotic behavior identified in Workshop #1.

3.2 Architecture Diagram

The following figure illustrates the proposed high-level system architecture, representing the logical flow of data and information between modules. The design emphasizes iterative feedback loops between data processing, validation, and modeling stages to prevent overfitting and enhance prediction reliability.

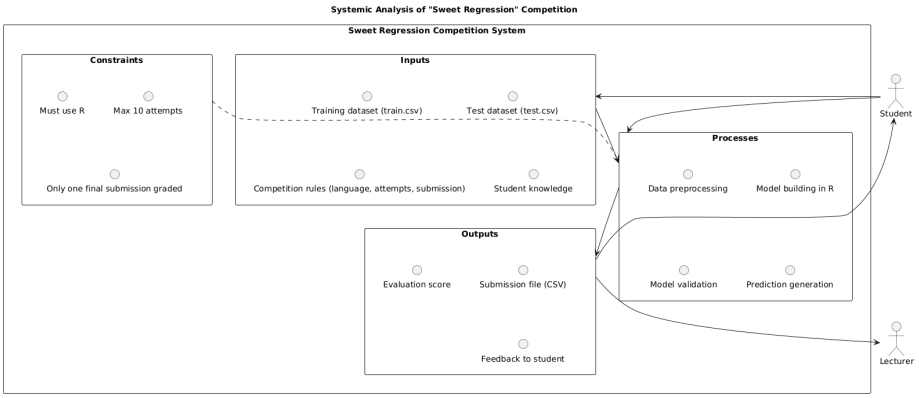
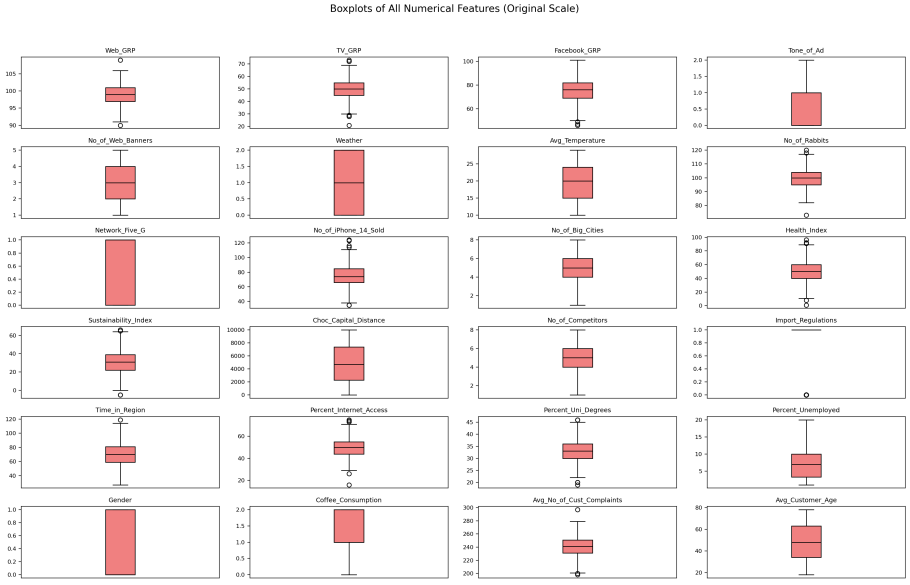


Figure 1: High-Level Architecture of the Chocolates 4U Predictive System

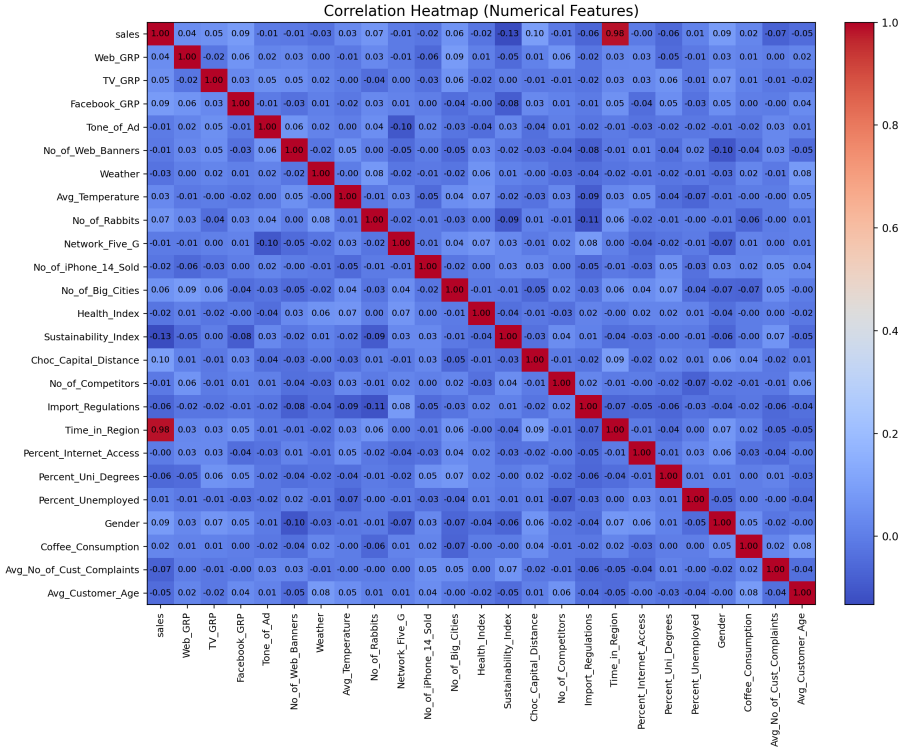
3.3 Module Descriptions and Responsibilities

1. **Data Ingestion Module** This component is responsible for importing the training and test datasets in `.csv` format. It verifies the structure, column types, and absence of missing values. Any schema inconsistency triggers validation alerts before proceeding.

2. **Preprocessing and Transformation Module** This module manages data cleaning, encoding of categorical variables (`Tone_of_Ad`, `Weather`, `Coffee_Consumption`), normalization of numerical attributes (e.g., GRPs, temperature), and outlier detection. The output is a standardized dataset ready for feature analysis.



3. **Feature Analysis and Selection Module** Performs correlation analysis and statistical tests (e.g., ANOVA) to identify the most influential predictors of `sales`. These findings determine the subset of variables that will be passed to the modeling engine.



4. **Modeling Engine** The central analytical component, designed to compare multiple regression approaches (linear, regularized, and ensemble-based). Each model's performance will be assessed using the Mean Absolute Error (MAE) metric. The engine's modular structure allows easy extension with new algorithms without altering other components.

5. **Validation and Control Unit** Integrates cross-validation and early-stopping mechanisms to ensure that models generalize well to unseen data. This module prevents overfitting by monitoring MAE.

6. **Output and Export Module** Generates structured outputs, including the final `.csv` file for submission and summary visualizations (e.g., feature importance, residual plots). This component also produces automated reports summarizing performance metrics, supporting interpretability and traceability.

3.4 System Flow and Interactions

The modules interact through a sequential yet iterative flow:

1. Raw data are loaded and validated by the **Data Ingestion Module**.

2. The **Preprocessing Module** transforms inputs into a clean, numerical dataset.
3. The **Feature Analysis Module** identifies relevant predictors and passes them to the **Modeling Engine**.
4. The **Modeling Engine** trains several candidate regression models and reports their MAE results to the **Validation Unit**.
5. Based on cross-validation outcomes, the **Validation Unit** selects the most generalizable configuration.
6. Finally, the **Output Module** generates prediction files and visual summaries for documentation and analysis.

3.5 Systems Engineering Principles Applied

The design adheres to core systems engineering principles to ensure robustness, maintainability, and transparency:

- **Modularity:** Each process (data ingestion, preprocessing, modeling, validation, export) operates independently, facilitating debugging and scalability.
- **Traceability:** Each dataset transformation and model iteration will be logged, ensuring full traceability from raw input to final prediction.
- **Feedback Control:** Validation metrics (MAE) feed back into the modeling process, forming a closed-loop control that minimizes error and overfitting.
- **Scalability:** The modular structure allows new data or algorithms to be integrated without altering the existing workflow.
- **Observability:** Intermediate outputs (e.g., correlations, validation plots) provide visibility into the system's internal processes and support informed adjustments.

4 Addressing Sensitivity and Chaos

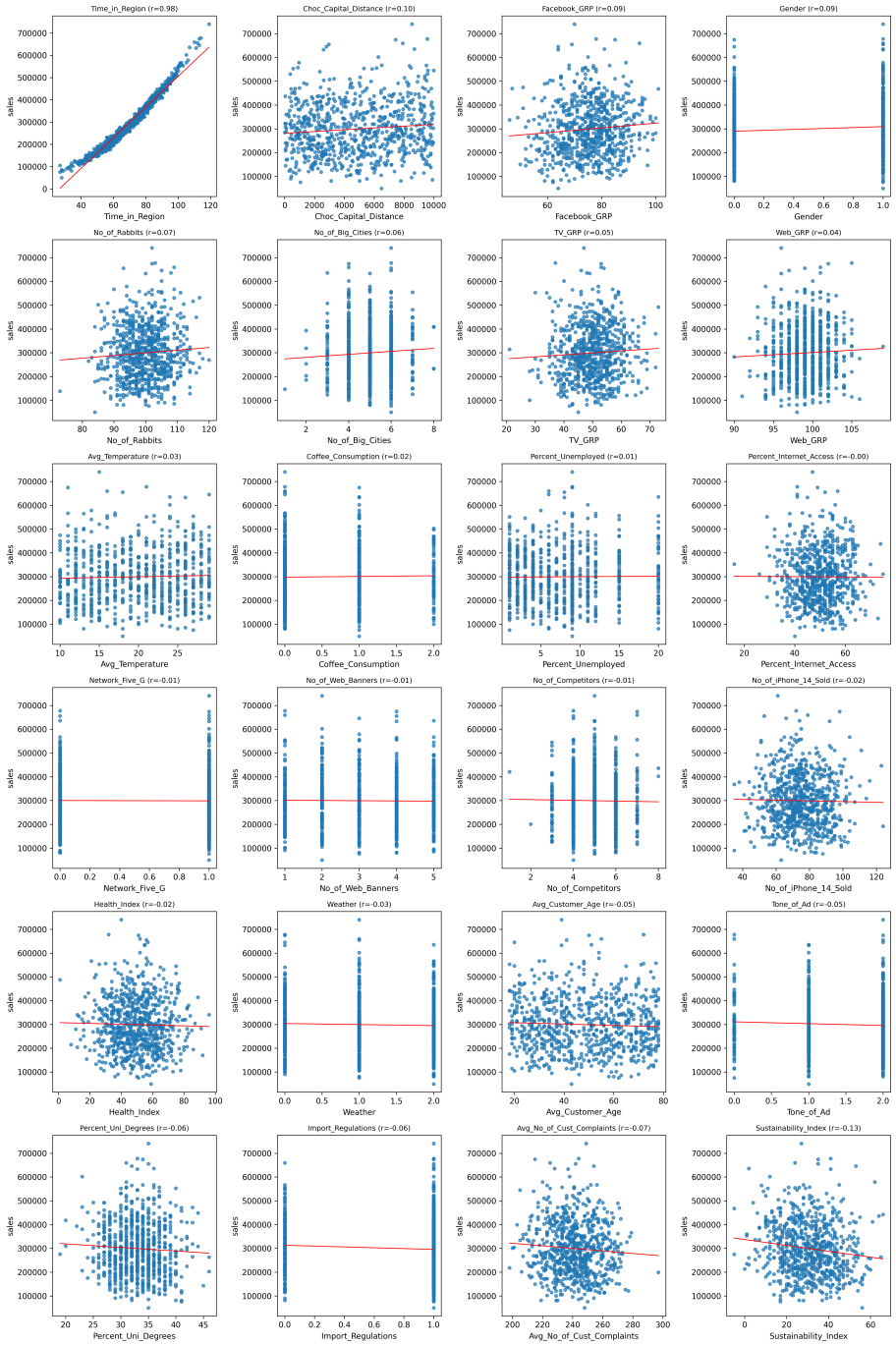
4.1 Overview

Given the stochastic and nonlinear nature of real-world data, the predictive system for *Chocolates 4U* must incorporate mechanisms to control high-sensitivity behaviors and mitigate chaotic responses to small input variations. The analytical context described in Workshop #1 identified several elements of instability—both in the data and the modeling process—that could affect the reliability of predictions. This section outlines the control strategies, monitoring routines, and design safeguards that will be integrated into the system to ensure stability, reproducibility, and consistent performance.

4.2 Sources of Sensitivity and Chaotic Behavior

During the analysis phase, several factors were identified that could make the predictive system sensitive or unstable if not properly controlled. These elements can cause the model to react strongly to small changes in the data or in its configuration. The main sources are the following:

- **Data fluctuations:** Small variations in marketing indicators such as `TV_GRP` or `Facebook_GRP` can lead to large differences in predicted sales. This means that even minor input changes may have a strong impact on the model's output.
- **Correlation between variables:** Some predictors are closely related to each other, which can confuse linear models and make them unstable. For example, if two advertising variables increase together, the model might struggle to decide which one truly influences sales.



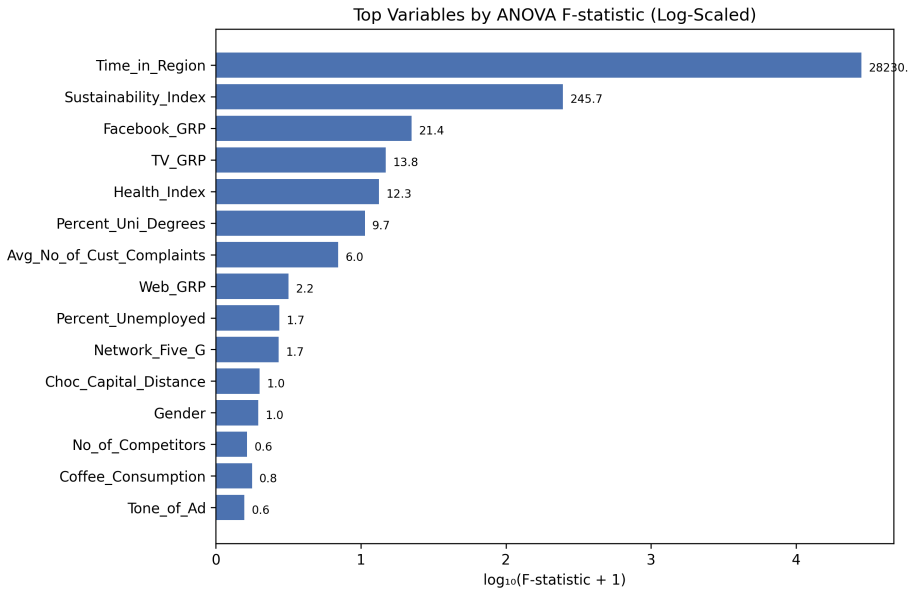
- **Randomness in model training:** Algorithms such as Random Forest or

Gradient Boosting include random elements (like random selection of data or features). Without proper control, these random factors can produce slightly different results each time the model is trained.

- **Differences in data preparation:** The way data are preprocessed—such as how variables are scaled or encoded—can affect model results. Different preprocessing decisions applied to the same dataset can lead to different outcomes.
- **Evaluation sensitivity:** Because model performance in the competition is compared to other teams' results, even small differences in the Mean Absolute Error (MAE) can change the relative ranking. This increases the importance of maintaining internal consistency and precision throughout the process.

4.3 Design Mechanisms for Sensitivity Control

The system architecture integrates multiple design mechanisms specifically intended to reduce or neutralize chaotic effects during model development and prediction generation:



1. **Fixed random seeds:** All stochastic processes (e.g., data splits, model initialization) will be executed with controlled random seeds to ensure reproducibility across runs.
2. **Regularization and feature selection:** The modeling engine will incorporate Lasso or Ridge penalties, as well as correlation-based feature

reduction, to minimize the amplification of noise due to redundant variables.

3. **Standardization of preprocessing:** A consistent preprocessing pipeline will normalize feature scales, encode categorical values uniformly, and detect outliers before modeling, thus minimizing chaotic variations between runs.

4. **Error bounding via validation thresholds:** When the validation set does not improve. The model stops, the model will be flagged as overfitted and retraining will be triggered.

5. **Outlier and anomaly detection:** Visual and statistical detection methods (e.g., interquartile range, residual analysis) will identify and, if necessary, exclude outliers that may disproportionately distort regression coefficients.

4.4 Monitoring and Error-Handling Routines

To manage unexpected variations or errors during execution, the system design includes monitoring and control routines that help detect problems early and maintain reliability throughout the process:

- **Data quality monitoring:** Before each execution, the ingestion module will check that the dataset matches the expected structure and value ranges. If inconsistencies or missing variables are detected, the system will stop and generate a warning report.

- **Training validation tracking:** During model training, MAE values will be recorded at each iteration or fold. This allows the team to identify when the model stops improving and to decide manually whether to interrupt training or adjust parameters.

- **Error logging:** Any execution errors—such as missing files, incorrect data types, or model convergence issues—will be automatically logged with the corresponding date, time, and configuration details. These logs will assist in identifying and correcting the source of problems.

- **Model rollback:** If instability or unexpected results occur, the team can revert to the last validated version of the model to ensure consistent performance.

- **Final verification:** Before exporting predictions, a residual analysis will be performed to confirm that the distribution of errors remains within an acceptable range. This helps detect potential overfitting or anomalies in the results before submission.

4.5 Feedback and Adaptation Mechanisms

Although the system is not self-adjusting, it is designed to provide clear feedback that guides manual decision-making during model refinement. The goal is to help analysts interpret performance results and decide which aspects of the process need adjustment.

- Cross-validation outcomes will help determine whether the current regularization level or selected features are appropriate, or if changes are needed to reduce error variability.
- Residual plots and error distributions will guide the review of preprocessing steps, such as data scaling or outlier handling.
- Validation metrics (especially MAE) will inform the comparison between different regression models, allowing the selection of the most stable and generalizable configuration.

In this way, the feedback provided by the system supports an iterative improvement process led by the analysts. Rather than making automatic corrections, the system delivers structured evidence that allows the team to make informed, data-driven decisions.

5 Technical Stack and Implementation Sketch

5.1 Overview

The implementation of the predictive system for *Chocolates 4U* will be based on a modular and reproducible software stack that ensures transparency, efficiency, and extensibility. Although the competition guidelines specify the use of `R` for model development and submission, the design strategically incorporates `Python` for the analytical and exploratory stages of the process. Python offers greater flexibility and ease of use for data visualization, statistical exploration, and team collaboration. This division of responsibilities allows the project to leverage the strengths of both environments: the analytical power and statistical rigor of `R`, combined with the visualization and preprocessing capabilities of `Python`.

5.2 Programming Languages and Core Libraries

- **Python (Analytical and Exploratory Environment):** Python will be used during the initial analytical phase, including data exploration, visualization, and statistical testing (such as ANOVA and correlation analysis). Its wide range of libraries facilitates interactive exploration and efficient feature analysis. The following packages will be employed:
  - `pandas`, `numpy` – data manipulation and numeric computation.
  - `matplotlib`, `seaborn` – visualization of trends, correlations, and feature distributions.

  - `scipy.stats` – statistical tests and ANOVA computations.
  - `scikit-learn` – basic regression testing, feature scaling, and validation utilities.
  - `xgboost` – exploratory boosting experiments to assess variable sensitivity.

  This stage will produce cleaned and preprocessed datasets, as well as graphical summaries exported in image format for documentation.

- **R (Modeling and Validation Environment):** R will be used for the main modeling phase and competition submission, following the official requirements. It offers strong packages for regression, validation, and result interpretation. The following libraries will be used:
  - `tidyverse` – data manipulation and integration.
  - `caret` – model training and cross-validation.
  - `glmnet` – regularized regression (Ridge, Lasso, ElasticNet).
  - `xgboost` – gradient boosting regression with early-stopping capabilities.
  - `ggplot2` – final visualizations and residual analysis.

5.3 Development Environment

The system will be developed within a version-controlled environment (`GitHub`) using the following workflow:

- **Repository Structure:** Each stage of the workflow will reside in a dedicated folder.
- **Version Control:** Git commits will be used to track modifications in preprocessing routines, model configurations, and output artifacts.

5.4 Implementation Plan

The following subsections describe the planned implementation steps for each architectural module introduced in Section 3.

1. **Data Ingestion Module** Implemented as a scripted function that reads the training and test datasets from the local `/data` directory. It verifies column consistency, reports missing values, and produces a validation summary file (`data_report.txt`).
2. **Preprocessing and Transformation Module** Implemented in Python as a sequence of pipeline functions that handle categorical encoding, normalization, and outlier filtering. All transformations will be logged and versioned to ensure reproducibility and alignment with the later R-based modeling phase.

3. **Feature Analysis Module** Conducted mainly in Python, this stage will perform correlation and ANOVA analyses to evaluate predictor significance. Visualizations (such as heatmaps and scatter plots) will be exported and stored in the `/results/analysis` folder for documentation and reference.

4. **Modeling Engine** Configured in R using the `caret` and `xgboost` packages. This module will train multiple regression models (Linear, Ridge, Lasso, Random Forest, XGBoost) under identical preprocessing parameters. MAE will be used as the main metric, and results will be summarized in `model_performance.csv`.

5. **Validation and Control Module** Implements early-stopping routines for iterative algorithms and compares training versus validation MAE values to detect overfitting. When excessive performance divergence is detected, manual parameter adjustments will be considered.

6. **Output and Reporting Module** Generates the final prediction file in `.csv` format for submission. Produces summary plots (MAE distribution, residual plots, feature importance) and compiles them into a report stored in `/results/final/`.

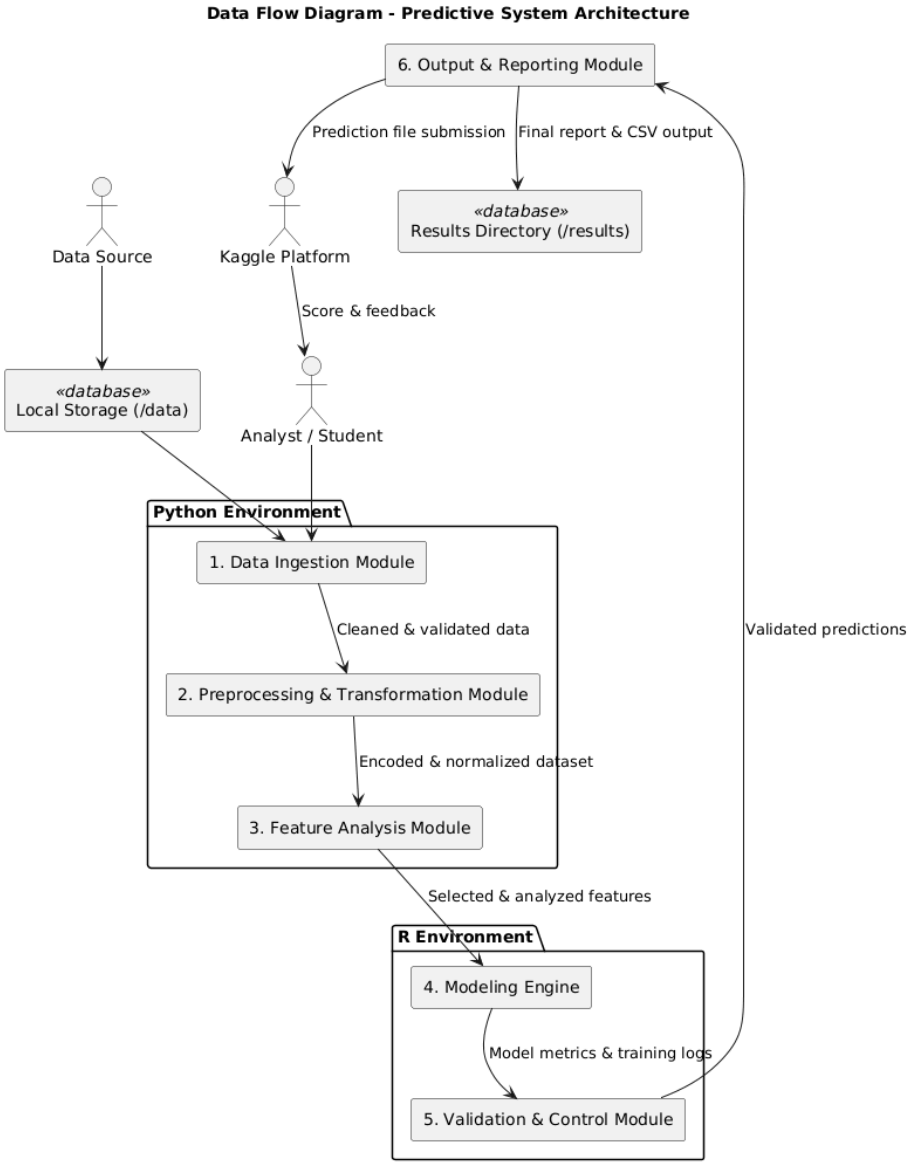


Figure 2: Flow Diagram of the Chocolates 4U Predictive System

5.5 Integration and Design Patterns

The implementation will follow several core software design principles and patterns:

- **Pipeline Pattern:** Each module executes as a stage in a sequential data

pipeline where the output of one stage becomes the input of the next. This ensures data traceability and simplifies debugging.

- **Separation of Concerns:** Analytical work (Python) and modeling work (R) are separated into independent environments, improving maintainability and team coordination.
- **Manual Feedback Loop:** The validation module will provide feedback on MAE performance and generalization, allowing the analysts to manually decide if model or preprocessing adjustments are needed.

5.6 Integration Workflow

The system's integration plan follows a linear–iterative flow:

1. Perform exploratory analysis and feature evaluation in Python, including statistical tests and visualizations.
2. Export the cleaned and prepared dataset to be used in R.
3. Implement and test the modeling engine with cross-validation and early-stopping mechanisms.
4. Validate pipeline stability by running full workflows on the training dataset.
5. Connect output generation scripts to produce and verify the final prediction file format.
6. Document environment configurations, model parameters, and reproducibility settings.