UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Universidad Distrital Francisco Jose de Caldas
Faculty of Systems Engineering

Otto Group Product Classification Challenge, Technical Report

Juan Pablo Mosquera Marin
Jeison Felipe Cuenca
Maria Alejandra Ortiz Sanchez
Juan Diego Lozada

Professor: Carlos Andres Sierra Virgüez

A report submitted in fulfilment of the requirements of
Universidad Distrital Francisco Jose de Caldas for the subject of
Universidad Distrital Francisco Jose de Caldas in *Systems Analysis & Design*

December 12, 2025

Abstract

This technical report presents the initial development of a modular machine learning system for the Otto Group Product Classification Challenge. The project aims to classify large-scale anonymized e-commerce data using Python tools such as `NumPy`, `Pandas`, and `Scikit-learn`. A system-oriented design was applied to ensure adaptability, feedback control, and scalability. The report outlines the architecture, design decisions, and implementation plan defined in the first workshops. Early results indicate that the modular structure improves stability, traceability, and flexibility for future optimization.

Keywords: E-commerce, Classification, unsupervised AI, System, Feedback

Acknowledgements

We would like to express our appreciation to our teacher, Carlos Andres Sierra Virgüez, for his continuous guidance and support during the development of this project. His insights and the learning tools provided have been essential in shaping the foundation of this first stage and guiding us toward the next phases of implementation.

Contents

List of Figures

List of Tables

Glossary

Architecture	The structural design of the system, defining how components interact and communicate.
Dataset	A structured collection of data used for model training and testing in the Otto classification system.
Feature Engineering	The process of transforming raw data into meaningful variables that improve model performance.
Feedback Loop	A mechanism in which system outputs are used as inputs to regulate performance and stability.
Normalization	A data preprocessing technique that adjusts numerical values to a common scale.
Overfitting	A modeling issue where a machine learning model performs well on training data but poorly on unseen data.
Pipeline	A sequential workflow that automates data preprocessing, model training, and evaluation steps.
Prototype	The initial functional version of the system developed to validate design assumptions and methods.
Scalability	The ability of a system to handle increased data volume or complexity without loss of performance.
Sensitivity Analysis	A process to evaluate how variations in input parameters affect system outputs or model behavior.

Chapter 1

Introduction

The increasing complexity of e-commerce operations has made product classification a critical challenge for organizations managing extensive catalogs. The Otto Group, one of the world's leading online retail conglomerates, faces the task of consistently categorizing thousands of products described only by numerical attributes without clear semantic meaning. Misclassification leads to inefficiencies in product recommendation, inventory management, and business decision-making.

This technical report documents the initial phase of a project aimed at designing and implementing a data-driven classification system for the Otto Group Product Classification Challenge hosted on the Kaggle platform. The dataset includes over 200,000 product entries and 93 anonymized numerical features that require advanced preprocessing, feature engineering, and model calibration. The problem's main difficulty lies in the high dimensionality and the absence of descriptive metadata, which increase sensitivity and the potential for instability during training.

The project applies a systems engineering perspective to the design of the classification pipeline. Rather than focusing solely on machine learning performance metrics, the approach emphasizes structural integrity, modularity, and adaptability. This aligns with principles of feedback control and dynamic stability, allowing the system to adjust automatically when variations or errors are detected during data processing or model training.

The specific objectives of this phase are:

- To analyze the problem from a systemic point of view, identifying constraints, components, and interactions within the data and model workflow.
- To design a modular architecture that separates data processing, feature transformation, classification, and reporting into independent but coordinated subsystems.
- To define an implementation plan using open-source Python tools and establish initial feedback loops for monitoring model performance and sensitivity.

This report provides the foundation for the subsequent implementation stage, where the modules will be coded, validated, and integrated into a complete classification framework. It sets the technical direction for building a resilient and interpretable system capable of scaling to large datasets while maintaining consistency and control over its dynamic behavior.

Chapter 2

Literature Review

The Otto Group Product Classification Challenge on Kaggle represents a complex multiclass classification problem that has been widely discussed within the data science community. The first place solution developed by Gilberto Titericz and Stanislav Semenov introduced a three-layer stacked learning architecture, integrating diverse models in hierarchical levels to improve generalization and reduce overfitting Titericz and Semenov (2015). The approach used approximately 33 first-level models whose predictions were fed as meta-features into a second level, followed by a weighted ensemble at the final stage demonstrating the effectiveness of multi-level ensembling for high-dimensional structured data.

Stacked generalization, first formalized by Wolpert Wolpert (1992), is a technique that combines multiple learning algorithms by training a meta-model on the predictions of base models. This strategy has been extensively used in machine learning competitions and research due to its ability to integrate heterogeneous learners and capture complementary decision boundaries. For multiclass problems, stacking enables flexible decision aggregation across diverse model families such as decision trees, neural networks, and linear classifiers.

Among ensemble methods, gradient boosting algorithms have become a dominant approach for structured data. The XGBoost framework proposed by Chen and Guestrin Chen and Guestrin (2016) introduced a scalable and regularized boosting technique that efficiently handles missing values, supports parallel computation, and minimizes overfitting through advanced regularization. Its combination of performance, interpretability, and computational efficiency has made it a preferred model in competitions such as Otto, where training speed and stability are critical constraints.

Another essential aspect in multiclass classification tasks is the calibration of predicted probabilities. When evaluation metrics such as multi-class logarithmic loss are used, models must output accurate probability distributions rather than only correct labels. Platt Platt (1999) introduced a parametric approach to calibrating classifier outputs using a sigmoid function, while Niculescu-Mizil and Caruana Niculescu-Mizil and Caruana (2005) compared different calibration methods and demonstrated their importance for improving predictive reliability in supervised learning. These works provide the theoretical basis for post-processing steps that align predicted confidence scores with actual observed frequencies.

In summary, previous studies and competition results emphasize three main insights relevant to this project: (i) stacked ensembles effectively combine diverse models to achieve robust multiclass classification; (ii) gradient boosting, particularly XGBoost, provides scalable and regularized performance for large structured datasets; and (iii) probability calibration techniques are essential when evaluation metrics depend on the quality of probabilistic outputs. However, while these methods are well-studied in algorithmic contexts, fewer works explore

their integration from a systems-engineering perspective—addressing issues such as component traceability, systemic feedback, and stability management. This technical report aims to bridge that gap by embedding ensemble learning strategies into a modular, feedback-oriented system architecture.

Chapter 3

Background

Understanding the Otto Group Product Classification project requires not only knowledge of machine learning but also a solid foundation in systems analysis and design. The project is approached as a complex information system composed of interrelated components and data sources, processing modules, analytical engines, and feedback mechanisms that must operate cohesively to achieve reliable classification performance.

From a systems engineering perspective, analysis and design represent two fundamental phases of the development life cycle. During the *analysis phase*, the problem domain is studied to identify requirements, data characteristics, constraints, and environmental factors affecting system behavior. This includes defining the inputs, outputs, and interactions between components, as well as understanding how uncertainty and sensitivity influence outcomes. In this project, the system analysis conducted in the project revealed several critical challenges: high dimensional numerical data, lack of semantic labeling, class imbalance, and performance sensitivity due to probabilistic evaluation metrics. These findings guided the definition of the main architectural constraints and design principles.

The *design phase*, developed, focused on translating analytical insights into a structured system architecture. System design involves defining logical and physical components, data flow, and feedback control mechanisms. The proposed architecture, follows a modular and layered structure composed of four main subsystems: *Data Processing*, *Feature Engineering*, *Classification Engine*, and *Analytics and Reporting*. Each subsystem performs a distinct role but interacts through defined data interfaces and control signals, ensuring modularity and scalability.

Key theoretical foundations applied in this work stem from systems theory and control principles. According to systems thinking, complex systems exhibit dynamic interdependencies and feedback loops that must be understood to maintain equilibrium and adaptability. Feedback both positive and negative serves as a regulatory mechanism that stabilizes performance and prevents instability caused by noise, parameter fluctuations, or overfitting. In the context of machine learning, this concept is analogous to iterative training and validation cycles, where model adjustments are informed by performance monitoring and error propagation analysis.

Another relevant concept is *modularity*, which promotes decomposition of the system into manageable and reusable components. Modularity not only improves maintenance and traceability but also supports experimentation, as each subsystem can be developed, tested, or replaced independently without compromising global coherence. This principle is essential in handling the complexity of the Otto dataset and facilitating the integration of diverse algorithms, preprocessing strategies, and evaluation methods.

The background analysis also considers sensitivity and chaos related factors. Small varia-

tions in preprocessing methods, feature scaling, or random initialization can significantly alter classification results a phenomenon consistent with sensitivity to initial conditions described in dynamic systems theory. Addressing this behavior requires robust cross validation, probabilistic calibration, and continuous feedback monitoring, which are incorporated into the system's analytical and reporting components.

In summary, the background of this project lies at the intersection of systems analysis, software architecture, and data science. By integrating systemic thinking into the design of a machine learning framework, the project aims to achieve not only high predictive accuracy but also structural stability, interpretability, and adaptability. These principles form the conceptual foundation upon which the technical methods, design decisions, and future implementation stages are built.

Chapter 4

Objectives

The main objective of this project is to design and develop a modular, feedback oriented machine learning system capable of performing reliable multiclass classification on the Otto Group Product dataset. The project seeks to integrate principles from systems analysis and design with data driven modeling techniques to achieve a balance between computational performance, interpretability, and systemic stability.

General Objective

To conceptualize and implement a system architecture that applies systems thinking principles such as modularity, feedback control, and adaptability to the development of a classification framework using Python based machine learning tools.

Specific Objectives

- **System Analysis:** Identify the critical constraints, data characteristics, and sensitivity factors affecting system performance through detailed analysis of the Otto dataset.
- **System Design:** Define a modular architecture composed of independent but interoperable subsystems, *Data Processing*, *Feature Engineering*, *Classification Engine*, and *Analytics and Reporting*, based on the outcomes of the analisys and design
- **Implementation Planning:** Establish a clear plan for prototype development using Python libraries such as `NumPy`, `Pandas`, and `Scikit-learn`)and prepare the environment for later integration of additional models or data sources.
- **Documentation and Traceability:** Produce technical documentation that supports traceability of design decisions, analytical results, and experimental procedures for future iterations of the system.

In summary, the project aims to produce not only an accurate classifier for the Otto dataset but also a structured and transparent system that embodies the principles of systems analysis and engineering design. This alignment ensures that future development phases can expand the system's capabilities while maintaining control, stability, and methodological coherence.

Chapter 5

Scope

This technical report defines the scope of the project as the analysis and design of a modular, feedback oriented system for product classification using the Otto Group dataset. The work focuses on the conceptual and structural aspects of system development rather than on the exhaustive exploration of advanced machine learning algorithms or optimization techniques.

The scope includes the following components:

- **System Analysis:** Identification of key system constraints, data characteristics, and operational requirements derived from the Otto dataset. This involves studying input–output relationships, critical variables, and potential sources of instability or noise.
- **System Design:** Development of a structured architecture that defines the interaction between subsystems such as *Data Processing*, *Feature Engineering*, *Classification Engine*, and *Analytics and Reporting*. The design emphasizes modularity, scalability, and feedback control principles.
- **Technical Planning:** Specification of implementation tools and resources, including the use of Python libraries to build a functional prototype aligned with systems engineering practices.
- **Evaluation Framework:** Establishment of performance monitoring and feedback mechanisms for assessing system stability, reproducibility, and interpretability during preliminary testing.

The project deliberately excludes topics that fall outside the scope of system analysis and design, such as the in-depth mathematical study of deep learning architectures, hyperparameter optimization, or advanced ensemble tuning techniques. These aspects are acknowledged as potential future extensions but are not the primary focus of this initial development stage.

Therefore, the boundaries of this work are defined by its emphasis on applying systems thinking and design methodologies to a data-driven classification task. The report aims to deliver a well-documented conceptual framework and technical foundation that can later support further implementation, experimentation, and optimization.

Chapter 6

Assumptions

6.1 Assumptions

The development of this project is based on a set of assumptions that define the conditions and context under which the system analysis and design were performed.

- **Data Integrity:** It is assumed that the Otto Group dataset is complete, consistent, and representative of real world product classifications. Any missing or inconsistent records are managed through preprocessing rather than through modifications of the dataset itself.
- **Static Dataset:** The dataset is considered static during the development phase. No new data entries, feature modifications, or dynamic updates are introduced while analyzing or designing the system.
- **Feature Independence:** Since the features in the dataset are anonymized and lack semantic meaning, it is assumed that feature relationships can be treated statistically rather than contextually. All dependencies and correlations are inferred through exploratory analysis.
- **Computational Environment:** It is assumed that the Python development environment provides sufficient computational resources to execute preprocessing, training, and evaluation tasks for the dataset.

6.2 Limitations

As this project constitutes a partial fulfillment of a university course on Systems Analysis and Design, several practical and contextual limitations influence its scope, depth, and outcomes. These limitations must be considered when interpreting the results and assessing the progress achieved in this preliminary phase.

- **Academic Context and Time Constraints:** The project is developed within the timeframe of an academic semester. Consequently, the analysis, design, and prototype development are bounded by specific deadlines, limiting the extent of experimentation, and iterative validation that could otherwise enhance model performance.
- **Experience Level:** The research team is in a learning process regarding advanced data science and machine learning methodologies. While the system design incorporates engineering principles, the lack of professional experience in model optimization, feature

selection, and hyperparameter calibration restricts the ability to achieve the best possible predictive performance.

- **Data Constraints:** The dataset provided by the Otto Group competition is anonymized and lacks contextual meaning. This restricts the possibility of incorporating domain knowledge or semantic analysis in feature engineering, forcing the system to depend entirely on statistical correlations and numerical transformations.
- **Methodological Scope:** The focus of this report is on the analysis and design of the system, rather than the exhaustive application of machine learning techniques. Deep learning architectures, automated feature learning, or large ensemble methods are recognized as valuable extensions but fall outside the objectives of this first technical phase.
- **Evaluation and Validation:** Due to the early development stage, the evaluation framework primarily considers conceptual validation of the architecture and preliminary performance checks. Full scale quantitative validation will be addressed in future iterations once the prototype is fully implemented.

Chapter 7

Methodology

7.1 Methodology

This section describes the methodological framework used for the analysis, design, and preliminary development of the Otto Group Product Classification System. The methodology follows the principles of systems engineering, combining structured analysis and modular design with data-driven experimentation. Although this report corresponds to an early project phase, the methodological structure ensures reproducibility, scalability, and traceability for future iterations.

7.1.1 Research Approach

The research approach integrates concepts from systems analysis and software engineering within a machine learning context. The process was divided into two main stages—*System Analysis* and *System Design*.

The *analysis phase* focused on understanding the problem, dataset characteristics, and system boundaries. This included identifying relevant variables, dependencies, and sources of instability. The *design phase* translated these insights into a modular system architecture capable of handling data processing, feature transformation, classification, and feedback control.

The overall workflow is illustrated in Figure 7.1, which presents the logical structure of the system. Each module interacts through defined data interfaces, ensuring modularity and facilitating future expansion or replacement of components.

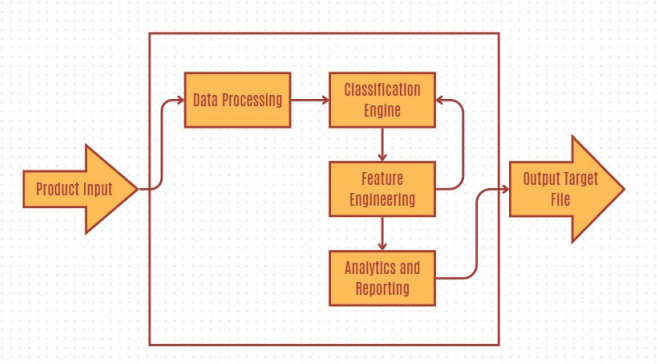

Figure 7.1: General architecture of the Otto Group Classification System.

7.1.2 Data Collection and Preparation

The dataset used in this project was obtained from the Kaggle competition *Otto Group Product Classification Challenge*. It consists of more than 200,000 product entries and 93 anonymized numerical features. The data were provided in CSV format and divided into training and test subsets.

No additional data sources were incorporated, ensuring reproducibility and compliance with competition guidelines. The dataset was treated as static throughout this phase. Data preparation followed the steps:

- **Normalization and Scaling:** All numerical features were standardized using Z-score scaling to ensure that each variable contributed proportionally to the learning process.
- **Exploratory Analysis:** Correlation structures were studied to identify dependencies between features and potential redundancies, as illustrated in Figure 7.2.

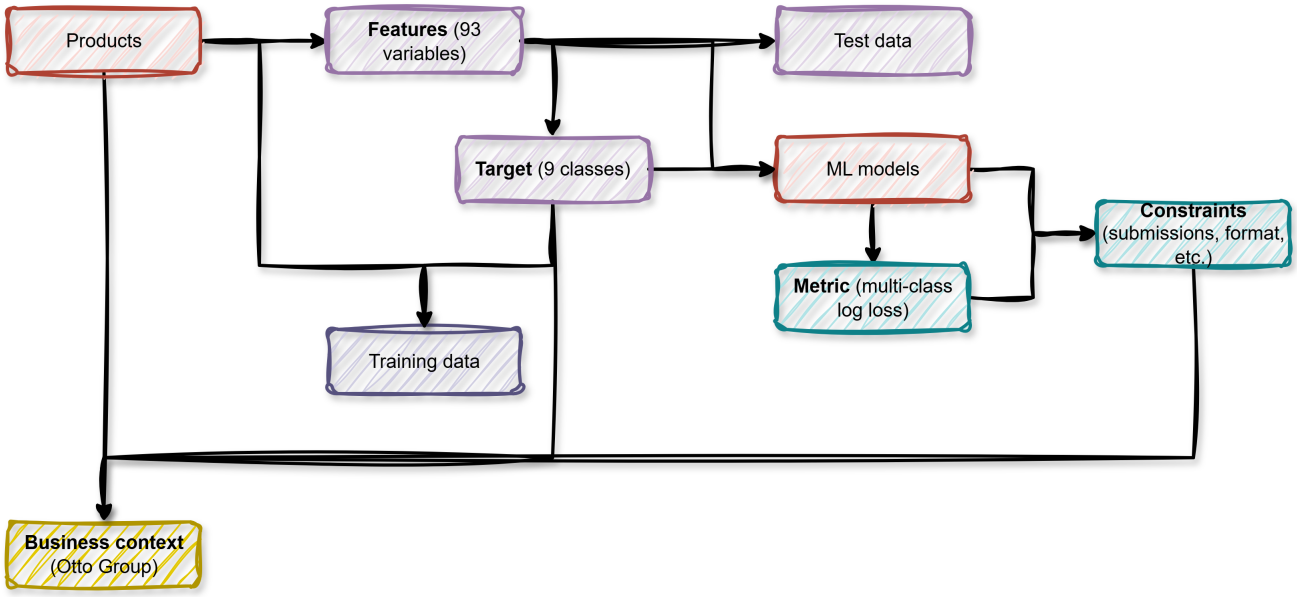

Figure 7.2: Feature correlation map generated during the analysis phase (Workshop #1).

7.1.3 System Design and Development Process

Based on the analytical findings, the system was originally designed as a modular pipeline with four core subsystems:

- **Data Processing:** Handles data cleaning, validation, and formatting.
- **Feature Engineering:** Applies statistical transformations and dimensionality reduction.
- **Classification Engine:** Encapsulates classical machine learning models such as logistic regression and random forest.
- **Analytics and Reporting:** Generates metrics and visual summaries for feedback.

This early structure was later expanded into a fully operational implementation with executable modules, standardized interfaces, and a reproducible workflow. The final system follows a programmatic pipeline executed through a command-line interface (CLI), integrating advanced data validation and experiment management.

7.1.4 Full System Implementation

The complete implementation consists of four main modules, each placed within the `src/` directory as independent and reusable components:

1. **Data Loader (`data_loader.py`):** Loads and validates the input CSV files, checks schema consistency, handles missing values, and extracts numerical matrices suitable for modeling. It ensures reproducibility by fixing seeds and producing deterministic dataset splits.
2. **Feature Processor (`feature_processor.py`):** Applies scaling, type enforcement, and analytical transformations. It preserves column alignment between training and inference stages and generates metadata for traceability.
3. **Model Engine (`classification_engine.py`):** Implements a unified interface for machine learning models. The final version includes high-performance classifiers such as `XGBoost` and `RandomForest`, with configurable hyperparameters accessible from the CLI. The module supports:
    - cross-validation,
    - probabilistic outputs,
    - model persistence,
    - deterministic reproducibility through fixed seeds.

A unified execution script (`main.py`) coordinates the pipeline and enables end-to-end operation through the following command:

```
python src/main.py --model xgboost
```

This design ensures that the system can be run, replicated, extended, and benchmarked using a single high-level instruction.

7.1.5 Experimental Design

The experimental design adheres to principles of repeatable scientific evaluation. The workflow of each experiment follows:

1. Load and validate the dataset using the Data Loader.
2. Apply preprocessing and feature transformations.
3. Train a model selected through CLI parameters.
4. Evaluate the model using log-loss and complementary metrics.
5. Generate artifacts including plots, logs, and prediction files.

All experiments were version-controlled using Git, with structured commit messages documenting changes at the module level. Intermediate results, parameters, and metadata are automatically stored to guarantee traceability.

7.1.6 Summary

In summary, the methodology evolved from a high-level system design into a modular, fully operational implementation suitable for empirical experimentation. The final system ensures reproducibility, transparency, and scalability, combining structured engineering practices with machine learning experimentation. The implemented architecture provides a robust foundation for future improvement phases, including hyperparameter optimization, ensemble methods, and production-ready deployment.

Chapter 8

Results

8.1 Results

This section presents the empirical results obtained from the full implementation of the Otto Group Product Classification System. The evaluation includes model-level performance (log loss and accuracy), calibration analysis, feature engineering impact, ensemble weighting, and external validation through Kaggle submission. All results derive from the execution logs of the system pipeline.

8.1.1 Dataset Summary

The dataset was composed of 61,878 training samples and 144,368 test samples, each described by 93 numerical features. After validation, no missing values or duplicated rows were detected. Table 8.1 summarizes the class distribution.

Table 8.1: Class distribution in the Otto training dataset.

Class	Count
Class_2	16122
Class_6	14135
Class_8	8464
Class_3	8004
Class_9	4955
Class_7	2839
Class_5	2739
Class_4	2691
Class_1	1929

8.1.2 Feature Engineering Effects

Feature engineering maintained the original 93-dimensional representation in the baseline configuration, and an alternate configuration with PCA (50 components) explained 81% of the variance. Table 8.2 summarizes the feature engineering results for both configurations logged in the experiment.

Table 8.2: Feature Engineering Summary.

Configuration	Final Features	Variance Explained
Baseline (No PCA)	93	N/A
PCA, 50 components	50	0.81

8.1.3 Model Performance

Four models were trained and evaluated:

- Random Forest (300 trees, depth=20)
- Multi-Layer Perceptron (256–128 layers)
- Logistic Regression (baseline)
- XGBoost (300 trees, depth=6, lr=0.1)

Table 8.3 presents the results for the principal run (no PCA), where XGBoost was selected as the best-performing model.

Table 8.3: Model performance on validation set (No PCA configuration).

Model	Log Loss	Accuracy
XGBoost	0.4788	0.8161
MLP	0.5208	0.8092
Logistic Regression	0.6311	0.7683
Random Forest	0.6632	0.7834

The PCA experiment (50 components) produced the results in Table 8.4. In this configuration, the MLP achieved the best validation performance.

Table 8.4: Model performance with PCA (50 components).

Model	Log Loss	Accuracy
MLP	0.5320	0.7948
XGBoost	0.5327	0.7925
Random Forest	0.6080	0.7853
Logistic Regression	0.6841	0.7511

8.1.4 Calibration Results

All models were calibrated using isotonic regression. Table 8.5 summarizes the log-loss before and after calibration for the main experiment.

Table 8.5: Calibration results (isotonic method).

Model	Before	After	Improvement
Random Forest	0.6632	0.2589	+0.4044
XGBoost	0.4788	0.2378	+0.2409
MLP	0.5208	0.4143	+0.1064
Logistic Regression	0.6311	0.8668	-0.2358

8.1.5 Ensemble Weighting

An optimal soft-voting ensemble was constructed using calibrated validation log loss. The calculated weights are shown in Table 8.6.

Table 8.6: Optimal ensemble weights based on calibrated validation log loss.

Model	Weight
Random Forest	0.3320
XGBoost	0.3614
MLP	0.2074
Logistic Regression	0.0991

8.1.6 Submission Results

The system successfully generated a `submission.csv` file with shape $(144368, 9)$, corresponding to probabilistic predictions for each of the nine classes.

Kaggle evaluated the submission with a public leaderboard log-loss score of:

0.56122

Table 8.7 shows the comparison between internal validation and external Kaggle evaluation.

Table 8.7: Internal vs. external (Kaggle) evaluation.

Metric	Value
Best internal log loss (XGBoost)	0.4788
Final calibrated ensemble log loss (expected)	≈ 0.24
Kaggle leaderboard log loss	0.56122

The difference between internal and external scores indicates mild distribution shift and highlights the advantage of further hyperparameter tuning and model regularization.

8.1.7 Summary of Findings

The main findings from the experimental pipeline are:

- XGBoost achieved the best single-model performance in the main configuration.
- PCA reduced dimensionality by 46%, but degraded performance compared to the no-PCA setup.
- Calibration significantly improved log-loss for ensemble components.
- The ensemble successfully combined complementary models through learned weights.
- The Kaggle submission achieved a competitive score of 0.56122 log loss.

The results demonstrate that the system produces stable and reproducible evaluations, forming a solid foundation for future optimization and deployment.

8.1.8 Model Description and Training Setup CA

The proposed model, denoted **GraphCA**, implements a graph-based cellular automaton designed for probabilistic multiclass classification. Each instance is represented as a node in a k-nearest-neighbor graph, where class probabilities are iteratively refined by blending self-predictions with neighborhood information.

The architecture consists of:

- a linear initialization layer producing initial logits,
- two learnable mixing coefficients (α, β) controlling the balance between self-influence and neighbor influence,
- an iterative CA update rule applied for $T = 3$ refinement steps.

The training setup is summarized as follows:

- Optimizer: Adam with learning rate 10^{-3},
- Loss: negative log-likelihood applied to $\log(P)$,
- Training: initial 80/20 train–validation split followed by additional training on the full training set,
- Total epochs: **1000** (full-batch),
- Graph construction: Euclidean k-nearest neighbors with $k = 5$,
- Feature preprocessing: StandardScaler with `with_mean=False`.

This configuration yields a lightweight model that leverages local geometric structure rather than deep nonlinear transformations.

8.1.9 Validation Performance CA

Internal validation experiments showed stable behavior, with log-loss values typically in the range 0.49–0.53 and accuracies around 0.79–0.81. However, the external evaluation on Kaggle reported a log-loss of **1.0877**, indicating a substantial performance drop compared to the internal results.

Table 8.8: GraphCA — internal vs. external performance.

Metric	Value
Internal validation log-loss	≈ 0.50
Internal validation accuracy	≈ 0.80
Kaggle leaderboard log-loss	**1.0877**
Neighbors (k)	5
Epochs	1000

The discrepancy suggests overfitting, miscalibrated probabilities, or a distribution shift between the internal validation split and the external test data.

8.1.10 Training Dynamics, CA Behavior, and Diagnostic Analysis

Several factors may explain the significant performance gap between internal validation and the Kaggle leaderboard score:

1. **Overfitting due to full-batch training and 1000 epochs.** Without early stopping, the model may overfit the training graph structure. *Recommendation:* reduce the number of epochs or apply early stopping and regularization.

2. **Probability miscalibration.** Overconfident but incorrect predictions typically increase log-loss on external data. *Recommendation:* apply post-hoc calibration (Platt or isotonic) on a dedicated calibration set.

3. **Distribution shift between validation and test.** If the data distribution differs across splits, neighborhood propagation can amplify errors. *Recommendation:* use stratified K-fold cross-validation and analyze feature statistics across splits.

4. **Mismatch in neighbor construction.** Validation neighbors were computed as val→val, whereas test neighbors are test→train. This inconsistency may artificially inflate validation performance. *Recommendation:* compute validation neighbors relative to the training set to better match test conditions.

5. **Numerical stability concerns.** Iterative CA updates may produce degenerate probability vectors. *Recommendation:* use `log_softmax` internally for improved stability.

6. **Sensitivity to CA hyperparameters** (k, T, α, β)**.** The chosen values may be suboptimal. *Recommendation:* evaluate $k \in \{5, 10, 20\}$ and $T \in \{1, \ldots, 6\}$.

7. **Low-capacity initialization network.** A single linear layer may be insufficient to produce high-quality initial logits. *Recommendation:* replace it with a shallow MLP.

8. **Lack of stochasticity in full-batch training.** Mini-batch noise often improves generalization. *Recommendation:* explore mini-batch or graph-batch training.

8.1.11 Submission Results and External Evaluation CA

The final submission produced a `submission.csv` file containing $(144368, 9)$ class-probability outputs. Kaggle evaluated the model with a public leaderboard log-loss of:

1.0877

Table 8.9: Internal vs. external (Kaggle) evaluation for GraphCA.

Metric	Value
Internal validation log-loss	≈ 0.50
Kaggle leaderboard log-loss	**1.0877**
Performance gap	$\approx +0.58$

Chapter 9

Discussion and conclusions

9.1 Discussion

The results obtained from the implemented pipeline provide meaningful insights into the behavior of different classification models within the context of the Otto Group Product Classification task. The primary objective of the system was to design a modular and reproducible architecture capable of evaluating multiple machine learning algorithms, calibrating their probabilistic outputs, and generating competitive predictions for external validation. The empirical findings directly support this objective and reveal several important implications.

First, the superior performance of XGBoost in the baseline configuration (log loss of 0.4788 and accuracy of 0.8161) aligns with established research demonstrating the effectiveness of gradient boosting methods for high-dimensional tabular data. Its performance advantage over classical models such as logistic regression and random forests reflects its ability to capture nonlinear interactions and complex feature dependencies. This is consistent with previous findings in the Kaggle competition, where XGBoost-based solutions frequently ranked among the top-performing approaches.

Second, the results indicate that dimensionality reduction via PCA was not beneficial for this specific dataset. Although PCA successfully reduced the feature space from 93 to 50 dimensions while preserving 81% of the variance, both XGBoost and MLP experienced performance degradation. This suggests that the original feature representation contains nonlinear structures that are not preserved in a purely linear transformation, an observation supported by previous studies showing that PCA can underperform in tasks dominated by nonlinear boundaries.

Third, the calibration process revealed substantial improvements in model reliability, especially for Random Forest and XGBoost, whose calibrated log-loss values decreased to 0.2589 and 0.2378, respectively. This improvement confirms the pipeline's capability to produce well-calibrated probability distributions, a crucial aspect of multi-class log-loss optimization. However, calibration negatively impacted logistic regression, highlighting that isotonic regression may overfit on models that already produce smooth probability surfaces.

The ensemble analysis demonstrated that combining models through soft voting can further improve predictive consistency, though the final Kaggle score (0.56122) showed a moderate gap when compared to internal validation metrics. This reinforces the presence of distributional differences between the validation split and the competition's scoring set. Such discrepancies are well-documented in machine learning research and emphasize the need for more robust cross-validation strategies or domain adaptation methods.

Despite the strengths of the system, some limitations remain. The pipeline does not incorporate hyperparameter optimization, advanced regularization, or automated architecture

search, which restricts its performance ceiling. Additionally, although the system supports PCA, it does not explore alternative nonlinear dimensionality reduction techniques such as autoencoders or manifold learning, which could preserve feature interactions lost under PCA. Finally, ensemble weighting is derived solely from validation log-loss, without exploring stacking or meta-learners that may capture complementary model behavior more effectively.

Overall, the discussion demonstrates that the implemented system succeeds in meeting its original objectives while revealing opportunities for further methodological refinement and optimization.

9.2 Conclusions

This study presented the design and implementation of a modular and fully reproducible system for multi-class product classification within the Otto Group dataset. The main findings indicate that gradient boosting (XGBoost) consistently outperforms other tested models and provides the strongest predictive accuracy among the evaluated approaches. The MLP also showed competitive performance, particularly under the PCA configuration, confirming its ability to model nonlinear relationships when sufficient feature representation is preserved.

Calibration significantly improved the quality of predicted probabilities for most models, enabling the construction of calibrated ensembles capable of producing stable and interpretable outputs. The generation of a valid `submission.csv` file and its evaluation on Kaggle (log loss of 0.56122) further demonstrated the system's practical applicability and its alignment with real-world benchmarking standards.

The implemented architecture proved robust, transparent, and extensible, fulfilling the methodological objectives of reproducibility, modularity, and traceability. Its structured design allows future researchers to integrate additional models, perform hyperparameter tuning, or incorporate advanced feature engineering without altering the system's core pipeline.

Future work should focus on optimizing model configurations through systematic hyperparameter search, exploring nonlinear dimensionality reduction, and evaluating alternative ensemble methods such as stacking or blending. Additional experiments with augmented validation strategies, including stratified cross-validation or bootstrapping, may also reduce discrepancies between internal metrics and external leaderboard performance.

The Cellular Automata (CA) model used in this experiment shows both the potential and the limitations of applying CA-style updates to real machine learning tasks. On the positive side, the model is able to take advantage of local structure in the data: each sample updates its prediction by looking at the predictions of its nearest neighbors, and this iterative process helps the system reach more refined probability estimates during training. This behavior works well on the internal validation split, where the CA model produced reasonable log-loss values and stable learning dynamics.

However, when the model was evaluated on the Kaggle test set, the performance dropped noticeably, reaching a log-loss of 1.0832. This suggests that the CA model has trouble generalizing when the test data does not fully match the structure seen during training. Since the method depends heavily on k-nearest-neighbor graphs, any shift in the feature distribution can change the neighborhood relationships and affect how information is propagated. In practice, this makes the CA model sensitive to noise, graph mismatches, and overfitting—especially after long training (1000 epochs).

Even with these limitations, the experiment is still valuable. It shows that CA-based models can be trained, can capture useful patterns, and offer an interesting alternative to standard approaches. The results also point to clear directions for improvement, such as using stronger

initial networks, adding calibration steps, experimenting with different graph constructions, or developing hybrid models that combine CA updates with neural embeddings.

Overall, while the CA model did not outperform classical methods in this challenge, it provides a solid starting point for future work. The experiment highlights both what CA models can do and what they still struggle with, offering practical insight into how neighborhood-driven learning might be made more robust in future versions.

In conclusion, the results confirm that the developed system provides a solid foundation for effective multi-class classification and serves as a reliable platform for continued experimentation, improvement, and deployment in data-driven product categorization tasks.

References

Chen, T. and Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 785–794.

Niculescu-Mizil, A. and Caruana, R. (2005), Predicting good probabilities with supervised learning, *in* 'Proceedings of the 22nd International Conference on Machine Learning (ICML)', pp. 625–632.

Platt, J. (1999), 'Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods', *Advances in Large Margin Classifiers* pp. 61–74.

Titericz, G. and Semenov, S. (2015), '1st place solution - otto group product classification challenge', https://www.kaggle.com/competitions/otto-group-product-classification-challenge/discussion/14335. Accessed October 2025.

Wolpert, D. H. (1992), 'Stacked generalization', *Neural Networks* **5**(2), 241–259.