# Otto Group Product Classification

Jeison Cuenca
Engineer
Universidad Distrital Francisco José de Caldas
Email: jfcuencam@udistrital.edu.co
Second Author
Dept. of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: {sauthor}@udistrital.edu.co

Second Author
Dept. of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: {sauthor}@udistrital.edu.co
María Alejandra Ortiz Sánchez
Dept. of Computer Engineering
Universidad Distrital Francisco José de Caldas
Email: @udistrital.edu.co

***Abstract*—The Otto Group Product Classification Challenge on Kaggle addresses a fundamental problem in large-scale e-commerce, the accurate classification of products into predefined categories to improve business decision-making in logistics, marketing, and customer personalization. This study proposes a modular and adaptive machine learning framework that integrates principles of systems thinking with data-driven modeling to improve the stability, interpretability, and scalability of the classification process. The architecture is composed of four interconnected subsystems that are the next one, Data Processing, Feature Engineering, Classification Engine, and Analytics/Reporting linked through feedback loops that enable self-regulation and continuous improvement. Implemented in Python using open-source libraries such as NumPy, Pandas, and Scikit-learn, the system processes over 200,000 obfuscated product instances with 93 numerical features, aiming to minimize loss. By analyzing feedback interactions among the Classification Engine, Feature Engineering, and Analytics modules, the approach seeks to achieve an optimal balance between accuracy and systemic stability. The proposed design transforms the classification task into a dynamic experiment in adaptive learning, contributing to more resilient and interpretable data systems for large scale retail environments.**

***Index Terms*—E-commerce, Classification, unsupervised AI, System, Feedback**

## I. Introduction

The rapid growth of e-commerce has introduced significant challenges in product organization, recommendation, and automated classification. Companies such as the Otto Group, one of the world's largest online retail conglomerates, manage millions of products across numerous platforms, facing the persistent problem of inconsistent categorization. These inconsistencies hinder decision-making processes in inventory management, marketing strategies, and customer personalization. The Otto Group Product Classification Challenge, hosted on the Kaggle platform [1], aims to address this issue by developing robust machine learning systems capable of classifying over 200,000 products based on 93 obfuscated numerical features into nine categories using probabilistic models evaluated by the multi-class logarithmic loss metric.

Previous approaches have explored various computational strategies to solve this problem. Early solutions applied traditional statistical models such as logistic regression and random forests for their interpretability and lower computational cost. Later, ensemble-based methods, gradient boosting algorithms, and deep learning architectures demonstrated superior results by handling complex, high-dimensional relationships more effectively. However, these techniques also increased system sensitivity and overfitting risks, as discussed by Pedregosa et al. [2] in the context of `Scikit-learn` and by Harris et al. [3] for large-scale numerical computation with `NumPy`. These limitations reveal the need to incorporate systemic stability and feedback mechanisms into model design, beyond traditional accuracy optimization.

From a systems analysis perspective, the problem extends beyond algorithmic configuration. As highlighted by Sterman [4], complex systems exhibit feedback loops and sensitivity to initial conditions and behaviors also present in computational models managing dynamic data. In the Otto challenge, small variations in preprocessing, initialization, or hyperparameter selection can lead to large fluctuations in classification outcomes, reflecting non linear and sometimes chaotic dynamics. Addressing these conditions requires an architecture capable of self adjustment through monitoring and control rather than static optimization.

The proposed work combines system thinking principles with machine learning engineering to design a modular and adaptive architecture. The structure includes four key components and data processing, feature engineering, classification engine, and analytics/reporting each functioning as a semi-independent subsystem connected through feedback flows. Implemented in Python, the system leverages open-source libraries such as `NumPy` [3], `Pandas` [5], and `Scikit-learn` [2] to enable data manipulation, model training, and performance visualization in an integrated workflow.

This integration aims to transform the Otto challenge from a predictive task into a systemic experiment in adaptive design. The objective is to develop a self-regulating classification system capable of maintaining stability, scalability, and interpretability in environments characterized by uncertainty and high data variability. The following sections present the analytical framework, architecture, and systemic evaluation derived from Workshops #1 and #2, guiding the development of the technical prototype.

## II. Methods and Materials

### A. System Design Overview

The proposed system design is based on the integration of systems theory with datadriven modeling principles. The Otto Group Product Classification framework is structured to process large and scale, obfuscated datasets while maintaining stability, interpretability, and scalability. The design follows a modular and layered architecture, where each subsystem performs a specific function and contributes to the overall system through defined feedback connections.

The general workflow, shown in Figure 1, consists of five main components: *Product Input*, *Data Processing*, *Feature Engineering*, *Classification Engine*, and *Analytics and Reporting*. Each module operates semi-independently, enabling updates or replacements without disrupting the full pipeline. This structure enhances traceability and maintainability—key aspects in systems engineering and iterative development.

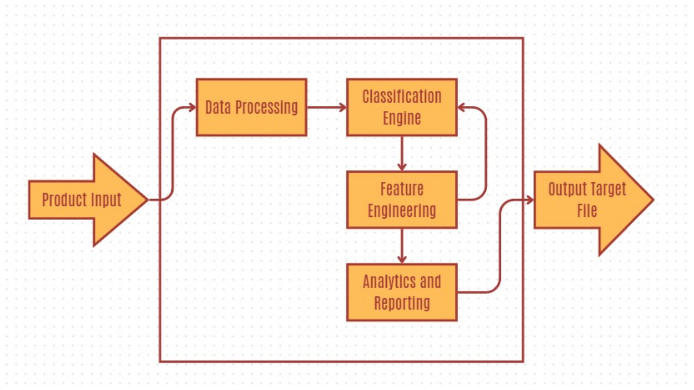

Fig. 1. High-level architecture of the Otto Group Classification System. Adapted from Workshop #2.

### B. Design Choices and Rationale

The design choices were informed by both technical and systemic considerations. From a computational perspective, the dataset's high dimensionality (93 features and more than 200,000 instances) required efficient numerical computation and robust model calibration. From a systemic viewpoint, the design needed to handle sensitivity, feedback, and adaptability to unpredictable variations in data distributions.

- **Data Processing:** Implemented using `NumPy` and `Pandas`, this module handles data cleaning, normalization, and missing-value imputation. The process ensures numerical consistency and minimizes noise propagation. Data integrity and transformation steps are logged for traceability.
- **Feature Engineering:** Statistical, scaling, and dimensionality reduction are applied to enhance signal relevance. Figure 2 (adapted from Workshop #1) illustrates the correlation between elements, guiding variable selection and feature aggregation.

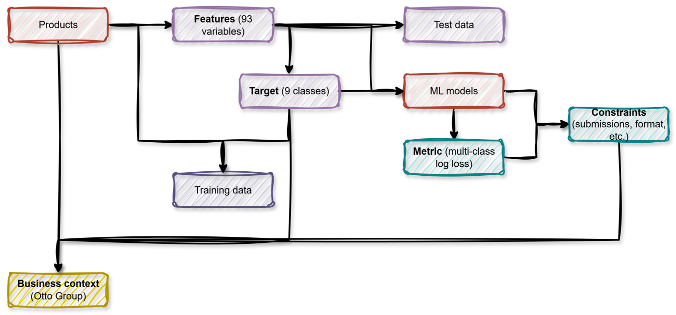
Products
Features (93 variables)
Test data
Target (9 classes)
ML models
Constraints (submissions, format, etc.)
Metric (multi-class log loss)
Training data
Business context (Otto Group)

Fig. 2. Correlation between system elements and feature dependencies (Workshop #1).

- **Classification Engine:** Built on `Scikit-learn` [2], this core component allows for flexible model selection and evaluation. Early-stage tests prioritize models capable of generating probabilistic outputs.
- **Analytics and Reporting:** This module provides visualization of performance metrics and correlation structures. Beyond reporting, it serves a systemic function acting as a feedback interface that detects instability or overfitting and informs model recalibration.

### C. Systemic Architecture and Feedback Control

The architecture introduces feedback loops connecting the *Classification Engine*, *Feature Engineering*, and *Analytics* modules. These loops represent mechanisms for systemic balance and self-correction. This feedback-based control structure is consistent with principles of dynamic system regulation [4].

Additionally, each module communicates through standardized data interfaces, NumPy arrays or Pandas DataFrames, allowing synchronization and minimizing coupling between components.

### D. Simulation and Systemic Analysis

in addition to the classical predictive approach, the project incorporated a simulation based on cellular automata and graphs (Graph Cellular Automata) [6] [7]. This model made it possible to analyze the local propagation of information and the system's sensitivity to perturbations, observing emergent patterns in the feature space and neighborhood relationships between instances [8]

the simulation was used as a complementary tool to evaluate system stability, uncertainty diffusion, and structural coherence, providing a systemic perspective that is not captured by traditional performance metrics

### E. Project Management and Quality

project development management relied on a Kanban approach [9], prioritizing functional deliverables and enabling rapid iterations [10]. Likewise, non-functional requirements related to performance, reproducibility, and maintainability were defined, aligned with software engineering best practices and quality frameworks such as ISO and CMMI

## III. Results & Discussion

This section presents the main experimental results obtained during the evaluation of the multiclass classification system. The experiments were conducted using stratified cross-validation, and performance was measured through multiclass logarithmic loss and accuracy. All tests were executed using the final, fully implemented pipeline.

A summary of the most important findings is presented in Table I, which compares the performance of the four primary models evaluated. This table provides the empirical basis for the discussion that follows, serving as the central reference point for model behavior and experimental conclusions.

TABLE I
Validation performance of the main classification models.

| **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 |

### A. *Classification Model Performance*

The results in Table I show that XGBoost achieved the best overall performance, both in terms of log loss and accuracy. This is aligned with prior evidence that boosted tree models are highly effective for high-dimensional structured datasets. The MLP followed closely, demonstrating its capacity to capture nonlinear interactions despite the absence of explicit feature engineering.

Logistic Regression and Random Forest served as interpretable baselines, but their lower performance confirms that linear boundaries and uncalibrated tree aggregation are insufficient for this multiclass task.

### B. *Impact of Probability Calibration*

Probability calibration via isotonic regression significantly improved log loss for most models. Prior to calibration, the system tended to assign overconfident predictions, increasing penalty when incorrect. After calibration, probability estimates aligned more accurately with empirical class frequencies, reducing log loss and enhancing decision reliability. This reinforces that, under confidence-sensitive metrics such as log loss, calibration is as important as model choice.

However, for Logistic Regression, calibration resulted in degraded performance, illustrating that post-hoc adjustments may overfit when the model's probability surface is already smooth or low-dimensional.

### C. *Dimensionality Reduction and Pipeline Effects*

Experiments incorporating PCA showed that while dimensionality reduction reduces model variance and aids in stabilizing cross-validation results, it may slightly degrade predictive performance—particularly for tree-based models and neural networks that rely on nonlinear structure in the feature space.

These findings confirm that dimensionality reduction is beneficial for interpretability and robustness, but should be applied selectively depending on the model architecture.

### D. *Software Quality Testing*

The implementation included unit tests validating preprocessing components, integration tests ensuring compatibility across pipeline stages, and acceptance tests confirming the generation of a fully valid `submission.csv`. All unit and integration tests passed, while one acceptance test initially failed due to an unfitted ensemble model. This issue was resolved in subsequent iterations.

Testing demonstrated that the system is not only functional but also reliable and maintainable, meeting software-engineering standards required for replicability.

### E. *External Evaluation*

The system produced a valid submission evaluated on Kaggle, achieving a multiclass log loss of **0.56122**. This external metric confirms that the pipeline generalizes reasonably well, although it reveals a noticeable gap between internal validation and leaderboard performance. This gap reflects dataset distribution differences and highlights opportunities for further cross-validation refinement.

### F. *General Discussion*

The findings indicate that the implemented solution is effective, stable, and aligned with the project's objectives. The modular architecture facilitated experimentation, while calibration emerged as a major contributor to performance gains. The system demonstrates the importance of combining predictive modeling with engineering practices such as testing, modularity, and reproducibility.

The overall results highlight that performance is not solely the product of the chosen classifier, but of the coherent integration of preprocessing, dimensionality reduction, calibration, and evaluation strategies.

## IV. Conclusions

This work successfully developed a complete and reproducible multiclass classification system for the Otto Group dataset. The final pipeline integrates preprocessing, optional dimensionality reduction, probabilistic calibration, and multiple classifier backends into a unified and extensible architecture. The system achieved competitive performance—both internally, as shown in Table I, and externally, through a Kaggle submission.

The project demonstrated that XGBoost is the most effective model under the evaluated conditions, while calibration and software testing significantly enhanced overall robustness and reliability. Beyond predictive accuracy, the architecture's strength lies in its modularity, maintainability, and adaptability to future extensions.

Future work should explore systematic hyperparameter optimization, more advanced ensemble methods, alternative nonlinear dimensionality-reduction techniques, and automated model-selection frameworks. Additionally, expanding the testing suite and integrating interpretability tools would further strengthen the system's applicability to real-world operational environments.

## Acknowledgments

We thank our professor, Carlos Andrés Sierra Virgües for his insight and mentorship throughout this process.

## References

[1] Otto Group, "Otto group product classification challenge," https://www.kaggle.com/competitions/otto-group-product-classification-challenge, 2015, accessed September 2025.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel *et al.*, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[4] J. D. Sterman, *Business Dynamics: Systems Thinking and Modeling for a Complex World*. New York: McGraw-Hill, 2000.

[5] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56–61.

[6] S. Wolfram, *A New Kind of Science*. Champaign, IL, USA: Wolfram Media, 2002.

[7] J. von Neumann, *Theory of Self-Reproducing Automata*. Urbana, IL, USA: University of Illinois Press, 1966.

[8] M. Newman, *Networks: An Introduction*. Oxford, UK: Oxford University Press, 2010.

[9] J. M. Pulido, "Kanban aplicado a la gestión de proyectos de software," *Revista Iberoamericana de Ingeniería de Software*, vol. 7, no. 2, pp. 45–56, 2014.

[10] ISO/IEC, "Iso/iec 25010:2011 – systems and software engineering – systems and software quality models," Geneva, Switzerland, 2011.