UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA
Ingeniería de Sistemas

Otto Group Product Classification Challenge, A System Design

Systems Analysis & Design

Group 020-82

Workshop #2

Students

Juan Diego Lozada 20222020014
Juan Pablo Mosquera 20221020026
María Alejandra Ortiz Sánchez 20242020223
Jeison Felipe Cuenca: 20242020043

Professor: Carlos Andrés Sierra Virguez

Date: October 2025

Contents

1 Workshop #1 Review

1.1 Summary of System Analysis Outcomes

The first workshop allowed us to understand the Otto Group Product Classification Challenge as a complex information system where several interacting components determine the system's overall behavior. The analysis identified critical constraints related to the nature of the data, the evaluation metric, and the competition's procedural rules. The system depends on large volumes of numerical data with unknown meaning, which limits the direct use of domain knowledge and forces the design of general, data-oriented processing methods. Another key constraint is the requirement that probability predictions must be accurate and consistent, since the evaluation metric penalizes uncertainty or overconfidence.

1.2 Data Characteristics

The dataset contains more than 200,000 product records described by 93 numerical variables. These features do not have explicit labels, which introduces uncertainty about their interpretation and relationships. In addition, the product categories are not evenly distributed, producing an unbalanced system that must be carefully managed to avoid biased results. Because of the data volume and complexity, the system must include mechanisms for efficient processing, error detection, and validation of outputs.

1.3 Chaos-Theory and Sensitivity Factors

During the analysis, it was observed that small changes in data handling, validation methods, or parameter selection can cause noticeable variations in results. This sensitivity suggests that the system behaves in a non-linear way, similar to what is described in chaos theory: minor modifications may lead to unexpected or amplified effects. Furthermore, the feedback process between internal validation and the public competition results introduces dynamic interactions that affect the model's stability and make prediction behavior less predictable.

1.4 Design Implications

The design ideas for the next stage should directly respond to these findings. It is necessary to emphasize the stability of the system, ensuring that data flows, model configurations, and evaluation processes are consistent and traceable. Simplifying the data processing steps, establishing clear validation criteria, and managing uncertainty are essential actions. Additionally, the design must include strategies to monitor performance variations and minimize the effect of random or chaotic behavior on the overall system response.

2 System Requirements

2.1 Design Requirements:

From the system analysis carried out in Workshop #1, the Otto Group Product Classification system is characterized by high dimensionality, data obfuscation, and the need for consistent classification under probabilistic evaluation. Based on these conditions, several measurable requirements can be defined:

- **Performance:** The system must process and classify more than 200,000 product records efficiently, maintaining acceptable response times during model training and evaluation. The data pipeline should support batch operations and parallel computation to reduce latency.
- **Reliability:** The classification process must produce stable and reproducible results under equivalent configurations. Small perturbations in data input or parameters should not cause significant deviations in the final predictions, ensuring systemic robustness.
- **Accuracy and Calibration:** Since the competition metric depends on probabilistic output, the system must generate well-calibrated probability distributions across all categories, minimizing overconfidence and improving prediction coherence.
- **Scalability:** The design should allow for incremental data updates or integration with larger datasets without requiring complete retraining. This ensures adaptability and system growth over time.
- **Traceability:** All stages of preprocessing, model configuration, and validation must be documented and traceable, enabling verification and future replication of results.

2.2 User-Centric and Systemic Considerations:

Even though the primary objective of the system is data classification, user-oriented and systemic properties remain essential for practical deployment:

- **Ease of Use:** Interfaces or configuration files should allow users to adjust parameters and execute the system without requiring deep technical intervention, improving accessibility.
- **Interpretability:** Despite the data being obfuscated, the system should include mechanisms to visualize and analyze relationships between variables and outputs, supporting decision-making and system understanding.
- **Security and Data Integrity:** Input and output data must be managed through controlled access and versioning mechanisms to prevent corruption or unauthorized manipulation.

- **System Stability and Feedback Control:** In alignment with systems theory principles, the system must include internal feedback loops to monitor deviations like performance drops or instability and self-adjust to maintain operational equilibrium.
- **Maintainability:** The architecture must allow modular updates of preprocessing components, model configurations, or evaluation metrics without affecting the global functionality.

Overall, these requirements integrate the findings from Workshop #1 with a systemic design vision. The system must balance computational performance with structural stability, ensuring a controlled, observable, and adaptable classification environment.

3 Problem Architecture

3.1 Prototype Architecture

The next figure shows the high-level architecture of the proposed unsupervised learning system for e-commerce product classification. The system aims to automatically group and categorize products based on their inherent characteristics without the need for labeled data. This architecture provides a structured pipeline that includes data processing, feature extraction, model execution, and analytical evaluation.

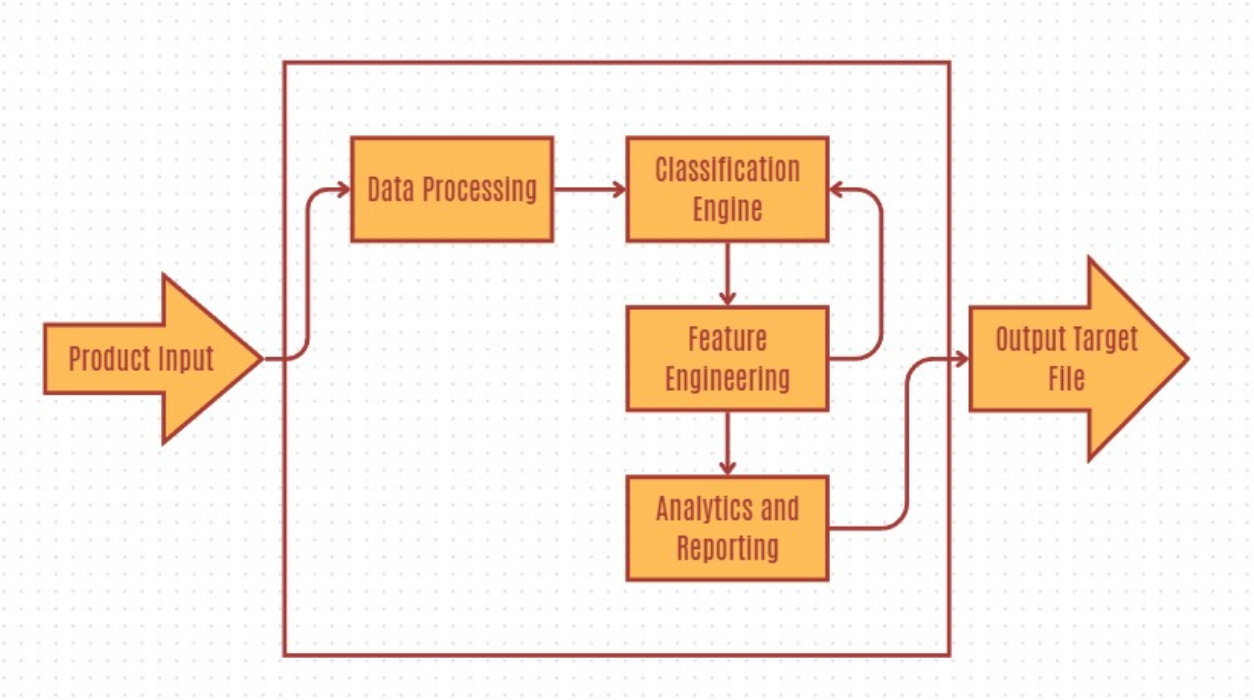

Figure 1: High-Level Architecture

The architecture consists of the following components:

- **Product Input:** This component represents the entry point of the system. It gathers raw product data from multiple sources such as product listings, images, textual descriptions, and metadata. This diverse input is essential for the classification process.
- **Data Processing:** The data processing module prepares the input data by cleaning, normalizing, and transforming it into a consistent format. Tasks include handling missing values, text tokenization, and encoding of categorical or numerical attributes to ensure data quality and usability.
- **Classification Engine:** This is the core of the unsupervised learning process. It applies algorithms such as K-Means, DBSCAN, or hierarchical clustering to group similar products. The classification engine detects patterns and forms product clusters based on shared characteristics.

- **Feature Engineering:** In this stage, relevant features are extracted or constructed to enhance model performance. Examples include text embeddings from product descriptions, image features from convolutional networks, and numerical feature scaling. Well-engineered features allow for better discrimination between product categories.

- **Analytics and Reporting:** This component provides analysis, metrics, and visualization of the classification outcomes. It supports evaluation of clustering quality and generates business insights such as product trends, anomalies, or emerging categories.

- **Output Target File:** The final output of the system consists of structured files containing classified product data. These files can be integrated into other systems, such as recommendation engines, search filters, or inventory management tools, to improve decision-making and user experience.

This architecture ensures a scalable and adaptive framework capable of evolving with the continuous flow of product information in e-commerce environments, while maintaining accuracy and operational efficiency.

4 Addresing Sensitivity And Chaos

4.1 Systemic Sensitivity and Feedback:

The architecture proposed for the Otto Group Product Classification system, shown in Figure, represents a dynamic process where each component depends on the accuracy and consistency of its predecessors. Due to the obfuscated and high-dimensional nature of the data, small changes in preprocessing parameters, input distributions, or feature selection can propagate through the system and significantly alter the final classification. This illustrates the system's sensitivity to initial conditions, a characteristic aligned with chaos theory in complex systems.

To mitigate these effects, the architecture incorporates feedback connections between the *Classification Engine*, *Feature Engineering*, and *Analytics and Reporting* modules. These feedback loops act as stabilizing mechanisms that allow continuous monitoring of system outputs and performance indicators. When deviations or anomalies are detected for example, unexpected accuracy variations or overfitting signs, the feedback allows reconfiguration of parameters to restore balance and maintain system reliability.

4.2 Stability and Control Strategies:

Sensitivity is addressed through structured data validation and modular control. The *Data Processing* stage ensures normalization and consistency of inputs to minimize random perturbations. The *Feature Engineering* component try a verification to prevent instability in downstream modules. Finally, the *Analytics and Reporting* unit provides real-time performance metrics, serving as a control center that supports decision-making and dynamic adjustment of model parameters.

These elements together form a self-regulating system capable of adapting to data variability without losing coherence in outputs.

4.3 System Adaptability and Learning:

Recognizing that uncertainty and randomness cannot be completely eliminated, the architecture is designed to learn from its operational environment. The iterative flow between modules allows the system to update feature transformations and calibration processes based on performance results. This adaptive behavior aligns with the concept of open systems, where continuous interaction with external data enables evolution and resilience over time.

4.4 Conclusion:

Addressing chaos and sensitivity in this architecture involves combining feedback control, monitoring, and adaptability. Instead of attempting to remove uncertainty, the design integrates it as part of the system's natural dynamics, ensuring robustness through observation, correction, and systemic balance.

5 Technical Stack and Implementation

5.1 Programming Environment:

The implementation will be developed in Python, due to its versatility, readability, and extensive ecosystem for data analysis and machine learning. Python provides modularity and supports integration between data processing, model training, and reporting components, which aligns with the system's need for flexibility and maintainability.

5.2 Core Libraries and Tools:

- **NumPy:** Used for numerical computation and efficient matrix operations. It provides the foundation for all data manipulation tasks, ensuring fast and stable arithmetic operations during preprocessing and feature transformation.
- **Pandas:** Facilitates structured data management, cleaning, and transformation. It enables the creation of reproducible data pipelines that integrate directly into the *Data Processing* module of the system.
- **Scikit-learn:** Serves as the main library for model development and evaluation. It offers a wide range of classification algorithms, preprocessing utilities, cross-validation tools, and model calibration functions, fitting the requirements of performance, reliability, and interpretability.
- **Matplotlib:** Used for visualization in the *Analytics and Reporting* component. These libraries allow graphical monitoring of performance metrics, correlations, and validation curves, supporting systemic feedback and reporting loops.

5.3 System integration

Each component of the technical stack contributes to the modular structure of the architecture. The data flow begins with NumPy and Pandas for preprocessing, continues through Scikit-learn for classification and feature engineering, and concludes with visualization libraries for analysis and feedback. This modular approach ensures interoperability, reduces coupling, and simplifies maintenance.

The implementation will follow a modular approach aligned with the system architecture defined in the figure. Each module *Data Processing*, *Feature Engineering*, *Classification Engine*, and *Analytics and Reporting* will be developed as an independent but interoperable component. This structure allows distributed development and facilitates maintenance or future scalability. The general data flow follows a pipeline structure, where each stage processes and passes results to the next component. This pattern promotes modularity, scalability, and reusability, especially for preprocessing and training stages.

6 Conclusions

The development of *Workshop #2: Otto Group Product Classification Challenge, a System Design* enabled a systemic understanding of the product classification problem by integrating concepts from systems analysis, chaos theory, and adaptive architecture design. Through the construction of the model and the definition of functional and non-functional requirements, the main factors affecting the stability, reliability, and performance of the system were identified.

First, it was observed that the nonlinear and high-dimensional nature of the data introduces significant sensitivity to small variations in preprocessing or parameter selection. This finding confirms the applicability of chaos theory principles in machine learning environments, where feedback and internal control are essential to maintain systemic coherence.

Second, the proposed modular and interoperable architecture proved to be an effective strategy to ensure scalability, traceability, and maintainability. The use of libraries such as `NumPy`, `Pandas`, `Scikit-learn`, and `Matplotlib` enabled a consistent, reproducible, and visually monitored data flow, strengthening the principles of systemic observability and control.

Additionally, the inclusion of feedback mechanisms among the *Feature Engineering*, *Classification Engine*, and *Analytics and Reporting* modules enhanced the adaptive learning capacity of the system, allowing for automatic adjustments in response to environmental changes or data variations.

In conclusion, the integration of systems thinking with data-driven design provides a solid foundation for building robust and sustainable solutions in highly complex environments. This approach not only improves the technical performance of the system but also promotes a holistic understanding of the interdependencies that characterize automated classification processes. For future work, the inclusion of deep learning techniques and more advanced self-evaluation mechanisms is suggested to increase precision and stability under dynamic and heterogeneous data conditions.

7 References

References

[1] Otto Group. (2015). *Otto Group Product Classification Challenge*. Kaggle. Recuperado de https://www.kaggle.com/competitions/otto-group-product-classification-challenge/

[2] Sterman, J. D. (2000). *Business dynamics: Systems thinking and modeling for a complex world*. New York: McGraw-Hill.

[3] Forrester, J. W. (1968). *Principles of systems*. Portland, OR: Productivity Press.

[4] Gleick, J. (1987). *Chaos: Making a new science*. New York: Viking.

[5] Meadows, D. H. (2008). *Thinking in systems: A primer*. White River Junction, VT: Chelsea Green Publishing.

[6] Senge, P. M. (1990). *The fifth discipline: The art and practice of the learning organization*. New York: Doubleday.

[7] Sterman, J. D. (2002). All models are wrong: Reflections on becoming a systems scientist. *System Dynamics Review, 18*(4), 501–531.

[8] Holland, J. H. (1998). *Emergence: From chaos to order*. Oxford: Oxford University Press.

[9] Simon, H. A. (1996). *The sciences of the artificial* (3rd ed.). Cambridge, MA: MIT Press.

[10] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., et al. (2020). Array programming with NumPy. *Nature, 585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[11] McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

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

[13] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

[14] Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

[15] Checkland, P. (1999). *Systems thinking, systems practice: Includes a 30-year retrospective*. Chichester: Wiley.

[16] Jackson, M. C. (2003). *Systems thinking: Creative holism for managers*. Chichester: Wiley.

[17] Beer, S. (1979). *The heart of enterprise*. Chichester: Wiley.

[18] Von Bertalanffy, L. (1968). *General system theory: Foundations, development, applications*. New York: George Braziller.