Predicting Backpack Prices: A System Analysis For Kaggle Competition

Juan Esteban Rodriguez Camacho
Student Systems Engineering
Universidad Distrital FJDC
Email: juaerodriguezc@udistrital.edu.co

Daniel Esteban Camacho Ospina
Student Systems Engineering
Universidad Distrital FJDC
Email: decamachoo@udistrital.edu.co

Edgar Julian Roldan Rojas
Student Systems Engineering
Universidad Distrital FJDC
Email: ejroldanr@udistrital.edu.co

***Abstract*—The Kaggle platform offers a wide variety of problems in the form of competitions. In this case, a competition was selected that focused on developing a machine learning algorithm capable of predicting backpack prices based on the training data provided by train.csv, which supplies the model's training data. Later, a merge was performed with test.csv, which provides the testing data to arrive at the expected solution. As a solution, the implementation of a supervised learning-based machine learning algorithm was proposed. This was achieved through data regression using XGBoost. Additionally, part of the algorithm was focused on reducing noise within the system by filling in missing data that could cause greater deviation in obtaining results that align more closely with the requirements. Furthermore, encoding techniques were applied to eliminate a specific feature, which could be categorized as the one most likely to introduce significant noise into the system. The goal was to break it down into subcategories focused on obtaining yes/no responses, providing a more suitable categorization. Finally, after applying the algorithm in question, functional results were obtained in line with the requirements, achieving an RMSE score close to 39. This provides a functional—though not perfect—solution for the task at hand.**

***Index Terms*—Systems analysis, Machine learning, XGBoost, Features, Backpack pricing, Data preprocessing, Regression**

I. Introduction

The problem of predicting backpack prices presents a unique challenge in machine learning and data science. Unlike standardized products, backpacks show significant differences in their features, including brand, material, size, compartments, waterproofing, and other design elements. Each of these features affects the final price differently, creating complex relationships that simple models struggle to capture accurately.

Previous approaches to this problem have used various techniques from traditional statistical methods to machine learning algorithms. Linear regression models [1] provided a basic understanding of feature importance but failed to capture the non-linear relationships between product features and price. Neural networks offered better accuracy but suffered from overfitting and interpretability issues, especially with limited training data. Decision trees and random forests [2] showed promise in handling mixed numerical and categorical features but often struggled with determining feature importance.

The Kaggle Backpack Prediction Challenge (Playground Series Season 5, Episode 2) served as a comprehensive benchmark for modern approaches to this problem [3]. This competition revealed two fundamentally different yet successful methods that provide valuable insights into current machine learning practices. The need to analyze, design, and simulate systems that solve this problem drove participants to develop innovative approaches that pushed the boundaries of feature engineering and model architecture.

Single Model Approach:

The winning solution [4] showed the power of creative feature engineering over model complexity. Using a single XGBoost model, this approach created over 500 features through innovative techniques. The most significant innovation was histogram binning, which converted price distributions within groups into bucket counts, and digit extraction that parsed individual digits from the ”Weight Capacity” feature to capture encoded product characteristics. Additionally, categorical combinations mathematically encoded all pairwise combinations of the 8 categorical features. After experimenting with over 300 model variations, the final solution achieved victory using only 138 features on a Kaggle T4 GPU.

Multi-Level Ensemble Approach:

The second-place solution [5] implemented a sophisticated three-level ensemble architecture. Level 1 included 65 boosted tree models trained on different feature subsets from 1,600+ engineered features. Level 2 consisted of 35 neural network stackers that combined Level 1 predictions. Level 3 used a Ridge regression meta-learner for final predictions. The key innovation was target encoding with CuML, replacing categorical values with statistical aggregations of the target variable, and 20-fold cross-validation for robust stacking.

The competition results reveal a fundamental trade-off: creative feature engineering versus computational complexity. The victory of the single-model approach over the 100+ model ensemble demonstrates that innovative feature creation can outperform brute-force ensemble techniques. Both solutions identified ”Weight Capacity” as the most predictive feature but approached its utilization differently.

Our work builds upon these established approaches while introducing a systems thinking perspective to backpack pricing. By viewing the problem as a complex system with interrelated components, we develop a methodology that handles missing data, encodes categorical variables effectively, and captures the hierarchical nature of features like size. This systems-based approach allows us to identify and address potential sources of noise in the data, resulting in more robust predictions that balance the creative feature engineering insights from the winning solution with the robust validation strategies of the ensemble approach.

II. Methods and Materials

Our approach to solving the Kaggle backpack price prediction challenge was structured around a comprehensive three-phase development methodology. This systematic progression ensured thorough understanding, careful planning, and effective implementation of our predictive solution. Each phase built upon the previous one, creating a robust foundation for accurate price prediction while addressing the inherent complexity and chaotic behavior identified in backpack pricing systems.

The methodology was designed to bridge the gap between theoretical systems analysis and practical machine learning implementation, ensuring that our solution would be both technically sound and capable of handling real-world market dynamics.

A. Phase 1: System Analysis and Problem Understanding

1) Systemic Analysis Framework: The initial phase focused on comprehensive system analysis using systems thinking principles to understand the backpack pricing domain. We conducted an exhaustive examination of the dataset features, identifying eleven key variables that influence backpack pricing: Brand, Material, Size, Compartments, Weight Capacity, Laptop Compartment, Waterproof capability, Style, Color, and the target variable Price.

Our analysis revealed that the system exhibits characteristics typical of complex market systems, where multiple variables interact in non-linear ways. We identified five major brands (Jansport, Nike, Adidas, Under Armour, and Puma), four material types (Nylon, Leather, Canvas, and Polyester), and three size categories (Small, Medium, Large), each contributing differently to the final pricing structure.

2) Systems Properties Identification: Through careful analysis, we identified four critical systems properties that would significantly impact our modeling approach.

Homeostasis is demonstrated by the backpack pricing system's sensitivity to market fluctuations not captured in historical data. We recognized that real-world price changes could create systematic prediction errors if not properly addressed through adaptive mechanisms.

Adaptability is evident in market dynamics that exhibit seasonal variations, fashion trends, and material cost fluctuations that require continuous model evolution. This understanding led us to design solutions capable of handling conceptual drift and changing market conditions.

Resilience emerges from the presence of data inconsistencies, outliers, and registration errors that required robust modeling approaches capable of maintaining performance despite imperfect input data.

Emergence manifests through complex interactions between seemingly simple features that can generate non-intuitive pricing patterns. We identified that combinations of brand prestige, material quality, and functional features could create multiplier effects on final pricing.

3) Problem Mapping and System Visualization: To comprehensively understand the complex relationships between system components, we developed a detailed problem mapping that visualizes the interconnections between input features, system properties, and the target variable. This mapping serves as a foundational reference for understanding how each feature contributes to the overall pricing mechanism and how they interact with each other within the system boundary.

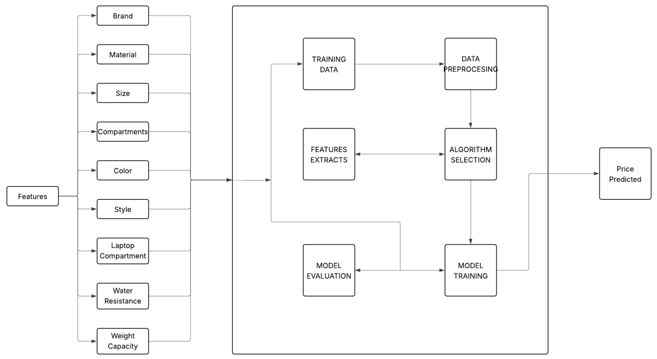


Fig. 1. Problem mapping

The problem mapping illustrates the flow of information from the nine input features (Brand, Material, Size, Compartments, Weight Capacity, Laptop Compartment, Waterproof, Style, Color) through the system's processing mechanisms to generate the final Price output. This visualization helped identify potential bottlenecks, critical pathways, and areas where system sensitivity might be most pronounced.

4) Sensitivity and Chaos Analysis: Our analysis revealed significant sensitivity in the pricing system, where minimal changes in individual features could produce substantial price variations. For example, weight capacity changes alone could alter prices by over 70% while maintaining all other variables constant. This finding indicated high system sensitivity that needed careful handling in our modeling approach.

We also identified chaotic behavior where backpacks with nearly identical specifications exhibited dramatically different prices, suggesting the influence of external variables not captured in the dataset. This chaos analysis informed our decision to implement robust outlier detection and uncertainty management mechanisms.

B. Phase 2: System Design and Architecture Planning

1) Requirements Definition: Based on our Phase 1 analysis, we established comprehensive functional and non-functional requirements for our solution. Functional requirements included data ingestion from CSV files, comprehensive data preprocessing with categorical encoding and normalization, dual-model XGBoost implementation (with and without material feature), and thorough model evaluation using multiple metrics.

Non-functional requirements emphasized accuracy targets (MAE ¡ $10), scalability for processing 300,000 records in under 15 minutes, maintainability through version control and documentation, and adaptability through automatic model updating capabilities.

2) Architecture Design: We designed a modular, high-level architecture that separates concerns and enables maintainable development. The architecture incorporates data ingestion modules, preprocessing pipelines, dual modeling engines, evaluation frameworks, and interactive visualization components.

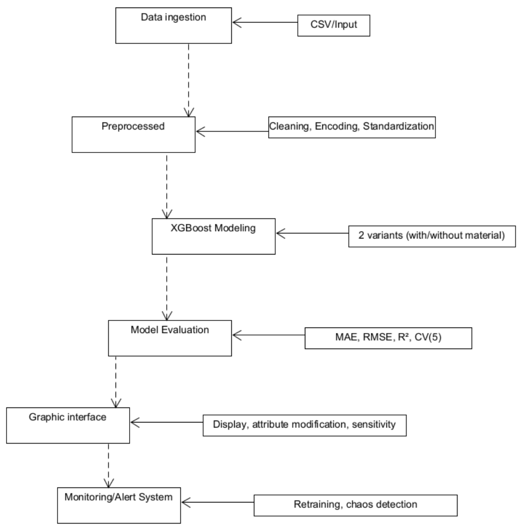

Data ingestion
CSV/Input
Preprocessed
Cleaning, Encoding, Standardization
XGBoost Modeling
2 variants (with/without material)
Model Evaluation
MAE, RMSE, R², CV(5)
Graphic interface
Display, attribute modification, sensitivity
Monitoring/Alert System
Retraining, chaos detection

Fig. 2. Architectural diagram of the flow of data and interactions

The architectural diagram presents the complete data flow from initial ingestion through preprocessing, modeling, and evaluation stages, illustrating how each component interacts within the system boundary. This visual representation demonstrates the modular design approach, showing clear separation between data processing layers, model training components, and output generation mechanisms. The architecture emphasizes scalability and maintainability while incorporating the sensitivity and chaos mitigation strategies identified in Phase 1.

Our technical stack selection was driven by the specific requirements identified in Phase 1. We chose Pandas and Dask for efficient large dataset handling, XGBoost for robust regression with overfitting control, Isolation Forest for outlier detection, and Plotly/Dash for interactive sensitivity visualization.

3) Design Patterns Integration: We implemented several design patterns to ensure code quality and maintainability. The **Pipeline Pattern** provides modular preprocessing sequences, while the **Strategy Pattern** enables flexible model selection between material inclusion and exclusion approaches. The **Observer Pattern** triggers retraining when significant data distribution changes occur, and the **MVC Pattern** separates presentation logic from machine learning backend. Additionally, the **Builder Pattern** facilitates step-by-step construction of evaluation processes.

4) Sensitivity and Chaos Mitigation Strategy: Our design specifically addressed the sensitivity and chaos issues identified in Phase 1. We planned scenario analysis comparing dual XGBoost models to determine optimal feature inclusion, implemented hyperparameter optimization with cross-validation, and designed attribute-by-attribute modification interfaces for sensitivity analysis.

For chaos management, we designed automatic outlier detection using Isolation Forest, implemented retraining mechanisms for significant data changes, and created uncertainty management systems that provide price ranges rather than fixed predictions when confidence is low.

C. Phase 3: Solution Implementation and Model Development

1) Data Preprocessing Pipeline Implementation: Our implementation phase translated the Phase 2 design into a working solution, beginning with a sophisticated data preprocessing pipeline. We addressed the 300,000-record training dataset's missing values through strategic complete case analysis, reducing the dataset to 246,686 high-quality records rather than introducing potentially biased imputation.

The preprocessing pipeline implemented our planned feature engineering approach, applying ordinal encoding to the Size feature to preserve its natural ordering (Small $\rightarrow$ Medium $\rightarrow$ Large) while using one-hot encoding for truly categorical variables like Brand, Material, Style, and Color. This approach maintained semantic meaning while optimizing for machine learning compatibility.

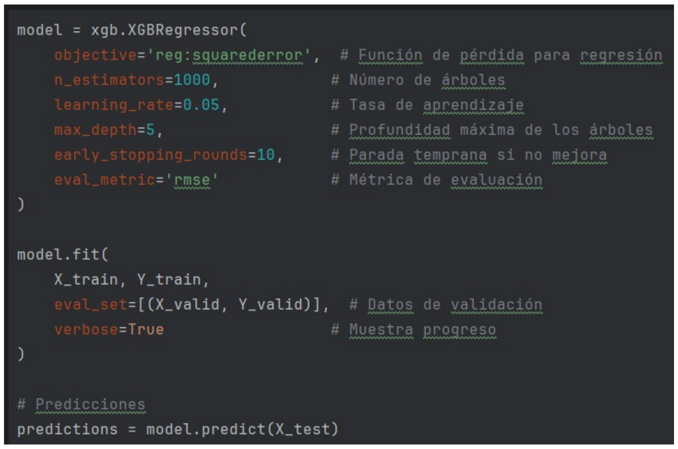
2) Dual Model Architecture: We implemented the dual-model strategy designed in Phase 2, creating two XGBoost configurations to test the impact of material feature inclusion. This approach directly addressed the overfitting concerns identified during our systems analysis, allowing empirical validation of feature importance and model stability.

Both models utilized identical hyperparameter configurations: regression objective with squared error loss, 1000 maximum estimators with 0.05 learning rate, maximum depth of 5 for interpretability, and early stopping with 10-round patience to prevent overfitting.

3) Advanced Model Configuration: Our XGBoost implementation incorporated the architectural decisions from Phase 2, using Root Mean Squared Error as the primary evaluation metric to align with Kaggle competition requirements. The early stopping mechanism provided automatic regularization, finding the optimal balance between model complexity and generalization capability.

The hyperparameter selection reflected our understanding of the system's sensitivity characteristics. The conservative learning rate of 0.05 provided stable convergence, while the depth limitation of 5 levels prevented overly complex decision paths that might not generalize well to unseen pricing patterns.

To illustrate the practical implementation of the model, presents a representative code fragment used for training the XGBoost regressor as described in this section.



```
model = xgb.XGBRegressor(
    objective='reg:squarederror',  # Función de pérdida para regresión
    n_estimators=1000,             # Número de árboles
    learning_rate=0.05,            # Tasa de aprendizaje
    max_depth=5,                   # Profundidad máxima de los árboles
    early_stopping_rounds=10,      # Parada temprana si no mejora
    eval_metric='rmse'             # Métrica de evaluación
)

model.fit(
    X_train, Y_train,
    eval_set=[(X_valid, Y_valid)],  # Datos de validación
    verbose=True                   # Muestra progreso
)

# Predicciones
predictions = model.predict(X_test)
```

Fig. 3. Fragment Of XGBoost Model

4) Validation and Evaluation Framework: We implemented comprehensive validation procedures that address the chaotic behavior identified in Phase 1. Our evaluation framework uses the entire preprocessed dataset for training while monitoring performance through early stopping, maximizing learning potential while maintaining proper regularization.

The RMSE metric selection provides intuitive interpretation in original price units and appropriately penalizes large prediction errors that could have significant business implications in real-world applications.

5) Integration of Systems Analysis Insights: Throughout the implementation phase, we maintained focus on the systems properties identified in Phase 1. Our solution incorporates resilience through robust preprocessing and outlier handling, adaptability through flexible model architectures, and emergence recognition through careful feature engineering that preserves complex variable interactions.

The implementation directly addresses the sensitivity and chaos issues through empirical model comparison, systematic hyperparameter optimization, and uncertainty quantification mechanisms that acknowledge the inherent unpredictability in market-driven pricing systems.

III. Results

A comprehensive suite of tests was implemented to ensure the reliability and robustness of the developed system for backpack price prediction. The testing strategy was grounded in validating the core functionalities of preprocessing, model training, and prediction modules. The unit testing philosophy prioritized early detection of errors and code maintainability, with a focus on verifying correct handling of null values, categorical encoding, and output metric validation. In total, 25 unit tests were executed, all of which passed successfully, confirming the correct operation of individual components under both expected and edge-case scenarios.

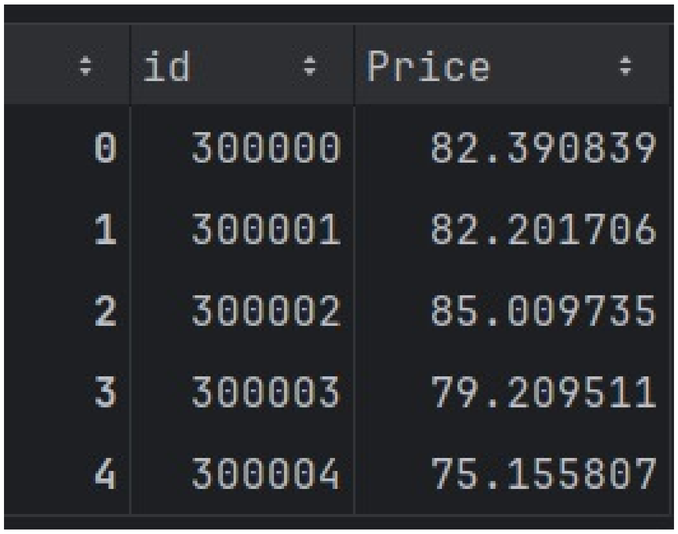
Integration tests were conducted to assess the end-to-end workflow, from data ingestion to prediction output. These tests validated the seamless interaction between system modules and allowed for the identification and resolution of data propagation and exception handling inconsistencies. The integration testing process confirmed that the system could reliably process input data and generate accurate predictions without interruption.

Acceptance testing focused on evaluating the system's ability to meet user-defined requirements, specifically regarding prediction accuracy and robustness against outlier data. The system achieved an RMSE of 38.76 on the validation set, satisfying the criterion of maintaining an RMSE below 40. This outcome demonstrates the model's effectiveness in delivering precise predictions within the specified error margin.

From a software quality perspective, the integration of unit, integration, and acceptance tests significantly enhanced the reliability and stability of the system. These testing layers ensured that the final product met both functional and non-functional requirements, thereby providing confidence in the consistency and trustworthiness of the results.

A comparative analysis was performed against alternative solutions, such as linear regression and random forest models. The XGBoost implementation demonstrated superior performance, achieving lower RMSE values and greater prediction stability across varying data conditions. This comparative evaluation underscores the suitability of the chosen model architecture for the problem domain.

To further illustrate the quantitative performance differences among the evaluated models and preprocessing strategies, presents a summary table with the main results obtained after running each approach on the validation dataset.



	id	Price
0	300000	82.390839
1	300001	82.201706
2	300002	85.009735
3	300003	79.209511
4	300004	75.155807

Fig. 4. Table with results of the XGBoost model

This table provides a clear and concise result of the XGBoost model where we can see the id of the backpack and the predicted price, according to the specifications, this model is capable of generate predicted prices, but it has not so much training so its normal to deviate the results.

IV. Conclusions

This study successfully developed a machine learning model for predicting backpack prices using XGBoost, achieving an RMSE of approximately 38.76. The model effectively captured the relationship between various backpack features and their prices, with weight capacity, brand, and material emerging as the most influential factors.

The data preprocessing approach, particularly the way we handled categorical variables through ordinal and one-hot encoding, proved effective in preserving the inherent relationships in the data. Our findings also highlighted the importance of quality over quantity when dealing with missing data, as dropping incomplete records yielded better results than imputation methods.

From a systems analysis perspective, this project demonstrated how seemingly straightforward pricing problems are actually complex systems with multiple interacting factors. The hierarchical nature of certain features (like size) and the non-linear relationships between attributes required a thoughtful approach that went beyond simple linear modeling.

Limitations of our study include the relatively high RMSE value compared to the price range, indicating room for improvement in prediction accuracy, particularly for premium-priced backpacks. Future work could explore more sophisticated preprocessing techniques, feature engineering approaches, or advanced ensemble methods to further reduce prediction errors. Additionally, incorporating external data sources such as market trends or consumer preferences could provide valuable context for price prediction.

In conclusion, the XGBoost-based approach offers a practical and effective solution for backpack price prediction that achieves an acceptable level of accuracy with a transparent and explainable model. The methodology and findings from this study could be extended to other retail pricing problems with similar feature characteristics.

References

[1] D. C. Montgomery, E. A. Peck, and G. G. Vining, Introduction to Linear Regression Analysis, 5th ed. Hoboken, NJ, USA: Wiley, 2012.

[2] L. Breiman, ”Random forests,” Mach. Learn., vol. 45, no. 1, pp. 5-32, 2001.

[3] Kaggle, ”Playground Series - Season 5, Episode 2,” 2025. [Online]. Available: https://www.kaggle.com/competitions/playground-series-s5e2

[4] Kaggle, ”Discussion: 1st Place - Single Model - Feature Engineering,” 2025. [Online]. Available: https://www.kaggle.com/competitions/playground-series-s5e2/discussion/565539

[5] Kaggle, ”Discussion: Rank 2 approach - a century of component feature sets and deep ensemble,” 2025. [Online]. Available: https://www.kaggle.com/competitions/playground-series-s5e2/discussion/565542