Workshop 2 - Kaggle System Design

Daniel Esteban Camacho Ospina
20231020046
Universidad Distrital FJDC

Edgar Julian Roldan Rojas
20241020041
Universidad Distrital FJDC

Juan Esteban Rodriguez Camacho
20241020029
Universidad Distrital FJDC

***Abstract*—This paper presents a robust system design for backpack price prediction, addressing the analytical findings from Workshop 1's Kaggle competition analysis. The proposed architecture integrates machine learning techniques with systems thinking principles to handle sensitivity and chaotic behavior observed in pricing patterns. We develop a dual-model approach using XGBoost, comparing performance with and without the material feature to mitigate overfitting risks. The system incorporates advanced preprocessing pipelines, outlier detection using Isolation Forest, and automated retraining mechanisms. Our design emphasizes both functional requirements (data processing, modeling, evaluation) and non-functional requirements (scalability, maintainability, adaptability). The implementation leverages modern tools including Dask for large dataset handling, Plotly/Dash for interactive visualization, and Docker for reproducible deployment. Through careful application of design patterns (Pipeline, Strategy, Observer) and rigorous validation methods, we create a system that not only predicts prices accurately but also maintains stability in the face of inherent market chaos and data uncertainty.**

I. REVIEW WORKSHOP 1 FINDINGS

During the development of Workshop1, the system analysis of a Kaggle competition was carried out, which consisted of predicting the price of backpacks from the pre-training of a system. From the analysis of the features provided by the dataset, relevant variables are identified that can produce critical changes or that express a direct affectation to the system, at the same time, external conditions that produce changes in the system are highlighted, introducing chaos and balancing it so that it is not completely based on logic.

From this we can highlight relevant aspects of the analysis by way of synthesis of the previous work, among which we will find:

- Features: After the analysis, variables such as brand, material, size, maximum weight, number of compartments, color, impermeability, and laptop compartment were identified as the variables that will directly influence the final price of the backpacks.
- Base model used: A multiple linear regression was implemented as the program logic using a linear regression with scikit-learn, which will allow establishing relationships between the predictor variables and the price.
- Sensitivity: It was observed that small variations in certain attributes, such as weight capacity or material, generated large changes in the final price, indicating a high level of sensitivity to variable inputs.
- Chaos and nonlinearity: Chaotic behavior was detected in the system, since backpacks with practically equal characteristics have prices that can vary to a large or small extent, which leads us to consider that there are variables not visible in the dataset that affect the final value of the backpack or aspects that have not been taken into account directly from the chaos in the system.

Finally, within the features, a possible relationship between the material and the rest of the features is identified at a glance, which can cause an overfitting in the machine learning model, which complicates the analysis of the data for correct price predictions.

II. DEFINE SYSTEM REQUIREMENTS

For the development of the system it is necessary to separate the requirements into functional and non-functional for a better guide in its development and to ensure a good implementation, reflecting the importance of separating requirements focused on functionalities and those focused on improving the user experience:

A. Functional requirements

- Data Ingest: To be performed from the file provided by Kaggle (CSV) or inputs from the user.
- Data processing: Cleaning, coding of categorical variables (One hot encoding), normalization of numeric variables (Min-Max) and feature analysis.
- Modeling with XGBoost: Backpack price prediction based on two models:
  - Model 1: Inclusion of the material variable to obtain the price.
  - Model 2: Exclusion of the material variable to obtain the price.

  In addition to offering specific parameters for these: max_depth = 5, learning_rate = 0.1, reg_lambda = 2 (overfitting control).
- Model evaluation: Take into account metrics such as: R^2, MAE, RMSE (RMSE is the output metric for the Kaggle rating), as well as offer stratified cross-validation (5 folds).
- Sensitivity analysis: pose partial dependency plots, which monitor the impact of a specific attribute on the price obtained.

B. Non-functional requirements

- Accuracy: Obtaining a model with high prediction level qualified from MAE $<$ \$10.
- Scalability: Ability to process high volumes of data (300k at a rate of less than 15 minutes) using Pandas with Chunks or Dask for basic parallelization.

- Maintainability: Version control, clear documentation and reproducible environment.
- Adaptability: Automatic updating of the model in case of data changes.
- Modularity: Code organized by modules, for possible reuse.
- Responsive interface: User interface integration that is comfortable for system use and that can be integrated on a cross-platform level.

III. HIGH-LEVEL ARCHITECTURE AND TECHNICAL STACK/IMPLEMENTATION SKETCH

In view of the analysis set out above in relation to the proposed system and its requirements, the following architectural diagram is modelled to show the flow of information and the interaction between component:

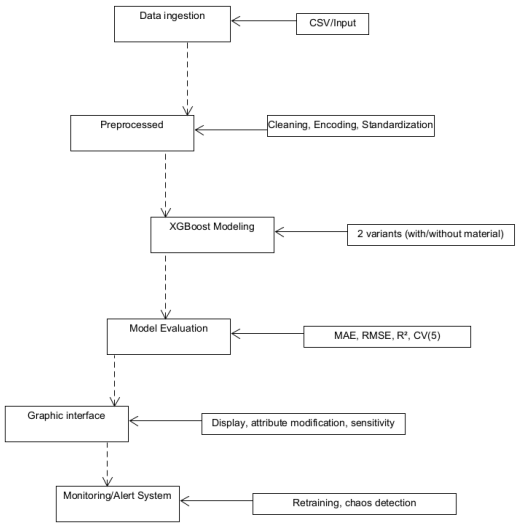

Fig. 1. Architectural diagram of the flow of data and interactions

Accordingly, each module, the tools used, patterns applied and the justification for the use of each of them will be presented below:

A. Used Tools

- Ingestion/Preprocessing:
  - Tools: Pandas, Dask
  - Justification: Better reading and handling of large datasets
- Modeling:
  - Tools: XGBoost, Scikit-learn
  - Justification: XGBoost for regression robustness and overfitting handling
- Validation:
  - Tool: Cross_val_score
  - Justification: Reliable cross-validation
- Outliers & Chaos:
  - Tool: Isolation Forest
  - Justification: Non-parametric outlier detection
- Visualization:
  - Tools: Plotly/Dash
  - Justification: Interactive sensitivity visualization
- Interface:
  - Tools: Streamlit/Flask
  - Justification: Fast UI integrable with the pipeline
- Deployment:
  - Tool: Docker
  - Justification: Reproducible and scalable environment
- Automation:
  - Tools: Airflow/Cron
  - Justification: Training/validation process automation

B. Implementation Plan and Design Patterns

- Preprocessing:
  - Applied Pattern: Pipeline Pattern
  - Description: Modular sequence of transformation steps
- Modeling:
  - Applied Pattern: Strategy Pattern
  - Description: Allows choosing between models with or without material
- Retraining:
  - Applied Pattern: Observer Pattern
  - Description: Triggers when high deviations or distribution changes are detected
- Interface:
  - Applied Pattern: MVC
  - Description: Separates presentation logic from ML backend
- Validation:
  - Applied Pattern: Builder Pattern
  - Description: Step-by-step construction of evaluation processes

IV. ADDRESING SENSITIVITY AND CHAOS

A. Sensitivity

- Scenario Analysis Based on RMSE: Applied to make a comparison between two XGBoost models trained with/without material variable, with the purpose of verifying if it introduces noise to the system. From this, a selection criterion will be determined, under which, if the RMSE of the Model that excludes the material is $<=$ 10% higher than that of the Model that includes all the features, the material variable will be discarded.
- Hyperparameter adjustment: Applied to optimize parameters with cross-validation using RMSE as metric, making use of reg_lambda = 2 (L2) and subsample = 0.7 to control variance.
- Attribute-by-attribute modification interface: Proposed interfaces to reflect changes in final price following attribute modification, can make use of graphs to better understand the data.

B. Chaos

- Outlier Detection (RMSE-Centric): Method to be applied:
  - Use Isolation Forest to detect outliers in the target (price).
  - Threshold: Records where price is outside ± 3 standard deviations of RMSE
- Automatic retraining: Implementation of retraining when significant deviations in input variables are detected.
- Uncertainty management:In the presence of high-level uncertainty in the system, it will not return a fixed knapsack price, but will instead display an estimated range of reliable values.
- Change detection: Implement change detection method to filter data that directly affect the learning of the proposed model.

V. CONCLUSION

- The way in which the system is modelled allows a good management of the information and the results to be obtained. Also, thanks to the approach with the implementation of double model, identifying the key differences between the presence of specific features and the disappearance of these when analysing them to avoid overfitting, it allows to check the comparison of effectiveness between these two models and to see that when making changes in the parameters, inputs, or the training data set, these results will be present in the rating metric R^2 and the RMSE, always seeking to reduce the error rate as much as possible to obtain better quality key information.

VI. BIBLIOGRAPHY

REFERENCES

[1] Carlos Andres Sierra, Systems Thinking, Systems analysis, Universidad Distrital FJDC, Bogota, Slides, 2025.

[2] Kaggle, Playground Series - Season 5, Episode 2, Kaggle, [En línea]. Disponible en: https://www.kaggle.com/competitions/playground-series-s5e2 [Fecha de acceso: abril 4, 2025].