YZV311E - Data Mining Project

Enes Yüksel
Artificial Intelligence and Data Engineering
Istanbul Technical University, Istanbul, Turkey
yukselen20@itu.edu.tr

Ramazan Hoşgör
Artificial Intelligence and Data Engineering
Istanbul Technical University, Istanbul, Turkey
hosgor21@itu.edu.tr

***Abstract*—Ensuring customers restock essential products in a timely manner is a critical challenge for global e-commerce platforms. This study leverages predictive analytics to anticipate customer needs for frequently purchased items such as personal care products, cleaning supplies, and pet food. Using data mining and machine learning techniques, we develop a robust model to predict when customers will need to replenish these products on a weekly basis. By analyzing historical sales data, we aim to improve customer satisfaction, foster loyalty, and optimize supply chain management. This paper details the methodology, dataset preprocessing, and results of our predictive model development.**

I. Introduction

Global e-commerce platforms offer a diverse range of products designed to meet the daily needs of their customers. These platforms continuously develop strategies to enhance user experience and improve operational efficiency. A significant challenge lies in ensuring that customers receive timely reminders to restock frequently purchased items.

Products such as personal care items, household cleaning supplies, and pet food require regular repurchasing. By predicting when customers will likely need these items, platforms can improve user satisfaction, strengthen loyalty, and optimize inventory management. Addressing this challenge necessitates robust predictive modeling using advanced data mining and machine learning techniques.

In today's competitive e-commerce landscape, accurate predictions of customer purchase behavior have become a key differentiator. Many platforms rely on historical data to identify patterns in buying habits and forecast future needs. However, existing solutions often fail to account for the dynamic nature of consumer behavior, seasonal variations, and product-specific trends. By integrating these factors into a unified predictive framework, our study aims to provide a more nuanced understanding of restocking requirements.

This research leverages a comprehensive dataset comprising transaction records, product characteristics, and customer attributes to develop a predictive model for weekly restocking. Using machine learning algorithms such as CatBoost and feature engineering techniques, we construct a robust model capable of capturing the intricacies of purchase timing. Furthermore, the insights gained from this analysis will not only enhance customer engagement but also streamline inventory and supply chain operations, reducing waste and ensuring availability.

The rest of this paper is organized as follows: Section II describes the dataset and preprocessing steps, Section III outlines the methodology employed, and Section IV presents experimental results and discussion. We conclude with key findings and future directions for this research.

II. Dataset Overview

The datasets used in this study provide comprehensive details regarding customer transactions, product information, and hierarchical relationships among product categories. Below is a detailed description of each dataset:

A. Transactions Dataset

The `transactions.csv` file is the primary dataset used to understand customer purchasing habits. It contains the following key columns:

- **customer_id:** A unique identifier for each customer.
- **product_id:** A unique identifier for each product purchased by the customer.
- **purchase_date:** The date on which the purchase occurred.
- **quantity:** The quantity of the product purchased in a single transaction.

This dataset enables the analysis of customer-product interactions, purchase frequencies, and temporal trends in purchasing behavior.

B. Product Catalog

The `product_catalog.csv` file provides detailed information about each product available on the platform. Key columns include:

- **product_id:** A unique identifier linked to the `transactions.csv` dataset.
- **manufacturer_id:** An identifier representing the product manufacturer.
- **attribute_1 to attribute_5:** Categorical attributes describing various characteristics of the product, such as material, usage type, or category.
- **categories:** A textual field listing the categories to which the product belongs.

The catalog facilitates the extraction of product-level features and supports the hierarchical categorization of products.

C. Product Category Map

The `product_category_map.csv` file defines the hierarchical structure of product categories. Key columns include:

- **category_id:** A unique identifier for each category.
- **parent_category_id:** The identifier for the parent category of each `category_id`, enabling the creation of a category tree.

Despite its potential utility, this dataset posed significant challenges. The hierarchical relationships it defined were difficult to integrate meaningfully into our feature engineering process. As a result, the `product_category_map.csv` was ultimately excluded from our final model due to the lack of a clear and practical use case.

D. Test Dataset

The `test.csv` file contains data structured similarly to the `transactions.csv` dataset but without target values. Key columns include:

- **id:** A unique identifier for each row.
- **customer_id:** The customer identifier.
- **product_id:** The product identifier.
- **prediction:** An empty column to be filled with predicted values.

This dataset is used to evaluate the performance of the predictive model by comparing predicted values with actual outcomes.

E. Dataset Summary

Table I summarizes the key characteristics of the datasets used in this study.

TABLE I
DATASET SUMMARY

Dataset	Rows	Columns
Transactions	*1071538*	4
Product Catalog	*32776*	5+
Product Category Map	*4332*	2
Test	*10000*	4

III. DATASET INTEGRATION AND PREPROCESSING

Our dataset comprises multiple files detailing customer transactions, product attributes, and hierarchical product categories. The preprocessing phase involves cleaning and transforming these datasets to ensure compatibility and maximize feature utility. Specific steps include:

- Handling missing and erroneous data by applying imputation techniques and outlier detection.
- Transforming the `purchase_date` column into usable temporal features such as year, month, week, and day.
- Creating unique customer-product combinations to ensure complete representation of all possible purchase scenarios.
- Merging hierarchical product category information to establish relationships between products.

While creating unique customer-product combinations, the process required a significant amount of memory due to the large volume of data. This resulted in computational errors, making it infeasible to include all combinations. To mitigate this, we limited combinations to products purchased by specific customers and further refined the analysis to monthly purchase patterns. However, this approach introduced an excessive number of zero values, which negatively impacted the model's prediction accuracy by skewing class distributions.

A. Feature Engineering

Feature engineering plays a critical role in improving model performance. We generate features at both customer and product levels:

- **Customer Features:** Aggregated metrics such as average purchase quantity, purchase frequency, and total purchases for each customer.
- **Product Features:** Metrics including average product demand, frequency of purchases, and total units sold.
- **Temporal Features:** Extraction of weekly trends and seasonal patterns to capture time-dependent purchasing behavior.
- **Categorical Encoding:** Encoding customer and product identifiers to allow for seamless integration into machine learning algorithms.

Attempts to identify correlations between attributes in the `product_catalog.csv` file revealed a lack of meaningful relationships, as illustrated in Figure 1. This limited our ability to extract additional insights from product attributes, posing a challenge to enhancing feature richness.

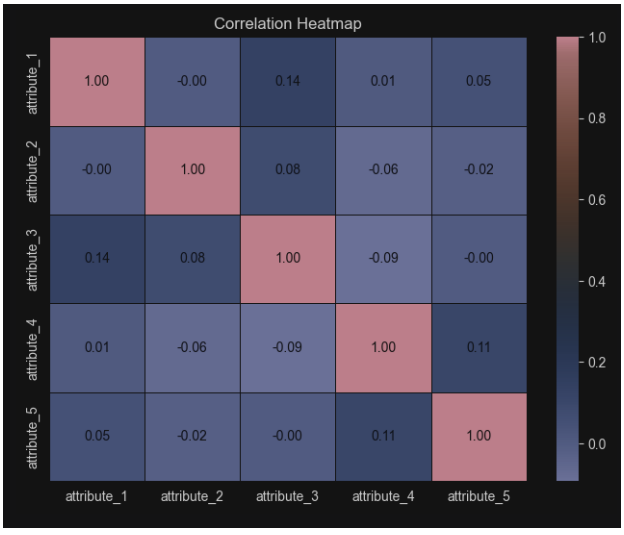

Fig. 1. Correlation Heatmap of Product Attributes

1) Justification for Statistical Features: To enhance the predictive accuracy of the model, several statistical measures were calculated and incorporated as features. These measures aim to capture the underlying patterns and variability in customer purchasing behavior and product demand:

- **Mean:** The average quantity of purchases provides a baseline understanding of customer or product-specific demand trends.
- **Standard Deviation (Std):** Indicates the variability or consistency of purchases. Higher variability might suggest irregular purchasing patterns, which could impact the prediction of restocking needs.
- **Sum:** Represents the total purchases made, helping to identify high-demand products or customers with significant purchasing power.
- **Count/Frequency:** Captures the number of purchase events, offering insight into how frequently a customer buys or a product is sold.

These features were engineered to provide the model with a richer representation of the data, helping it to identify temporal trends, high-demand periods, and customer-specific purchasing patterns. By incorporating these aggregated metrics, the model can better distinguish between regular and irregular purchasing behaviors, ultimately enhancing its ability to predict the week of restocking.

```python
# 4. Generate statistics for customers and products
# Customer statistics
customer_stats = transactions.groupby('customer_id')['quantity'].agg(['mean', 'std', 'sum', 'count']).reset_index()
customer_stats.rename(columns={
    'mean': 'customer_mean',
    'std': 'customer_std',
    'sum': 'customer_total',
    'count': 'customer_frequency'
}, inplace=True)

# Product statistics
product_stats = transactions.groupby('product_id')['quantity'].agg(['mean', 'std', 'sum', 'count']).reset_index()
product_stats.rename(columns={
    'mean': 'product_mean',
    'std': 'product_std',
    'sum': 'product_total',
    'count': 'product_frequency'
}, inplace=True)
```

Fig. 2. Code snippet for generating statistical features: This figure shows the Python code used to calculate statistical measures (mean, standard deviation, sum, and count) for both customers and products based on transaction quantities. These features were designed to capture purchasing behavior and demand variability.

B. Model Development

We employ the CatBoostClassifier, a gradient-boosting decision tree algorithm optimized for handling categorical features. Key characteristics of our model development include:

- Hyperparameter tuning to optimize learning rate, tree depth, and iteration count.
- Integration of categorical features without the need for extensive preprocessing, thanks to CatBoost's native capabilities.
- Robust evaluation using train-test splits and cross-validation to ensure generalizability.

C. Evaluation and Metrics

The effectiveness of our model is evaluated using a combination of metrics:

- **Accuracy:** To measure overall correctness of predictions.
- **Classification Report:** Precision, recall, and F1-score for detailed performance analysis across all target classes.
- **Leaderboard Performance:** Submitting predictions to the Kaggle competition to gauge model efficacy on unseen test data.

D. Challenges Encountered

During the project, we faced several challenges that impacted the workflow and model development process:

- **Memory Constraints:** Creating unique combinations of all customer-product pairs led to excessive memory usage, resulting in system errors. To address this, we refined the combinations to include only products previously purchased by specific customers, but this introduced a high number of zero values in the dataset, making it challenging for the model to achieve accurate predictions.
- **Sparse Target Distribution:** The excessive number of zero values in the dataset significantly skewed the class distribution, affecting the model's ability to generalize effectively.
- **Attribute Correlation:** Analysis of attributes in the `product_catalog.csv` file revealed minimal correlations among the attributes, limiting their utility in feature engineering and reducing the potential to enhance the model's predictive power.
- **Category Map Utilization:** Despite its hierarchical structure, the `product_category_map.csv` file could not be effectively utilized due to difficulties in extracting meaningful features from the category relationships it defined. Consequently, this dataset was excluded from the final model.

E. Expected Outcomes

The proposed model is expected to:

- Accurately predict weekly restocking needs for diverse product categories.
- Improve customer satisfaction through timely notifications.
- Enhance inventory management and reduce operational inefficiencies.

This comprehensive framework combines domain-specific insights with state-of-the-art machine learning techniques to deliver a scalable and effective solution for restocking predictions.

IV. Model Implementation and Results

A. Model Implementation

For this study, we utilized the CatBoostClassifier, a gradient-boosting decision tree algorithm optimized for categorical features. The implementation followed these key steps:

- **Data Preparation:** The dataset was split into training and testing sets, with a 20% holdout for testing. Categorical variables such as `customer_id` and `product_id` were directly fed into the model without preprocessing, thanks to CatBoost's native handling of categorical features.
- **Model Training:** The model was trained with the following hyperparameters:
  - Iterations: 500
  - Learning rate: 0.001
  - Depth: 6

The verbose output helped monitor the training process and ensured convergence during iterative training.

- **Prediction:** Predictions were made for the test set using the trained model. The results were then evaluated using classification metrics.
- **Feature Integration:** Customer-level and product-level features (e.g., average purchase quantity, product frequency) were engineered and merged into the dataset to enrich the predictive power.

B. Performance Metrics

The classification results for the test dataset are summarized in Table II, extracted from the classification report. The model achieved an overall accuracy of 39%, with a weighted F1-score of 37%. The per-class metrics indicate room for improvement, especially in classes with imbalanced data distributions.

Classification Report:

	precision	recall	f1-score	support
1	0.38	0.61	0.47	44802
2	0.40	0.39	0.40	43635
3	0.42	0.32	0.36	40267
4	0.37	0.35	0.36	35985
5	0.33	0.02	0.03	14597
accuracy			0.39	179286
macro avg	0.38	0.34	0.32	179286
weighted avg	0.39	0.39	0.37	179286

TABLE II
CLASSIFICATION REPORT FOR CATBOOST MODEL PREDICTIONS

C. Kaggle Results

The model was evaluated on Kaggle's platform, where it achieved the following scores:

- **Public Leaderboard Score:** 5988
- **Private Leaderboard Score:** 2460

These scores reflect moderate performance and align with the results observed during local evaluation. The gap between public and private leaderboard scores suggests the need for further investigation into dataset splits and model generalization.

D. Observations

- **Strengths:**
  - The CatBoostClassifier efficiently handled categorical variables, reducing preprocessing overhead.
  - The use of aggregated customer and product-level statistics improved feature relevance.
- **Weaknesses:**
  - The imbalance in class distributions (a large number of zero targets) led to biased predictions.
  - The limited correlation among product attributes restricted the impact of feature engineering.

This implementation and evaluation form the foundation for potential iterative improvements, focusing on data balancing techniques and advanced feature selection methods.

V. CONCLUSION

This study developed a framework for predicting weekly product restocking needs using data mining and machine learning. Leveraging transaction data, product attributes, and customer behaviors, the CatBoostClassifier effectively utilized categorical features, achieving moderate prediction accuracy.

Key challenges included memory constraints, sparse target distributions, and minimal correlation among product attributes, highlighting the need for advanced feature engineering. The Kaggle leaderboard results (public score: 5988, private score: 2460) demonstrated the model's potential and areas for refinement.

Future work will address class imbalances, seasonality, promotions, and ensemble methods to enhance performance. This research highlights the value of predictive analytics for scalable and efficient e-commerce restocking solutions.