Returns in Online Retail

GABRIEL WALLON, K. ADITI, and MEHMOOD ALI, University of Colorado Boulder, USA

In this project, we discuss and analyze the topic of retail returns with the help of Data Mining. Retail return is a process where customers return products back to the retailer from whom they were bought. Return services are offered by retail giants like Amazon to even small online retailers as this has become a means to develop customer trust towards the retailers. With the growth of e-commerce around the world, the numbers of returns have exploded as returns are unavoidable in online retail. The reasons for these returns are due to issues with product quality, product expectation, sizing, impulsive buying, buyer's remorse and changing market trends. These returns are almost always free since it is a good strategy to attract customers and fight competition. However, these free-returns are never really free if seen from the perspective of the retailer and the environment. The aim of this project is to use suitable data on retail return and use data mining techniques to segment customers on their return behavior, discover the trends and identify customer patterns for these returns. Upon analyzing the dataset, we build several machine learning models that predict the return accurately.

ACM Reference Format:
Gabriel Wallon, K. Aditi, and Mehmood Ali. 2018. Returns in Online Retail. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Retail returns pose a significant financial challenge for retailers, with expenses stemming from handling returns, restocking inventory, and potential damage to returned items. According to recent reports, these costs exceeded 500 billion dollars in 2022 and are expected to rise annually[1]. The return rate has surged from 8% in 2019 to over 16% in 2022, exacerbating the financial strain on retailers, particularly impacting small businesses (see figure 1). Additionally, beyond its financial toll, retail returns have detrimental environmental consequences. The transportation involved in returns contributes to greenhouse gas emissions, while discarded items often end up in landfills, further depleting resources and exacerbating pollution. Ultimately, the retail return process leaves a substantial carbon footprint, underscoring the need for more sustainable practices in the retail industry. Analyzing return rates is crucial for retailers due to the significant financial and environmental implications associated with retail returns[1].

By understanding and addressing the factors contributing to return rates, retailers can mitigate the financial burden of handling returns, reduce their environmental impact, and implement more sustainable practices. The key idea for visualizing and predicting returns for the chosen datatset is by grouping data with respect to given features. We created different visualizations that gave us insights into product attributes, customer behavior. We performed feature engineering by leveraging insights gained from our exploratory data analysis (EDA) and our intuition regarding the factors influencing customer product returns. This process involved creating and selecting features that capture relevant aspects of the data and are likely to have predictive power in identifying potential return behavior.

Authors' Contact Information: Gabriel Wallon, Gabriel.Wallon@colorado.edu; K. Aditi, kuchibhotla.aditi@colorado.edu; Mehmood Ali, Mehmood.Ali@colorado.edu, University of Colorado Boulder, Boulder, CO, USA.

Manuscript submitted to ACM

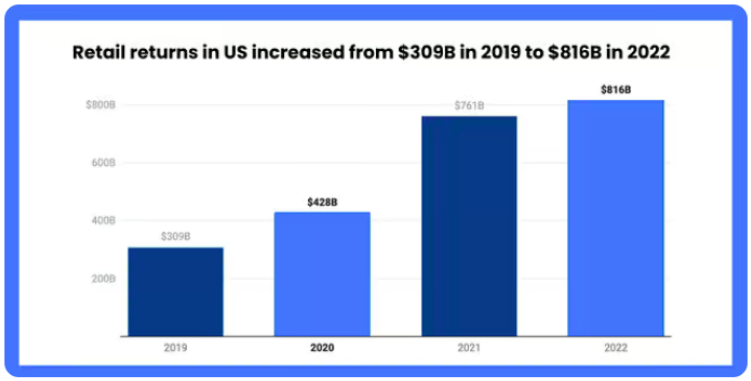


Fig. 1. Increase in retail returns [2]

2 APPROACH

This section provides a comprehensive overview of our approach to addressing the problem. Our methodology comprises two main steps: data preparation/preprocessing and data modeling for predicting returns.

2.1 Data

The dataset utilized for our project was sourced from the Data Mining Cup held in 2016, which consists of retail transaction records. Spanning from January 2-14 to September 2015, the dataset encompasses a vast collection of approximately 2.3 million samples. Each sample within the dataset comprises various features, including but not limited to orderID, detailed product information, pricing details, and the count of returns associated with each transaction. This rich and extensive dataset provides valuable insights into customer behavior, purchasing patterns, and potential factors influencing product returns, making it an ideal choice for our predictive modeling task. Through thorough analysis and exploration of this dataset, we aimed to uncover hidden patterns, understand customer preferences, and develop robust predictive models to anticipate and mitigate product returns in retail environments.

2.2 Data Preprocessing

We initiated our data preprocessing phase with attention to detail to enhance the robustness and facilitate comprehensive analysis of the dataset. We began by addressing the treatment of missing values, a critical step in ensuring the integrity of our data. We removed 353 missing values represented by *NA*, since these records only constituted about 0.015% of our original dataset and it did not have any significant effect the size of our dataset.

Furthermore, we conducted thorough data filtering to enhance the quality of our dataset. This involved removing samples where the quantity was zero, as such instances could potentially skew our analysis and compromise the accuracy of our findings. Additionally, we filtered out records where quantities exceeded the purchased item quantities, as these anomalies could introduce noise and distortions in our analysis. Further, we also found records with 0 price value and ended up filtering such records too. The total number of inconsistent records were found to be 46, 844, which constituted about 2% of the dataset (obtained after dropping missing values).

By addressing missing values and filtering out anomalies, we aimed to create a refined and reliable dataset, setting a solid foundation for subsequent analysis and modeling tasks.

	pricePerItem	totalPrice	truePricePerItem	discountPricePerItem	discountRatioPerItem	quantityPerOrder	originalValPerOrder	avgOrderValue	productGroupsPerOrder	sizesPerOrder	articlesPerOrder	sameArticleIDPerOrder	sameProductGroupPerOrder	sameColorCodePerOrder	samesizeCodePerOrder
0	10.00	30.00	10.00	19.99	0.666556	2	69.98	15.000	1	1	2	0	1	0	1
1	20.00	30.00	20.00	19.99	0.499875	2	69.98	15.000	1	1	2	0	1	0	1
2	35.00	84.99	35.00	14.99	0.299860	2	99.98	42.495	1	1	2	0	1	0	1
3	49.99	84.99	49.99	0.00	0.000000	2	99.98	42.495	1	1	2	0	1	0	1
4	10.00	60.00	10.00	25.99	0.722145	4	151.96	15.000	1	2	3	1	1	0	1

Fig. 2. Example of Feature Engineering

Next, we noticed that the sizes of items were recorded on different scales. The original scales had the following ranges:

(1) 34, 36, 38, 40, 42, 44
(2) 24, 25, 26, 27, 28, 29, 30, 32, 32, 33
(3) 75, 80, 85, 90, 95, 100
(4) A, I

The most common of these ranges was the first one, thus using a frequent itemset method (Apriori Algorithm), we were able to convert all the sizes to this first scale.

2.3 Feature Engineering

The exploratory data analysis (EDA) revealed that the features within the original dataset lacked sufficient predictive power. Given the substantial volume of records at our disposal, we decided to pursue feature engineering as a strategic approach. Feature engineering allows us to either craft novel features or enhance existing ones, thereby capturing the underlying structures within the data more effectively. We found that the new engineered features lead to improved model performance by providing more relevant information for making predictions. Furthermore, one of our approaches to modeling did not consolidate the data according to orderID, where another of our approaches did.

Looking at our data, one natural choice was to group information by features like *orderID, customerID, articleID*, etc. Upon grouping the data, we performed aggregation functions such as *sum, mean, nunique, min, max*, etc., on other features. The statistics/information obtained after aggregation are merged back to the original dataset using the *map* and *transform* operations. Examples of some of the new features that we created are:

- We created a new feature called *quantityPerOrder* which is the total number of articles bought within the same *orderID* tag. We then created a new feature called *totalPrice* which is obtained by summing the *price* after grouping the data by *orderID*. These features further helped us to create a feature called *avgOrderValue* which the ratio of *totalPrice* and *quantityPerOrder*. We also categorized the price level into bins (cheap regular, expensive and luxury) based on the retail price *rrp* which helps us understand the customer behaviour.
- Other engineered features consisted of *productGroupsPerOrder, sizesPerOrder, articlesPerOrder* by grouping data according to *orderID* and applying the aggregation function *nunique* on *productGroup, sizeCode, articleID* respectively. One could imagine that if someone were purchasing a lot of different sizes, they would be more

	aveargeOrderValPerCustomer	aveargeRrpPerCustomer	MaxDiscountPricePerItem	MinDiscountPricePerItem	MeanDiscountPricePerItem
0	28.50	67.98	19.99	19.99	19.990000
1	28.50	67.98	24.99	19.99	22.466415
2	84.99	99.98	34.99	14.99	23.532714
3	84.99	99.98	34.99	0.00	15.041304
4	60.00	151.96	25.99	25.99	25.990000

Fig. 3. Example of Feature Engineering

likely to return an item. Such features give our model the opportunity to learn whether an order with items in different colors and sizes is more or less likely to return an item. We also counted the number of the same articles and the number of the same product groups purchased in each order. It could be possible that these quantities could be correlated with a greater probability of returning one of these items. While the implications of a large number of different product groups or articles in an order are not obvious, including this information in our model gives it the opportunity to uncover such a pattern if it exists.

- We formed several other features by grouping the data according to *customerID* such as *totalOrdersPerCustomer, sizeCodePerCustomer, aveargeOrderValPerCustomer*. We used these new features to divide unique customers (*customerID*) in the dataset into 5 groups by using K-means clustering. It allowed us to avoid one-hot encoding 311165 unique customers which would have been impractical. We instead used *customerGroup* as a feature for the predictions.
- The *sizeCode* feature present in the original dataset were in different units for different *productGroup* such as $\{24, 25, 26, \ldots, 33\}$, $\{XS, S, \ldots, XL\}$, $\{75, 80, 85, \ldots, 100\}$ and $\{34, 36, 38, \ldots, 44\}$. We decided to treat sizes $34-44$ as the standard sizing unit and convert all other sizing units into the standard unit. This mapping/conversion was done using the concept of frequent patterns and association rules. We grouped the data by *customerID* and for each customer we created a set of sizes bought by the customers over the time span of the dataset. The idea was to find frequent patterns like $\{(XS, 34), (XS, 36), \ldots, (XS, 44), (S, 34), \ldots, (S, 44), \ldots (75, 34), \ldots, (100, 44)\}$ and so on. The final mapping was obtained by building associated rules where antecedents were the sizes in the other units and the consequents were the sizes in the standard unit. In order to map each non-standard unit to a unique size in the standard unit, we chose association rules with maximum confidence, for example, $M \Rightarrow 38$ with a confidence value of 0.62.
- We generated new binary features indicating the occurrence of multiple records with the same order ID where customers order multiple products with same features, for example, (size, product group), (color, product group), (size, color, product group), etc. These features aim to indicate a search for the right product size or variant. This behavior may suggest a higher likelihood of product returns, as customers may end up returning items that do not meet their expectations. By flagging such orders, we can potentially improve our understanding of customer behavior and anticipate returns more effectively.
- We also created features indicating whether the order date was a weekend or occurred during holiday months of November or December.Given the occurrence of Christmas and other gifting holidays in November and December, it's plausible that the likelihood of return increases during these periods.
- Discounts and Vouchers: We also incorporated information about discounts on the products *discountPricePerItem* given by the difference between *rrp* and the newly created feature *pricePerItem*. We also created a feature called *truePricePerItem* uisng the *voucherAmount* feature which represents the actual price paid by the customer after applying the voucher.

2.4 Modeling

Through our feature engineering approach, we expanded from 14 features to 67 features, incorporating new features and one-hot encoded features while excluding some original features that may not have been conducive to prediction. Using our augmented datasets, we trained various machine learning models for return prediction. We employ model stacking which is an ensemble learning technique where weak base classifiers are combined to improve predictive performance of the final model. It involves training a set of base models on the training data, then using the predictions

of these base models as features to train the final model. In the context of identifying orders with high probabilities of at least one item being returned, the goal is to save the retailer money, and provide them with expectations regarding the losses they will incur due to returns.

- Logistic Regression: Utilized as the baseline model for binary classification of return or no-return. We performed hyperparameter tuning through Scikit-learn's GridSearchCV[3] for different values of regularizaion parameter $C = \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ and *penalty* parameters of logistic regression model. We found the best values for these hyperparameters to be $C = 0.001$, *penalty* = 'l_2'.
- Ensemble methods: Random Forest, Gradient Boosting, and XGBoost were employed in our analysis to effectively handle the complexity and nonlinearity present in the dataset. Random Forest is a powerful technique that parallelly builds multiple decision trees and combines their predictions to improve accuracy and robustness. It is well-suited for handling high-dimensional data and capturing complex feature relationships just like in our dataset. Unlike Random Forest, Gradient Boosting sequentially trains weak base learners to correct the errors of the previous models, resulting in a strong predictive model. It captures subtle patterns and relationships in the data, making it particularly effective for regression and classification tasks. XGBoost is an optimized implementation of Gradient Boosting which enhances performance by employing regularization techniques and parallel computing. It is known for its scalability and speed on several datasets. By leveraging the strengths of these ensemble methods, we were able to boost our accuracy in predicting returns. GridsearchCV was used to optimize the *criterion* and *maximum depth* parameters, where criterion was chosen as either *entropy* or *gini*, and maximum depth chosen from the list $[2, 4, 6, 8, 10, 12, 14]$. This resulted in an optimal model using gini as the criterion, with a maximum depth of 10.
- Feature Stacking: We implemented a stacked ensemble approach where the first layer comprised of models like Logistic Regression, Random Forest, and Gradient Boosting. Their predictions were then used as features in the second layer model made of XGBoost classifier, leveraging the strengths of individual models for improved predictions.

3 EXPERIMENTS

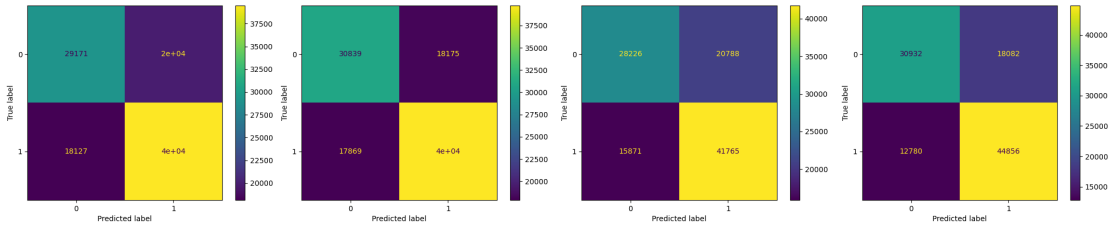
3.1 Evaluation Method

For our base models which from the first layer of our final classifier, we used accuracy along with precision and recall as they are equally important in evaluating the success of a model. Precision is important because we would not like to have a high false positive rate in our classifications, and overestimating the amount of returned items the retailer should expect. However, recall is equally as important, as we would not like to underestimate the number of returned items. Thus, the F1-score is the model evaluation metric we choose to focus on.

For our second layer of XGB classifier, we used ***logloss***, ***mean_squared_error*** and ***mean_absolute_error*** as the evaluation metric. We thresholded the predictions from the second layer at 0.5 in order to get binary classification labels which were further used for calculating **accuracy**, **precision**, **recall** and **F1-score**.

3.2 Experimental details

We had categorical features like *deviceID*, *productGroup*, *paymentMethod* and *priceLevel*, which were one-hot encoded so that we can convert them into a format suitable for machine learning models. On the other hand, numerical features



(a) Logistic Regression (b) Random Forest (c) Gradient Boosting (d) Final Model

Fig. 4. Comparison of Confusion Matrices

that we had were not on the same scale. In order to bring them to a common scale, we applied normalization using the Scikit-learn's *StandardScaler* and this rescaling of the data helped us to remove the biases due to a varied range of data.

Label encoding the *sizeCode* feature instead of using one-hot encoding was a deliberate choice, considering the ordinal nature of sizes where larger values correspond to bigger sizes. This approach effectively captures the ordinal relationship between sizes, allowing the model to understand the inherent hierarchy. By encoding sizes numerically, the model can learn and generalize patterns based on the relative magnitudes of sizes, potentially enhancing its ability to make accurate predictions.

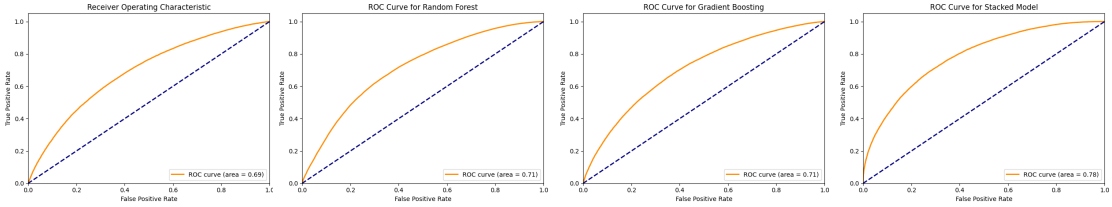
For training purpose, the train-test split we chose was 80-20%. The base classifiers were trained with hyper-parameters found with the help of grid search. The second layer of XGB classifier was trained on the following values of the hyper-parameters: $learning\_rate = 0.1, n\_estimators = 100, max\_depth = 15$.

3.3 Results

The observations from the confusion matrices of different models applied to the retail returns dataset reflects varied levels of predictive accuracy and precision. The model stacking approach shows good prediction performance for both classes. Gradient boosting also performs well, albeit with a slightly higher misclassification rate. Logistic regression displays a higher number of false negatives, which could point to a conservative model. Lastly, the random forest model shows a high number of correct predictions for non-returns but a relatively lower correct prediction rate for actual returns, suggesting a potential bias towards predicting non-returns.

The ROC curves for the stacked model, gradient boosting, and random forest indicate distinct levels of performance in discriminating between returned and not returned items in a retail setting. The stacked model, with an area under the curve (AUC) of 0.78, demonstrates the highest discriminatory ability among the three, indicating a superior balance between true positive rate and false positive rate. Both the gradient boosting and random forest models have an AUC of 0.71, suggesting they have good but identical performance, though not as effective as the stacked model. These ROC curves reflect the trade-off each model makes between sensitivity and specificity and show the stacked model's slight edge in predictive power for return behavior.

We also present the top 10 features that had the most predictive power for the base models. We can observe that all the models rely on quite different features for their predictions indicating the difference in their working principle. Some features that were used by all the models are total price, payment method (BPRG), mean discount per item, etc. We also observe that the binary features we created for indicating products with same pair of features within an order, play a crucial role in prediction. It is very intuitive that these binary features play a big role in suggesting a possible



(a) Logistic Regression (b) Random Forest (c) Gradient Boosting (d) Final Model

Fig. 5. Comparison of ROC

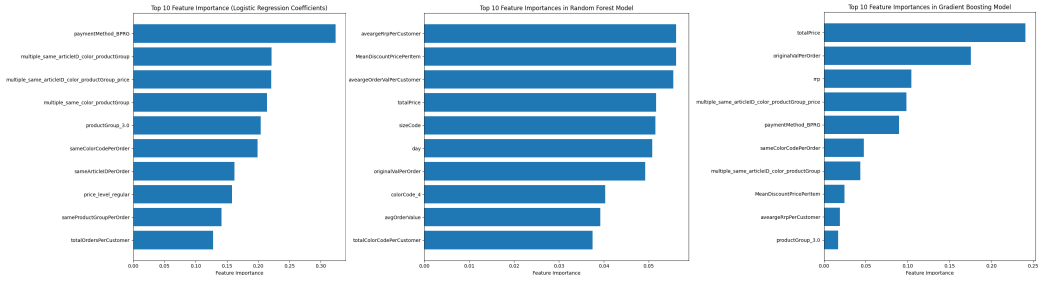
return and this intuition is also confirmed by our results. This shows that our feature engineering helped boost the performance of these ML models, and the data in itself was not representative enough for these models to perform.

Our final result shows that the model stacking approach proved to be powerful technique for the return prediction. We were able to train different weak classifiers, use ensemble learning and model stacking to build a model that was a strong predictor.

3.4 Analysis

The comprehensive analysis of our predictive models has yielded valuable insights into their effectiveness in predicting product returns in retail. Each model exhibits unique strengths and weaknesses, which are discussed below:

- **Logistic Regression:** As the baseline model, offered a fundamental comprehension of the features impacting returns. It did, however, show a larger percentage of false negatives, suggesting a tendency toward conservative prediction. Underestimating the return rate, as a result, could cause issues with inventory and return process management.
- **Random Forest:** Although the model's overall accuracy was strong, it was skewed toward predicting non-returns. Given the higher frequency of non-returns in the dataset, this trend may be due to the inherent nature of the model's handling of imbalanced classes. Because different customer behavior and product interactions result in non-linear correlations, which are typical in retail data, the Random Forest method proved especially successful in managing categorical data.



(a) Logistic Regression (b) Random Forest (c) Gradient Boosting

Fig. 6. Top Features in Prediction

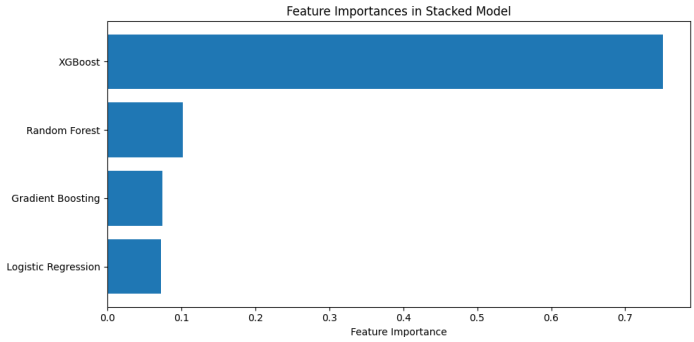


Fig. 7. Comparison between different Models

- **Gradient Boosting:** When it came to identifying small trends and connections that other models frequently miss, the Gradient Boosting model did well. It outperformed the logistic regression model in terms of misclassification rates, but similar to Random Forest, it also demonstrated a higher accuracy for non-returns than actual returns.
- **XGBoost:** The model improved prediction performance by utilizing the base models' outputs as part of the stacked ensemble. With the highest AUC of all the models, it successfully reduced false positives as well as false negatives. This suggests that the model has a good balance in its capacity to correctly categorize returns and non-returns.
- **Stacked Model:** The ensemble method using layered structure of XGBoost and Logistic Regression, Random Forest, and Gradient Boosting showed the best predictive power. In addition to balancing the advantages and disadvantages of the various base models, this model decreased bias and variance, producing predictions that were more accurate. The retail dataset's many scenarios and data kinds were better represented by the model thanks to the stacking technique.

3.5 Performance Comparison

- **Accuracy and Precision:** In terms of accuracy and precision, the stacked model performed better than any other model, which is important for ensuring that the predictions are accurate and appropriate.
- **Recall and F1-Score** The Gradient Boosting and XGBoost models outperformed Logistic Regression and Random Forest in terms of recall, demonstrating their efficacy in reducing false negatives. The ensemble technique appears to offer the optimal balance between recall and precision, based on the F1-score obtained across all models.
- **AUC (Area Under-Curve)** With an AUC of 0.78, the stacked model had the highest AUC, followed by Random Forest and Gradient Boosting at 0.71 and Logistic Regression at a lower value. This statistic is very significant since it shows how well the model can distinguish across classes.

The significance of choosing a model according to the features of the dataset and the current predicting task is highlighted by this thorough analysis. Retail transactions are one example of a scenario where the application of ensemble methods, especially stacking, seems to bring significant benefits. These scenarios involve complex and varied data.

4 CONCLUSION AND FUTURE SCOPE

The study's conclusions highlight how machine learning has the potential to revolutionize retail management, particularly in terms of reducing losses from product returns. Our models assist in understanding the underlying customer behaviors that generate these returns in addition to offering useful insights into return likelihoods.

- **Integration of Real-Time Data** Real-time data streams from online retail platforms could be incorporated into dynamic prediction models to allow for real-time updates and faster decision-making insights.
- **Expansion to the Other Retail Sectors** Although this study concentrated on a particular dataset, the approaches utilized here can be extended to other retail sectors, potentially providing broad industry relevance.
- **Advance Ensemble Techniques** Investigating more sophisticated ensemble methods and deep learning models may provide even more profound insights and raise the accuracy of predictions.
- **Sustainability Practices** Future models could incorporate sustainability criteria in light of the environmental impact of returns, assisting merchants in minimizing their environmental impact while also reducing financial losses.

This study provides up opportunities for more advanced analytical tools in the retail industry, with the potential to significantly increase consumer happiness and operational efficiency. These capabilities will probably continue to be improved by machine learning models as they continue to evolve, making them essential tools for contemporary retail management.

REFERENCES

[1] Loyal Guru. 2024. *Understanding Retail Returns: 9 ways to reduce return rates.* https://www.loyal.guru/retail-trends/retail-returns/
[2] NRF. 2022. *2022 Consumer Returns in the Retail Industry.* https://nrf.com/research/2022-consumer-returns-retail-industry
[3] scikit learn. 2024. *scikit-learn Machine Learning in Python.* https://scikit-learn.org/stable/