Enhancing Product Categorization on Amazon through NLP and Machine Learning Techniques

Baraa Zekeria
bzekeria@ucsd.edu

Tiffany Gunawan
tgunawan@ucsd.edu

Neha Sharma
nsharma@ucsd.edu

COGS 109: Modeling and Data Analysis

June 14, 2023

1 Introduction

The classification of products into different categories is a fundamental task in the realm of online retail, as it plays a crucial role in enhancing user experience and improving operational efficiency. However, accurately distinguishing between various product categories poses a significant challenge due to the vast amount of textual data associated with each item. In order to address this challenge and enhance the accuracy of product classification, this project presents a comprehensive analysis utilizing the `Amazon Products Sales Dataset 2023` from `Kaggle` [1]. To focus on the essential aspects of the data, we have selected two key columns (predictors): `name` and `main_category` from the 300,000 observations. The `name` column provides textual descriptions of the products, while the `main_category` column denotes the corresponding category to which each product belongs. The code and data can be found on our GitHub repository.

The objective here is to develop a machine learning-based (ML) product categorization system that can automatically assign products to their respective categories based on their textual descriptions through **Natural Language Processing** (NLP). As such, we propose the following question: *How can ML and NLP techniques be applied to predict the correct product category based on the product name in an Amazon data set?*. Our hypotheses are the following:

- H_1: The performance of the logistic Regression model will significantly improve as more principal components added.

In the following sections, we will describe our data, the methodology and models employed, present the results of our models, discuss their implications, and conclude with the significance of our project.

2 Data

2.1 Data Preparation

In the analysis of imbalanced data sets, where certain categories have a significantly larger number of data points compared to others, stratified random sampling by category is commonly employed. This technique ensures a proportional representation of each category in the sample, promoting a more balanced analysis and modeling process. However, it is important to acknowledge that the smaller sample size may impact the generalizability of the results and introduce some degree of sampling variability.

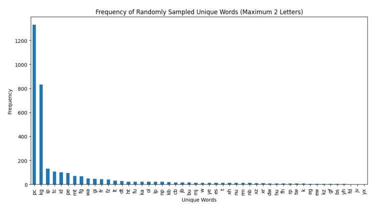
For this project, the original data set consisted of over 300,000 rows and the objective is to reduce the data and create a representative sample for analysis and modeling. Considering resource constraints and the need to balance computational feasibility and sample size, a decision was made to reduce the sample size ranging from about 4,500 per category (some categories have less than that due to low population size) resulting in 77,184 rows in total. While the smaller sample size may not fully represent the entire population, it enables exploration of key category characteristics and addresses the challenges associated with class imbalance.

2.2 Data Cleaning

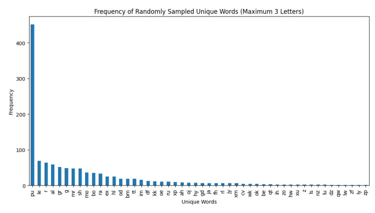
In our data cleaning process, we followed several steps to prepare our data for analysis. Firstly, unnecessary information from the product names, focusing on specific product categories. We also checked for null values and empty strings in the data frame and eliminated them. Next, we performed text cleaning on the product category names, including removing punctuation, words with numbers, and single string characters. We lower-cased all the names for easier analysis and removed stop words (e.g., for, and, the, is, etc.are the columns the predictorsa) that don't contribute much meaning. Additionally, we used **part-of-speech tagging (POS)** to assign grammatical categories to each word. Finally, we applied lemmatization to reduce words to their base form.

2.3 Exploratory Data Analysis

In our exploratory data analysis, we examined and visualized the data by plotting some bar charts to show the frequency of randomly sampled unique words, including words that had a minimum or maximum of either two or three letters. As shown in the charts below, these unique words were a combination of consonant letters, and they had little significance, so we removed them from product names in each category.



(a) 2 Letter Words



(b) 3 Letter Words

Figure 1: Distribution of 50 Randomly Sampled Words

In addition, we applied additional filters to further clean the data. We removed words that did not contain any vowels (a, e, i, o, u, and y) and words that consisted of the same letter repeated three times in a row. These filters helped eliminate words that had little or no contribution to the product names. We also checked for any null values in the dataset and dropped those rows.

Furthermore, during the analysis of the product categories, we noticed that three columns, `Home & Kitchen` and `Home, Kitchen, Pets` and `Pet Supplies`, were similar in nature. To streamline the analysis and avoid redundancy, we examined the most common words in these categories. This examination included identifying keywords related to animal-related names and other terms such as treat, toy, and food. Based on this analysis, we created a pet-related word list. We then investigated the `Home, Kitchen, Pets` category to determine if it contained any pet-related products, given its name. However, we found that this category did not include any pet-related words. Consequently, we merged the `Home, Kitchen, Pets` category with the `Home  Kitchen` category.

Product Categories
Accessories, Appliances, Bags & Luggage, Beauty & Health,
Car & Motorbike, Grocery & Gourmet Foods, Home & Kitchen,
Industrial Supplies, Kids' Fashion,
Men's Clothing, Men's Shoes, Music, Pet Supplies,
Sports & Fitness, Stores, Toys & Baby Products,
TV, Audio & Cameras, Women's Clothing, Women's Shoes

Table 1: List of Product Categories

After finalizing the data cleaning, we created word clouds to give a visually appealing summary of the most frequent words in product names, which helps us identify the main themes and characteristics in each product category.

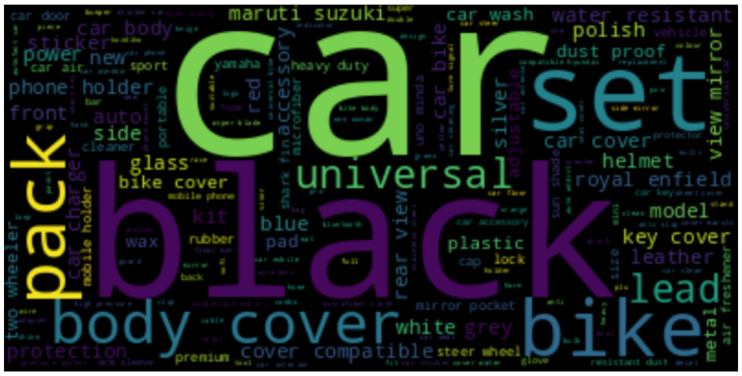


Figure 2: Word Cloud of Product Category: `Cars and Motorbikes`

3 Methodology

We initially considered the **Bag-of-Words (BoW) model** as a potential **feature extraction** method for our task. The BoW model represents text data by creating a set of unique terms (words), disregarding their order, and counting their occurrences in each document (product name). It can be expressed as follows:

$$BoW(d) = (t_1, Count(t_1, d)), (t_2, Count(t_2, d)), \ldots, (t_n, Count(t_n, d))$$

where $t_1, t_2, \ldots, t_n$ are the distinct terms in the document, and $Count(t_i, d)$ represents the number of occurrences of term t_i in document d.

While the BoW model is a simple and widely used approach, it has certain limitations that led us to explore alternative methods. One limitation of the BoW model is that it does not consider the importance of words within a document or in the entire **corpus** [1] (i.e. it treats all words equally).

To address this limitation, we decided to use the **Term Frequency-Inverse Document Frequency (TF-IDF)** as our preferred feature extraction method to numerically represent and analyze the product names in order to classify them into their respective categories.

TF measures the frequency of a term within a document. It is computed as the number of occurrences of the term in the product name divided by the total number of terms in the product name (aka BoW). Mathematically, the TF formula can be represented as:

$$TF(t,\ d) = \frac{Number\ of\ occurrences\ of\ term\ t\ in\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}$$

IDF measures the importance of a term across the entire corpus. It considers how many documents contain the term. IDF is calculated as the logarithm of the total number of documents divided by the number of documents containing the term. The IDF formula can be expressed as:

$$IDF(t,\ D) = \log\left(\frac{Total\ number\ of\ documents\ in\ the\ corpus\ D}{Number\ of\ documents\ containing\ term\ t}\right)$$

The $TF - IDF$ value is obtained by multiplying the TF and IDF values for each term in a product name: $TF - IDF(t,\ d,\ D) = TF(t,\ d) \cdot IDF(t,\ D)$. It assigns higher weights to terms that are frequent within the product name but rare across the entire data set. This captures the relative importance of the features (words) in distinguishing between different product categories.

4 Model

4.1 Model Selection

4.1.1 Model #1: Multinomial Naive Bayes Classifier

To find the most probable category c given a document d (product name), we need to calculate $P(c|d)$, the probability of category c given the document d. In the **Multinomial Naive Bayes Classifier (MNB)**, this can be done by estimating the conditional probability $P(c|d)$ using **Bayes' theorem**:

$$P(c|d) = \frac{P(d|c) \cdot P(c)}{P(d)}$$

[1]**Corpus:** large and structured collection of text documents

where $P(d|c)$ is the probability of product d given category c, $P(c)$ is the prior probability of category c, and $P(d)$ is the probability of product d.

To estimate $P(d|c)$, we consider the features present in product d and calculate their probabilities given category c:

$$P(d|c) = P(f_1|c) \cdot P(f_2|c) \cdot \ldots \cdot P(f_n|c)$$

where $f_1, f_2, \ldots, f_n$ represent the features in product d. In the MNB Classifier, we assume that the features follow a multinomial distribution and are *naively* independent from each other.

To calculate $P(f_i|c)$, the probability of feature f_i given category c, we can use the following formula:

$$P(f_i|c) = \frac{tf - idf(f_i, d) + \alpha}{\sum_{f \in F}(tf - idf(f, d) + \alpha \cdot |F|)}$$

where $tf - idf(f_i, d)$ represents the TF-IDF value of feature f_i in product d. The smoothing parameter α is added to the numerator, ensuring non-zero probabilities for features with zero count. The denominator includes the term $\alpha \cdot |F|$ to normalize the probabilities, where $|F|$ is the cardinality of the set of all features. This normalization guarantees that the probabilities sum up to 1 for each category c and balances the probabilities across all features. By incorporating α and $|F|$, we handle unseen features and achieve well-distributed and normalized probabilities, enhancing the robustness and reliability of the classification results.

4.1.2 Model #2: Logistic Regression

Multinomial Logistic Regression is a technique used to estimate the probability of a document belonging to a specific category. For each class c, the conditional probability $P(c|d)$ of an observation d belonging to that class is calculated using the softmax function. The softmax function ensures that the probabilities for all classes sum up to 1.

$$P(c|d) = \frac{e^{z_c}}{\sum_i e^{z_i}}$$

Here, z_c represents the linear combination of the input features and their respective weights for class c. It can be expressed as:

$$z_c = w_{c0} + w_{c1}f_1 + w_{c2}f_2 + \ldots + w_{cn}f_n$$

where $w_{c0}, w_{c1}, \ldots, w_{cn}$ are the TF-IDF weights assigned to each feature $f_1, f_2, \ldots, f_n$, and n is the number of features.

Multinomial logistic regression was chosen for this project because it is suitable for problems with multiple classes. It estimates separate logistic regression models for each class and compares them to a reference class, predicting the most likely class based on calculated probabilities. This approach provides flexibility and interpretability, allowing for the examination of coefficients and the assessment of model fit and significance.

4.2 Model Training

The **training data** consists of a total of **61,747 rows and 49,140 columns**. Each row represents a product name, and each column corresponds to a unique word in the data set. The MNB and LR classifiers are implemented using the `MultinomialNB` and `LogisticRegression` functions from the `scikit-learn` library, respectively. To train the model, we fitted the models to the training data. The training phase aims to capture the statistical properties of the training data and create a model that can make accurate predictions for unseen product names. After training the model, we evaluated its performance using the test data. The **test data** comprises of **15,437 rows and 49,140 columns**, providing a representative sample to assess the model's ability to generalize and accurately predict the product categories based on unseen product names.

4.2.1 Logistic Regression

To handle the high-dimensional data effectively, we applied **TruncatedSVD (Truncated Singular Value Decomposition)**. This technique served two purposes: capturing the essential patterns and relationships within the data and reducing its dimensionality. By employing TruncatedSVD, we were able to reduce the dimensionality of the data to 1000 components.

After applying TruncatedSVD, we obtained components and singular values. Components represent the directions in the original feature space capturing the most variance, while singular values indicate the importance of each component in capturing variance.

We then transformed the original training using the obtained components, singular values, and their inverses. This transformation further reduced dimensionality, resulting in a new representation of the training data based on selected components.

Finally, we applied the Multinomial Logistic Regression and obtained the report for 3 arbitrary number of components in order to find pa relation between number of principal components (PCs) and model accuracy. We ended up using $k = 300$, `k = 600`, and `k= 900`.

4.3 Model Evaluation

After fitting the MNB and LR on the training and test sets, we now proceed to evaluate the performance of our categorization model.

4.3.1 Multinomial Naive Bayes

While our data set exhibits some level of class imbalance, it is important to note that this imbalance primarily stems from a few specific categories being underrepresented, rather than a pervasive imbalance across all categories. The majority of categories, approximately 15 out of 20, have an equal number of instances, ensuring a relatively balanced representation. However, it is still crucial to account for the underrepresented categories when evaluating our model's performance. Failing to do so may lead to biased assessments and inadequate insights, particularly for those categories with limited instances. In this context, relying solely on conventional **accuracy**, $\frac{TP+TN}{TP+TN+FP+FN}$ [2], [3], [4], [5], can be misleading, as it does not consider the imbalances within specific categories. To address this issue and provide a more comprehensive evaluation, we have chosen to utilize the **F1-score** as our primary metric. The following table presents the evaluation metrics, along with their respective formulas:

Metric	**Formula**	**Description**
Precision	$\frac{TP}{TP+FP}$	Proportion of correctly predicted positive instances
Recall	$\frac{TP}{TP+FN}$	Proportion of actual positive instances correctly predicted
F1-score	$2 \times \frac{Precision \times Recall}{Precision + Recall}$	Harmonic mean of precision and recall

Table 2: Evaluation Metrics

The following table summarizes these evaluation metrics, providing a comprehensive overview of the models' performance on the test data set:

	Precision	**Recall**	**F1-Score**	**Support**
macro avg	0.82	0.78	0.79	15432
weighted avg	0.80	0.80	0.79	15432

Table 3: Evaluation metrics for the Multinomial Naive Bayes classifier

Considering the class imbalance in the data set, we also examined the macro average and weighted average scores. The macro average F1-score was 78%, indicating the overall performance across all categories. The weighted average F1-score, considering the influence of class imbalance, was 79%.

As seen in the code, for individual categories (see `03_feature_engineering.ipynb`), there are variations in precision, recall, and F1-score. Some categories, like `Grocery & Gourmet Foods`, exhibit high precision, recall, and F1-score, indicating successful classification.

4.3.2 Logistic Regression

The evaluation metrics for the logistic regression model are presented below:

[2]**True Positive (TP):** the number of correctly classified positive instances
[3]**True Negative (TN):** the number of correctly classified negative instances
[4]**False Positive (FP):** the number of incorrectly classified negative instances as positive
[5]**False Negative (FN):** the number of incorrectly classified positive instances as negative

	Precision	**Recall**	**F1-Score**	**Support**
Macro Avg	0.77	0.76	0.76	15432
Weighted Avg	0.76	0.76	0.76	15432

Table 4: Evaluation metrics Using `k = 300` PC's

	Precision	**Recall**	**F1-Score**	**Support**
Macro Avg	0.80	0.80	0.80	15432
Weighted Avg	0.79	0.79	0.79	15432

Table 5: Evaluation metrics Using `k = 800` PC's

5 Results

5.1 Model Selection and Estimation

The performance evaluation of our LR and MNB models has yielded interesting findings. Firstly, the MNB model consistently outperforms all LR models (`k = 300`, `k = 600`, and `k = 900`) in terms of macro average. This observation can be attributed to the assumption of independence made by both MNB and macro average metrics.

Secondly, we observed a marginal performance improvement of the LR model at k = 1300, surpassing the MNB model by a slight margin of 1%.

Additionally, we compared the performance of the Logistic Regression model at different levels of complexity. Our results indicate that as the number of principal components increases, the model performance improves. However, it is noteworthy that the performance reaches a peak around `k = 800` principal components and somewhat plateaus thereafter. This observation is visually represented in the graphs below:

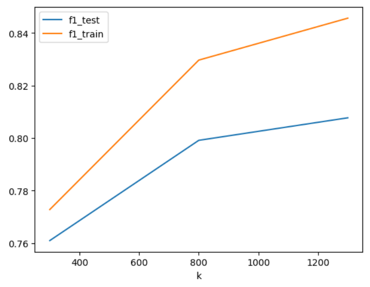


Figure 3: PCs vs. F1 scores

Based on the comprehensive analysis conducted, it can be concluded that the LR model exhibits superior performance for achieving our specific goal. This model stands out due to its consideration of the potential lack of independence among the variables, which aligns with the nature of our problem. By incor-

	Precision	**Recall**	**F1-Score**	**Support**
Macro Avg	0.81	0.80	0.81	15432
Weighted Avg	0.80	0.80	0.80	15432

Table 6: Evaluation metrics Using `k = 1300` PC's

porating such dependencies, LR demonstrates its ability to capture complex relationships and provide more accurate predictions.

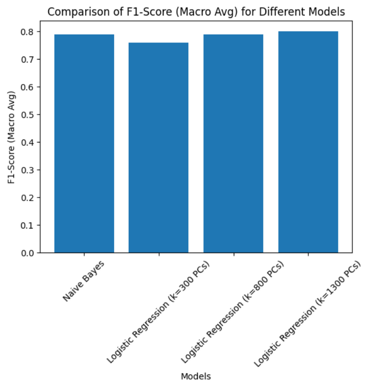


Figure 4: Comparison of F1-Scores between Multiple Models

6 Conclusion

Our model results provide valuable insights and lead to several conclusions. Firstly, the superior performance of the MNB model over LR in terms of macro average can be attributed to their shared assumption of independence. MNB aligns well with the macro average metric due to this inherent similarity in their underlying principles.

Secondly, the LR model's surpassing performance at $k = 1300$ can be attributed to its distinct characteristics. LR does not assume independence and employs a weighted average metric that considers class imbalance. These factors likely contribute to its slightly superior performance compared to MNB in this specific scenario.

Finally, hypothesis can be partially rejected. While increasing the number of principal components in our models positively impacts performance, there is a point of diminishing returns. Beyond a certain threshold, further increments in the number of principal components do not lead to significant improvements in model accuracy.

6.1 Potential Implications & Next Steps

In our project, we encountered several limitations that should be considered. Firstly, the presence of imbalanced classes in our dataset affected the model's ability to accurately predict the minority classes, as the imbalanced nature of the data favored the majority classes. To address this limitation, future steps could involve implementing techniques such as oversampling or undersampling to balance the class distribution. Additionally, exploring ensemble methods like AdaBoost or XGBoost could potentially improve the model's performance on minority classes.

Another limitation we faced was related to the TF-IDF approach. We encountered challenges with words that appeared in the test data but were not present in the training data, potentially leading to inaccurate classifications. However, to further improve the model's performance, future implications include using a larger and more diverse training dataset to capture a broader vocabulary. Additionally, incorporating word embeddings like Word2Vec or GloVe, which capture semantic relationships between words, could provide more robust representations.

The MNB model made a *naive* independence assumption, disregarding complex dependencies between words within a product name. To address this limitation, future implications involve exploring more advanced models like transformer-based models like BERT.

The LR model faced a potential limitation of data leakage. To mitigate this, proper train-test split procedures should be followed, ensuring data transformations and vectorization are performed solely on the training data. Implementing a more robust cross-validation technique can provide a more reliable estimate of the model's performance and ensure its generalization to unseen data.

Determining the optimal value of k in logistic regression was another challenge. Future steps could involve conducting a more extensive hyperparameter search using techniques like grid search or random search. This would allow for a systematic exploration of different k values and improve the model's performance by finding the optimal balance between dimensionality reduction and information preservation.

Finally, the limited size of our training and testing data, representing only a quarter of the original dataset, hinders the model's performance. To address this, future work should focus on utilizing the complete dataset, which offers a more comprehensive representation of product names, categories, and variations. However, this would require sufficient computational resources to handle the increased data volume and ensure accurate model training and evaluation.

Overall, the general implication of our project is the potential for automation and efficiency in the categorization process. By developing accurate and reliable classification models, we can reduce the reliance on manual categorization and streamline the workflow for assigning categories to a large volume of product names. This has the potential to save time and resources for e-commerce platforms like Amazon, allowing them to handle product listings more effectively and improve the overall user experience.

References

[1] Lokesh Parab. Amazon products sales dataset 2023. `https://www.kaggle.com/datasets/lokeshparab/amazon-products-dataset?resource=download&select=Amazon-Products.csv`, 2023.