Predicting Price Ranges of Amazon Products

Group Name: Group A

Group Members:

First name	Last Name	Student number
Sourab	Botu	
Sarala	Sunkara	
Raj	Golakiya	
Chadrick	Clarke	
Olumide	Emmanuel Akinyemi	
Sri Padma Paanidhar	Pala	

Submission date:

Table of Contents

Abstract

This project uses natural language processing (NLP) techniques to analyze customer sentiment from Amazon product reviews to address the problem of declining product sales. The goal is to predict the best pricing tactics to increase an electronics manufacturing and retail company's profitability.

The study uses an organized methodology that includes web scraping for data collection, preprocessing the data for text analysis, and producing models with the help of algorithms for linear and random forest regression. Essential techniques like sentiment scoring and n-gram analysis are used to glean insights from customer reviews.

Both classification models, Random Forest Classifier (TF-IDF vectorizer) and Logistic Regression (TF-IDF vectorizer) had 72% and 79% accuracy, respectively. The results illustrate practical approaches to product design and marketing that consider customer sentiment and preferences. This will not only hedge the current problem of decreased sales but also create a new trend of data-based decision-making in Product Management.

Introduction

Jake's Electronics Manufacturing and Retail produced electronic products for several years by conceptualizing, designing, and manufacturing. The product was priced for retail based on the expenses to make the item and the profit margin desired. However, this model has proved less than efficient in terms of time and resources, and in recent years, the company has seen a 40% decline in sales of its product offerings.

The Chief Executive Officer (CEO) is desirous of finding a solution to the decline in sales and has charged the newly assembled Analytics department manager to find an innovative way of improving product sales and, in turn, the profits earned.

To solve this problem, the analytics team plans to use machine learning to derive insights from consumers' perceived value of products by determining the sentiment from consumer reviews and using this to label the pricing of products in a desired category. This classification model will enable the company to plan, design, and manufacture effectively and efficiently based on the price model, providing the required profit margin.

Amazon's online retail store provides a wealth of product information and consumer reviews, which is an ideal source for the required data. The project comprised 5 key parts: *Data Collection, Data Preprocessing, Data Visualization and Insights, Text Analysis* and *Model production* and *Evaluation*, and *Deployment*. The language of choice for this project was Python.

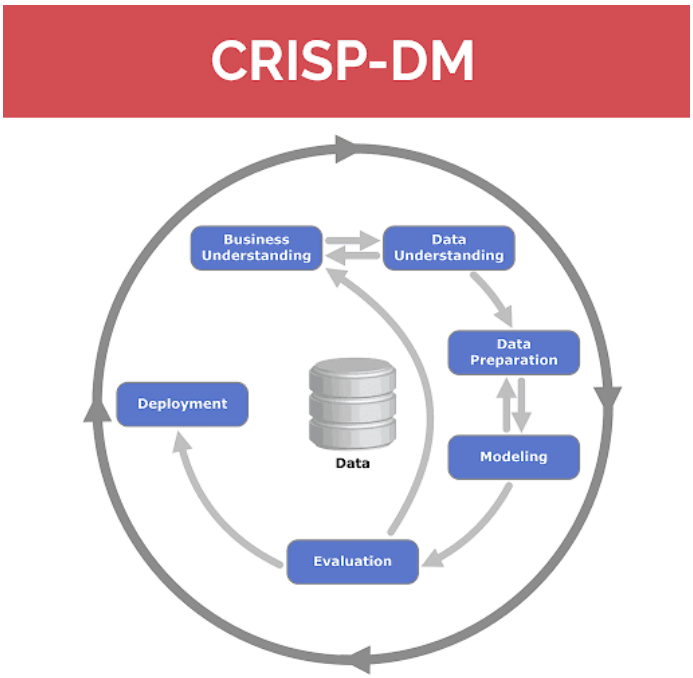
Data Collection

Retrieving the data from the Amazon website required scraping data from the relevant web pages. To achieve this, the Python library Beautiful Soup was used; the library provides functions to enable iteration of the webpages and search for and select desired products. The data collected was transformed into a tabular format and then saved as a *Microsoft Excel Comma Separated File (CSV)*.

We faced a severe challenge in scrapping the data as it was a large amount of data. Amazon asked us to verify as humans by throwing captchas most of the time, and the server blocked us from spamming like a bot but had a workaround to solve the issue.

Methods

The project plan established some critical components required to achieve the goal in ascending order (start to finish). This plan was inspired by the CRISP-DM model for data science projects, which comprises the following key elements: Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment (see Figure 1). The project components are Data Collection, Data Preprocessing, Machine Learning Model and Evaluation, and Deployment.



It was decided that the Amazon online retail store had sufficient data to mine insights and establish a predictive model. The Amazon website allows users to search for a specific product or category (i.e., Samsung 40-inch smart TV or Television). It returns a list of products that exactly or closely match the user's request. The user can also search by categories and subcategories and filter search by several options to find the ideal product.

According to Jungle Scout, there are over 40 categories in the Amazon online retail store, and the project is primarily concerned with the electronics category.

Figure 1. *The CRISP-DM Project Cycle*

Web Scraping

The data was collected using *Selenium (v4.1.0)*, a browser automation toolkit whose primary function was testing a website's behavior (Scrap fly). However, Selenium is more widely known today for its ability to scrape data from modern web pages. For the needs of this project, the following data was scraped from a product page on the Amazon website (www.amazon.com): *Product ID, Product Name, Category, Price, Description, About, Rating, Image -URL, Brand, and Product Reviews.*

A class called *Scrapper* was built using both Selenium and another library package called *Beautiful Soup (v4.9.3),* which consisted of all the appropriate functions required to scrape the data needed.

Scrapper Class Process Flow

1. An instance of the Selenium WebDriver is created, which will be used to access the webpage for the Chrome browser. The Options function was also instantiated, which enables Selenium to run silently in the background and not load all the GUI features of the web pages it is scraping from.
2. The *.get* function enables the user to fetch the desired web page, which is parsed using the *BeautifulSoup* object created. The if-else statement handles scenarios where parsing and fetching the web page raises an error.
3. The remaining portions of the class utilized elements of the previous code to parse information regarding product details. A function was created to encapsulate all these elements.
4. To parse Product ASIN (ID), regular expression was used to pull the data from the URL and to identify the ID on the page; If matched, it would be stored. The product title and brand are retrieved using the *.get* function.
5. Similar code structures and functions are used to parse product price, image URL, product description, and product rating.
6. One of the final steps in the class uses a user-defined function called *ReviewScrapper* to parse customer reviews for each product. The remainder of the class compiles all the information scraped into a *Microsoft Excel comma-separated file (CSV).*

The number of records we collected is 15,819, and 10 features (ID, Name, Category, Price, Descriptions, Rating, ImageUrl, Brand, Review, About)

Data Preprocessing

This portion of the project accounted for 70% of the project's time and resources. Some of the methods applied in this stage include the removal of punctuations and digits, handling words with exaggerated spelling (i.e. gooood → good), splitting sentences into a list of words using *nltk word_tokenize* and removal of commonly used words (i.e., is, the, are). This step proceeds after the data collection, cleaning the entire data set and transforming it into a form acceptable to the chosen machine learning algorithm. This dataset primarily contains attributes with ‘string’ data types, with two (2) containing numeric data (integer and float). Data cleaning required two approaches: firstly, to handle missing values for each attribute and secondly, to prepare the text data found in product reviews and products to allow for sentiment analysis and use inappropriate algorithms.

Data Wrangling

According to (Harvard Business School) data wrangling combines techniques to transform raw data into a usable form. When transforming raw text for use in various machine learning applications, this term is often called *Text Wrangling.* We used the Pandas (v2.2.2) library for data wrangling,

and a Python library was used to manipulate any structure's data. The dataset was examined first for missing values, data inconsistency, and outliers.

Considering how the data was scraped, the dataset would contain several rows of product reviews per product scraped. This results in "duplication" of values in features such as '*id,' 'Name,' 'Category,' 'Price,' 'Description,' 'About,' 'Rating,' 'Image Url,' and 'Brand'* however, it is not an accurate duplication of the entire record(row) as each product review is unique.

Missing values were checked programmatically and through understanding the data. This revealed that the *'Description'* feature had several empty values due to some products' lack of descriptions on Amazon. This was observed during the web scraping process, and a solution to address this was to scrape the information from the *'About'* section of the web page. The Python function '*isnull'* was employed to check the dataset's features for missing values programmatically using *KNNImputer* from the scikit-learn (1.5.0) library for the missing values in the price feature to impute the price with its best replacement value. The dataset had no null values except the 'Description' feature from this check.

The text data within the dataset was further examined, and relevant *Text Wrangling* techniques were applied in the order listed: remove digits and punctuations, contract exaggerated words, tokenization, removal of stop words, and lemmatization. These steps were programmatically handled by creating a class called *'TextWrangler,' which had user-defined functions and libraries (NLTK (v3.8.1), spacy (3.7.5), text blob (0.18.0) word cloud* (v1.9.3)) to execute each of the techniques mentioned above.

Moreover, user-defined functions are loaded with functionalities like removing special characters, removing digits, cleaning the text and removing punctuation, removing HTML tags, reducing every character that is repeated three or more times to just two instances will fix the words, tokenizing the given words into words, lemmatizing using *spacy* library, removing stop words, create bag of words vectorizing the text, and final function in the class, creating a word cloud from the given text using *WordCloud* library.

After cleaning the text, n-grams are generated and analyzed. From the *nltk* library, n-grams are imported and applied to the tokenized text, which is stored in a list for creating visualization to see the most frequent unigram, bigram, and trigram. The insights gleaned from analyzing the n-grams data will be further discussed in *Results*.

Applying vectorization (both Bag of Words and TF-IDF) to the tokenized text and Sentiment Analysis were the final steps of the Data Preprocessing phase. Bag Of Words (BoW) and Term Frequency–Inverse Document Frequency (Tf-IDF) were imported from the Sklearn library. These statistical methods measure how important a term is relative to the document containing it.

To execute the Sentiment Analysis of all the product reviews, *TextBlob* was imported from a Python library bearing the same name; *TextBlob* returns a score ranging from -1 to 1, representing the sentiment score.

Machine Learning Models

Similar to the prior steps in the project life cycle, a class was created to handle the components of the model production. Random forest classifier and Logistic Regression algorithms were selected as the key classification algorithms to build this class.

The dataset scraped has seven out of the ten features with a data type of string, and because of the text data, it will increase the dimensionality of the dataset. Random Forest Regressor is ideal for datasets with high dimensionality and can identify non-linear relationships between features; this is beneficial for text data. Additionally, the results from this algorithm are relatively easy to understand and visualize, suitable for explaining results to executive management.

RegressionModel Class

The class constructed is called ClassificationModel, which facilitates applying *Random Forest Classifier* or *Logistic Regression* on either TF-IDF or BoW vectorizer with a specified number of n-grams. This will allow for efficient running of two models on different input parameters to analyze the impact on model performance.

The class also contained a function that would generate the classification report for each model. The classification report provides several performance metrics that help evaluate the model. The metrics are precision, recall, f1-score, accuracy, and support.

Key Functions

Before entry into the model, the dataset was partitioned into training and test data using the 'Train_Test_Split function from the Sklearn Library. The data was split into a train and test dataset at a ratio of 80:20, respectively.

The parameters for both classification models were tuned using GridSearchCV, which was imported from the Sklearn library. This function takes the parameters selected for tuning and evaluates the model for the best combination of parameter values to yield the best model performance.

Results

Pre-Model Analysis

According to (C. Konowal) n-grams play a valuable role in cutting down the noise of thousands of rows of data on individual search terms. Aggregating text data at various n-gram levels can provide insight into key phrases or words a target audience uses. The results could inform the design of future electronic products and the product's marketing strategy.

A horizontal bar plot was used to visualize the number of times keywords or phrases were used in the about/description or review features. The Ingram plots on cleaned Review text revealed that the following unigrams were frequently used: used, not, work, and good. However, These words do not offer sufficient insight into any common theme within the customer reviews. However, when the

plot for bigrams is examined, terms like *'highly recommended,' 'work great,' 'battery life,' 'do not,'* and *'easy use'* were among the most frequently used.

Though there are some unigrams or bigrams that suggest negation, which could be negative in sentiment when the corresponding word cloud visual is examined words like good, love, work well, great, perfect, use, and one were observed; this presents an impression that consumers generally had good feedback on the products they bought. This also presents an opportunity to explore when critical terms like "battery life" are mentioned in a review, whether that overall review is positive or negative. Further analysis of this nature could provide insight into features of products that customers find satisfying when good or a nuisance when wrong.

WordCloud was used to visualize the words most frequently used in the product's about, description, and review sections. The more frequent a word is directly proportional to the size and color intensity of the word seen in the WordCloud. Notably, the word difference was displayed when the visualizations were created for TF-IFD and BoW vectorizers.

As with the case of ngrams, the frequently used words from the word cloud were use, need, and make, indicating that functionality is the priority among the consumers, and favorable terms like love and good indicate overall satisfaction with the product experience with user, but critical takeaway will be the prevalence of words like camera, speed, etc., which are relevant to products like mobiles and fans. These key terms indicate the general preference for the products. And can become the basis for the marketing strategies.

Post-Model Analysis

Logistic Regression

```
TF-IDF with Logistic Regression - Train Accuracy: 0.8217317487266553
TF-IDF with Logistic Regression - Test Accuracy: 0.7905405405405406
```

Figure 2: *Test and Train accuracy for TF-IDF*

Based on Figure 2, it is observed that the test data's accuracy is less compared to the train data when TF-IDF is selected as a feature to train the model, which says the model is over-fitted.

```
Bag of Words with Logistic Regression - Train Accuracy: 0.99830220713073
Bag of Words with Logistic Regression - Test Accuracy: 0.7162162162162162
```

Figure 3: *Test and Train accuracy for Bag of Words*

The test data's accuracy is much lower than the train data when TF-IDF is selected as a feature to train the model, which says the model is over-fitted.

When compared between TF-IDF and BoW, the model with TF-IDF as a feature should be chosen over BoW.

TF-IDF Logistic Regression - Test Data:

	precision	recall	f1-score	support
low-range	0.79	0.93	0.85	71
mid-range	0.80	0.75	0.77	68
premium	1.00	0.00	0.00	9
accuracy			0.79	148
macro avg	0.86	0.56	0.54	148
weighted avg	0.80	0.79	0.76	148

Figure 4: *Classification report on Test Data for TF-IDF*

The TF-IDF model yields an overall accuracy of 0.79 on the test data. The performance metrics for each class are as follows:

- **Low-range:** Precision is 0.79, recall is 0.93, and f1-score is 0.85, indicating that the model is highly effective in identifying low-range items.
- **Mid-range:** Precision is 0.80, recall is 0.75, and f1-score is 0.77, showing a balanced performance but slightly lower recall than precision.
- **Premium:** The model achieves a precision of 1.00 but a recall of 0.00, resulting in an f1-score of 0.00. This indicates that the model predicts the premium class perfectly but fails to recall any actual premium instances correctly.

Bag of Words Logistic Regression - Test Data:

	precision	recall	f1-score	support
low-range	0.76	0.79	0.77	71
mid-range	0.71	0.72	0.72	68
premium	0.20	0.11	0.14	9
accuracy			0.72	148
macro avg	0.56	0.54	0.54	148
weighted avg	0.70	0.72	0.71	148

Figure 5: *Classification report on Test Data for BoW*

The BoW model has a slightly lower overall accuracy of 0.72. The performance metrics for each class are:

- **Low-range:** Precision is 0.76, recall is 0.79, and f1-score is 0.77, indicating a solid performance but slightly less effective than the TF-IDF model for this class.

- **Mid-range:** Precision is 0.71, recall is 0.72, and f1-score is 0.72, showing a consistent but modest performance.
- **Premium:** Precision is 0.20, recall is 0.11, and f1-score is 0.14, demonstrating that the model struggles significantly with the premium class, similar to the TF-IDF model but with even lower precision and recall.

Random Forest

```
TF-IDF with Random Forest - Train Accuracy: 0.9830220713073005
TF-IDF with Random Forest - Test Accuracy: 0.581081081081081
```

Figure 6: *Test and Train accuracy for TF-IDF*

It is noted from the above figure that the Random Forest is an over-fitted model due to very low accuracy on the test data.

```
Bag of Words with Random Forest - Train Accuracy: 0.9864176570458404
Bag of Words with Random Forest - Test Accuracy: 0.581081081081081
```

Figure 7: *Test and Train accuracy for BoW*

It is a similar response when the model is trained with Bag of Words, and the Random Forest comes out to be over-fitted.

TF-IDF Random Forest - Test Data:

	precision	recall	f1-score	support
low-range	0.59	0.77	0.67	71
mid-range	0.58	0.46	0.51	68
premium	0.00	0.00	1.00	9
accuracy			0.58	148
macro avg	0.39	0.41	0.73	148
weighted avg	0.55	0.58	0.62	148

Figure 8: *Classification report on Test Data for TF-IDF*

The TF-IDF Random Forest model demonstrates an overall accuracy of 0.58 on the test data. The performance metrics for each class are as follows:

- **Low-range:** Precision is 0.59, recall is 0.77, and f1-score is 0.67, indicating a reasonable ability to correctly identify low-range items, with a stronger recall than precision.
- **Mid-range:** Precision is 0.58, recall is 0.46, and f1-score is 0.51, suggesting a balanced but less effective performance compared to the low-range class.

- **Premium:** Precision and recall are both 0.00, resulting in an f1-score of 1.00 due to a mathematical anomaly, highlighting that the model fails to identify any premium instances correctly.

Bag of Words Logistic Regression - Test Data:

	precision	recall	f1-score	support
low-range	0.76	0.79	0.77	71
mid-range	0.71	0.72	0.72	68
premium	0.20	0.11	0.14	9
accuracy			0.72	148
macro avg	0.56	0.54	0.54	148
weighted avg	0.70	0.72	0.71	148

Figure 9: *Classification report on Test Data for BoW*

The BoW Logistic Regression model exhibits a higher overall accuracy of 0.72. The performance metrics for each class are:

- **Low-range:** Precision is 0.76, recall is 0.79, and f1-score is 0.77, showing strong performance in correctly identifying low-range items.
- **Mid-range:** Precision is 0.71, recall is 0.72, and f1-score is 0.72, indicating consistent and reliable performance.
- **Premium:** Precision is 0.20, recall is 0.11, and f1-score is 0.14, demonstrating difficulty correctly identifying premium items, although slightly better than the TF-IDF Random Forest model.

Conclusions and Future Work

In Conclusion, when comparing these models, the TF-IDF Logistic Regression and Bag of Words Logistic Regression models have a clear edge over the rest of the models. The Logistic Regression model with the TF-IDF achieves better precision, recall, and f1-score in low-range and mid-range classes, which means it performs strongly for these particular types of classes. However, similar to the other model, it doesn't seem to predict the premium class correctly, most probably because there is not so much support for the premium class in the dataset. The Bag of Words Logistic Regression model outperforms the TF-IDF Random Forest model regarding overall accuracy and more balanced class-specific performance metrics. The Random Forest model, especially with TF-IDF, shows significantly poorer performance, particularly for the premium class, indicating it may not be suitable for this classification task without further tuning or additional data. Therefore, the Bag of Words Logistic Regression model is recommended for its overall reliability, while further improvements are needed for better handling of the premium class in all models.

The future work includes:

Model Enhancement: More advanced models need to be employed, like gradient boosting, XG boost, support vector machines, and deep learning models like Convolutional neural networks.

Feature expansion: Incorporating more features like customer details and customer demographics from another source with product data present will further help improve the pricing labels' segmenting.

Scalability: Increasing the scope and scale of the project by including a wide range of products instead of only electronics

References

Jungle Scout: Amazon Product Finder & Research Tool - FBA Sales Data Tracker. (2019). Jungle Scout: Amazon Product Research Made Easy. https://www.junglescout.com/

Scrapfly Web Scraping API for developer. (n.d.). Scrapfly.io. Retrieved June 21, 2024, from https://scrapfly.io/

Stobierski, T. (2021, January 19). *Data Wrangling: What It Is & Why It's Important*. Business Insights - Blog. https://online.hbs.edu/blog/post/data-wrangling

Machine Learning Pipelines: Benefits, Challenges, Use Cases. (2023, August 11). Plat.AI. https://plat.ai/blog/machine-learning pipeline/#:~:text=By%20defining%20a%20standard%20sequence%20of%20steps%2C%20pipelines

What Are N-Grams? Use Cases to Improve the Profitability of Your PPC. (n.d.). Www.seerinteractive.com. https://www.seerinteractive.com/insights/what-are-ngrams-and-uses-case

December 6, B. C., & 2023. (2023, December 6). *Top Amazon Product Categories in 2024*. Jungle Scout. https://www.junglescout.com/resources/articles/amazon-product-categories/

Web Scraping with Selenium and Python. (2022, January 10). ScrapFly Blog. https://scrapfly.io/blog/web-scraping-with-selenium-and-python/