Harnessing Helpfulness: Amazon Product Reviews to Elevate Marketplace Experience

Eshita Gupta, Monica Lokare, Sneha Karri, and Veena Ramesh Beknal

Department of Applied Data Science, San Jose University

DATA 270: Data Analytics Process

Dr. Linsey Pang

May 06, 2024

Abstract

In the digital era, online reviews wield significant influence over customer decisions, notably on platforms like Amazon. This project seeks to improve review utility by categorizing them based on product types and predicting their helpfulness through machine learning leveraging classification algorithms to analyze features extracted from review text, vocabulary richness, and metadata, aiming to augment users with insights for informed purchasing decisions. We aim to perform text processing on raw review text features using techniques like TFIDF, Lemmatization, Stopword removal, etc., to generate features. These features will be processed using machine-learning models like Logistic Regression, Random Forest, Support Vector Machine, and XGBoost algorithms offering unique advantages such as simplicity, robustness, and efficiency. We achieved a peak accuracy of 93.75% in Cellphones category and measured success using other metrics like precision, recall, area under the curve (AUC), and F1 score. Our primary objective is to develop predictive models that evaluate review helpfulness and its drivers across various Amazon product categories. We will integrate the predictive models into the review-writing process by providing customers with keywords to enhance review quality. We will also enable sellers to optimize product listings by using critical keywords to cater to customer needs. Project outcomes have broad applications like enhancing consumer access to relevant information, improving Amazon product listings, and enhancing review effectiveness. The project enriches user-generated content efficacy on e-commerce platforms, leading to a better shopping experience for all stakeholders.

Keywords: Amazon, e-commerce, predictive models, review helpfulness, classification algorithms

Introduction

1.1. Project Background and Execute Summary

Reviews have become an essential part of the online shopping experience. Reviews serve as the means for customers to make well-informed decisions on the quality of the product and services offered after the purchasing of the product. In the pre-Internet era, word-of-mouth opinions served as a means of making buying decisions. However, without an opinion, buying decisions largely remained uninformed, leading to dissatisfaction if the product did not serve the intended purpose. This changed with the large-scale internet adoption, especially for buying products online - which we now call e-commerce. E-commerce platforms like Amazon.com allow users to write reviews about the products, and these reviews help customers make informed buying decisions and also help sellers gauge customer feedback on their listed products. Amazon sells different categories of products such as electronics, books, automotive, clothes, etc. Each listing contains metadata and reviews of the other customers. Amazon is the US e-commerce leader, accounting for 38% market share, and reported its annual Net sales as $574.8 billion in 2023, a 12% increase compared with $514.0 billion in 2022. Product reviews have helped replace the earlier opinions and help make buying decisions in online marketplaces. Now, all reviews are not equal and are still technically opinions that are prone to bias and subjectivity. To overcome this, e-commerce platforms like Amazon have included features such as questions and answers and helpfulness voting to help elevate more helpful reviews to customers.

The E-commerce experience is evolving in the digital era, transforming how purchase decisions are made after validating the reviews. There is a shift in the perception and behaviors of customers. Recent research suggests that 77% of shoppers explicitly seek out websites with

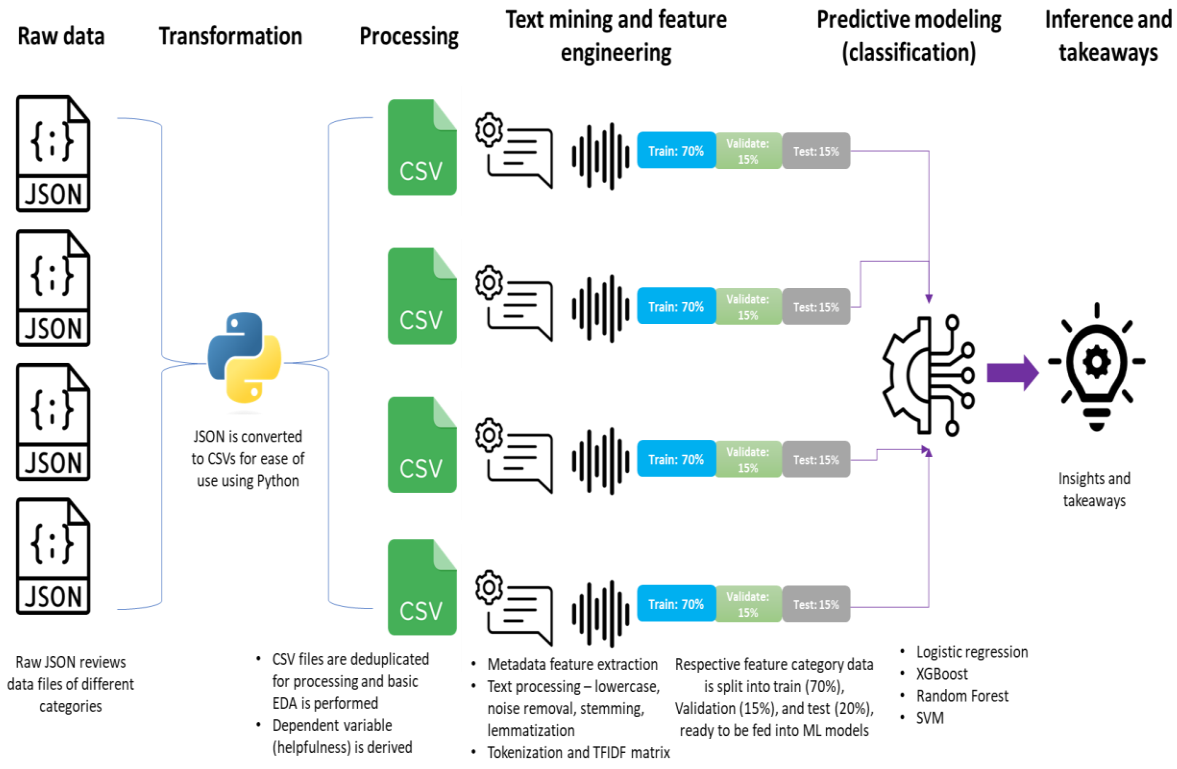
reviews and ratings. The same study mentioned that 98% of surveyed consumers say reviews are their crucial resource for making purchase judgments, a claim attested by Amazon in its official blog. Our research aims to smoothen the decision-making process by evaluating customer insights and understanding what they find most helpful about products based on reviews. In this study, we are attempting to classify the reviews as being helpful or not based on their content, metadata, sentiment, and relevance to the product. While Amazon has universal rules for writing helpful reviews, these are not tailored to categories, which leaves some room for a better customer experience. This study has dual benefits - insights about review helpfulness can benefit both stakeholders in the marketplace, i.e., buyers/customers and sellers. We hypothesize that helpful reviews (voted by customers) are crucial in influencing buying decisions and enabling customers to write more helpful reviews and these can improve their engagement and experience on the platform. Customers can take advantage by being prompted with appropriate words or keywords that result in writing helpful reviews. These insights will also help sellers enhance their product listings, aiding the platform and sellers in redefining their marketing strategies and customer satisfaction.

Our research uses a unique approach to identify the product categories to be studied. We focus on product categories for which the reviews can be considered ‘objective’. To elaborate, we’re consciously avoiding categories for which reviews can have a spectrum of opinions and a lot of subjectivity, like books, fashion, beauty products, etc., The decision to segment categories based on the subjectivity of their reviews was arrived at through heuristic methods. The next step is defining our target variable - a combination of two metrics available from the review data, i.e., helpful votes and verified buyers. Helpful votes are the number of times other customers have answered ‘Yes’ to the question ‘Is this review helpful?’ under the reviews. Verified buyers are

those that Amazon has verified as having bought the products for which they have provided reviews - this helps reduce the instances of fake or malicious reviews and informs the review's integrity. If a review is verified and has a meaningful volume of helpful votes, it will be labeled as 1, i.e., helpful, else 0, implying not helpful. This thorough research process is how we convert our problem into a binary classification problem, with our outcome being to predict review helpfulness with reasonable accuracy and precision. We propose the usage of multiple supervised machine-learning algorithms such as Logistic Regression, Support Vector Machine, XGBoost, and Random Forest. The proposed model workflow diagram is shown in Figure 1 for our project, and we will also perform text processing using techniques like TFIDF, Lemmatization, Stopford removal, etc., on the raw review text to generate features.

Figure 1

Proposed Model Workflow



The insights from our study can influence customers to write more helpful reviews, insights can help augment listing content to suit customer needs. Some of the applications for the project can be automated feedback prompting i.e. customers can be prompted on the platform or post-purchase communication via email to provide helpful feedback based on meaningful signals. We can uncover the underlying needs of customers for critical categories, leading to better recommendations and personalization. Another application could be category-specific recommendations wherein sellers can identify category-specific customer needs based on the insights from this research and improve their listing content/services accordingly. Sellers can address customers' concerns more proactively, thereby improving their ratings, which in turn can lead to enhanced product discoverability and, therefore, sales. Another significant benefit of this study is the ability for sellers to improve their buy-box presence, the key to which is speculated to be influenced by ratings and reviews. From a platform application point of view, Amazon can play a vital role in providing a web-based solution or monthly report published for the sellers to manage feedback provided by the customers, allowing them to access the insights captured and optimize their listing and product strategies.

1.2. Project Requirements

For our project, we select only objective categories such as Appliances, Automotive, Cell Phones and Accessories, and Tools and Home Improvement. In the study project, we will label our target variable i.e., helpfulness, by merging two features from the review data: they are 'votes' and 'verified' - votes refer to the volume of helpful votes assembled by the review, and verified refers to whether the user who posted the review was a verified buyer or not. Our project utilizes TF-IDF, which stands for Term Frequency-Inverse Document Frequency. It calculates how vital a word or term is in a large text corpus. TF-IDF will be used on the review Text feature

from the raw data (the actual review text) to generate features, which is the crux of our modeling efforts. We will also use metadata features about the reviews and the review title.

Our project will integrate the classification machine learning models listed below and evaluate their performance using metrics such as accuracy, precision, recall, receiver operating characteristic curve (ROC), area under the curve (AUC), and F1 Score.

Logistic Regression. is used for classification in supervised machine learning. It predicts the log odds of an event/observation belonging to a specific class. It's a technique in statistics that predicts the probability of a binary outcome based on one or more independent variables. In our research, we aim to use logistic regression to predict whether the log odds of a review are helpful or not, which can then be converted to a probability.

Support Vector Machine (SVM). is a popular supervised machine learning algorithm used for classification when the relationship between independent variables are complex or when two classes aren't linearly separable. SVM classifies or differentiates two data classes by finding the optimal line or hyperplane that maximizes the margin between the closest data points of two classes. This enables the algorithm to find the decision boundary between two classes, which will, in turn, generalize well with the new data and make accurate predictions. The vectors that run through the data points determining the maximum margin are referred to as support vectors. In our project, we aim to use SVM to find the optimal hyperplane that separates reviews between being helpful or not with a maximum margin.

Random Forest. is a tree-based ensemble supervised machine learning algorithm for supervised learning used for regression and classification. In our project scope, we will be using the classification approach. Random Forest is an ensemble model that combines multiple decision trees' output to determine the final output rather than relying on single decision trees.

Bagging is a technique used in random forest that involves training multiple models individually in parallel on random subsets of data and later aggregating their predictions through voting. In our study, we use Random Forest classification to find the optimal separation between helpful or not reviews.

XGBoost (Extreme Gradient Boosting). is efficient and powerful as it uses a boosting technique tree-based ensemble algorithm. It is commonly used for both regression and classification tasks, making it versatile. Boosting is a technique in an ensemble model where the output of each decision tree is input to the next tree, which leads to weak learners getting stronger over multiple iterations. In our study, we use XGBoost classification to find the optimal separation between helpful or not reviews.

The dataset we focused on is a subset of the "Amazon Review Data (2018)" published by Ni (2018), UCSD. The dataset contains a total of 233.3 million reviews and different categories, and these reviews are in the range of May 1996 to Oct 2018. For our project, we will consider the data from 2010-2018 and only the objective categories, including appliances, automotive, cell phones and accessories, and tools and home improvement. There are two sets of data: raw review data, which contains 233.1 million reviews, and a 5-core subset of data in which all the users and items have at least five reviews, which includes 75.26 million reviews. We will be choosing appliance data from the complete review dataset and automotive, cell phones and accessories, tools, and home improvement from the K-score dataset. Table 1 specifies all the features from raw data that will be part of our project.

Table 1

Description of the raw data columns

Column Name	Description
reviewed	ID of the reviewer
asin	ID of the product
reviewer Name	Name of the reviewer
vote	Helpful votes of the review
style	A dictionary of the product metadata
review Text	Text of the review
overall	Rating of the product
summary	Summary of the review or title
unixReviewTime	Time of the review (unix time)
reviewTime	Time of the review (raw)
image	Images that users post after they have received the product

1.3. Project Deliverables

The objective of this project is to conduct a detailed analysis of reviews in the data collection phase and employ a rule-based approach for labeling the data for binary classification of helpfulness across diverse Amazon categories, as defined earlier. We will utilize the review text feature to convert that into multidimensional features using TF-IDF. We will also use several metadata features. For our project, we will use Agile methodology and also combine the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology for project management, with each sprint lasting two weeks, and will accomplish specific deliverables during each sprint spanned over six phases of CRISP-DM. The project execution utilizes Generalized Activity Normalization Timetable (GANTT) and Program Evaluation Review

Technique (PERT) charts and delivers a burndown chart as a deliverable. The Python notebook code components will comprise data extraction, data transformation, data treatment, exploratory data analysis (EDA), data preprocessing and model training, and evaluation metrics as part of the deliverables. Furthermore, the project deliverable will implement multiple supervised machine-learning classification algorithms. We aim to explore in our study the evaluation metrics, including recall, accuracy, F1 score, precision, and area under the curve (AUC) of the receiver operating characteristics (ROC) curve. We will be comparing the models' performance against each other. In addition, a comprehensive research report on the approach used to predict helpfulness utilizing a combination of multiple independent features will be delivered and we're aiming to achieved an accuracy 93.75%. Project deliverables and timelines are shown in Figure 2 in detail. The project also aims to identify critical keywords that will aid sellers in understanding customer needs, ultimately improving the customer experience on the platform and facilitating sellers to enhance their listings on Amazon.

Figure 2

Project Deliverables and Timelines

Phase	Activities	Sprint	Timeline
Project Proposal	Project proposal and Abstract	Sprint 1	Jan-29 to Feb-11
Project Introduction - part 1	Project Background and Execute Summary Project Requirements Project Deliverables	Sprint 2	Feb-12 to Feb-25
Project Introduction - part 2	Technology and Solution Survey Literature Survey of Existing Research	Sprint 3	Feb-26 to Mar-11
Data and Project Management	Data Management Plan Project Development Methodology Business Understanding Data Understanding Data Preparation Modeling Evaluation Deployment Project Organization Plan Project Resource Requirements & Plan Project Schedule	Sprint 4	Mar-12 to Mar-25
Data Engineering - part 1	Data Process Data Collection Data Pre-Processing Data Transformation	Sprint 5	Mar-26 to Apr-09
Data Engineering - part 2 & Model Development - part 1	Data Preparation Data Statistics Data Analytics Results Model Proposals Model Supports	Sprint 6	Apr-10 to Apr-24
Model Development - part 2	Model Comparison and Justification Model Evaluation Methods Confusion Matrix Model Validation and Evaluation	Sprint 7	Apr-25 to May-06

1.4. Technology and Solution Survey

Evaluation of published papers/thesis and research work performed in the area of review classification and analysis provides a nuanced understanding of the different techniques that can be used for our use case. Sara et al. (2019) focuses on classification of mobile phone reviews based on sentiment. A review is classified as positive, negative, or neutral based on how the customer feels about a product. To identify the feeling/sentiment from a set of words, various features are used that help to derive meaningful patterns that can, with a certain degree of accuracy, predict the sentiment. Research conducted by Sara et al. (2019) talks about various techniques that can be used for pre-processing the data, like deduplication, tokenization, removing stop words, Stemming and Lemmatization, lower casing, and punctuation elimination. Once this is done using techniques like BOW (Bag of Words) and TF (Term Frequency), the frequency of keywords is identified. Using unigrams, bigrams, and trigrams with TF and TF-IDF, multiple regression models are trained like Logistic Regression, Naive Bayes, Gradient Descent, and Convolutional Neural Networks (CNN). The author concluded that when the size of n increases in n-gram, the accuracy also increases. With Naive Bayes the accuracy increased from 83.34% to 88.90% when using trigram instead of unigram with TF-IDF. Similar results were observed with Logistic regression, where the accuracy increased from 86.63% with unigrams to 88.90% with trigrams. The length of the review played a vital role in classifying the sentiment and can be a crucial feature to be included for training the models. These findings are in line with the study performed by Haoyang Li. et al. (2017) using a similar methodology on multiple categories of Amazon reviews. They used Naive Bayes as a baseline with 56% accuracy, and their findings led to an accuracy of 79% with the Logistic Regression model. In

comparison, Sara et al. (2019) were able to achieve an accuracy of 86-88% using unigrams and trigrams.

Mahdikhani, M. (2023), on similar lines, analyzed 828,700 Amazon fashion reviews using topic modeling (LDA) and combined them with lexical features like TF-IDF n-grams to predict review helpfulness. Models like Random, Forest, SVM, Logistic regression, and Gradient Boosting were evaluated. Random Forest classification on a combined feature set of Latent Dirichlet Allocation (LDA) topics and TF-IDF weighted n-grams achieving 0.81 accuracy in predicting review helpfulness turned out to be the strongest performing approach. With just topic modeling (LDA), the accuracy with gradient boosting was 0.7, while the other models underperformed with accuracy in the range of 0.56-0.65. With this study, the power of combining lexical features like TF-IDF and Topic modeling with machine learning for intelligent analysis was elegantly demonstrated. Linguistic analysis with a high degree of accuracy allows for extracting important attributes that are regarded as crucial by customers.

Park, Y. J. (2018) suggested treating the review helpfulness prediction as a regression problem by evaluating helpfulness as a ratio of helpful votes to the total votes. His paper primarily focused on categories like beauty, cellphones, clothing, grocery, and video. Independent variables were categorized into psychological, linguistic, and metadata, and the target variable was a prediction of helpfulness. Psychological included analytic (reviews containing analytical thinking), Clout (Reviews containing more professional expressions), Authentic (Reviews containing more personal expressions), CogProc (Ratio of cognitive thinking words), Percept (Ratio of perceptual words), PosEmo (Ratio of positive emotion words), NegEmo (Ratio of negative emotion words). Linguistic features included WC (length of review measured by the number of words in the review text), WPS (level of conciseness of

review, measured by the average number of words per sentence), and Compare (ratio of comparison words). Metadata features consisted of ratings from 1-5. Using different data mining techniques like Support Vector Regression (SVR), M5P, Random Forest, and Linear Regression, he examined the different features influencing the helpfulness of reviews. SVR proved to be the superior model as per the study compared to other models with a low mean absolute error per category. For beauty products, for example, SVR achieved an error rate of 11.7203, as shown in Figure 3 (Park, Y. J. (2018), in predicting review helpfulness that outperformed other modes (M5P: 12.0229, LR: 12.1396, RandF: 12.1729). It is evident from the results that the sentiment and comparative expressions influence reviews only for some product categories. However, the rating, word count, and analytical thinking affect the helpfulness of reviews across all categories. So, the enhanced focus should be on feature modeling and data pre-processing, as that can skew the results significantly for certain product categories.

Figure 3

Average MAEs of employing the data mining methods for each fold (beauty).

Fold	Data #	LR	SVR	M5P	RandF
0	836	12.0820	11.6135	11.7944	11.9215
1	836	12.7120	12.2497	12.4989	12.6919
2	836	13.2242	13.1854	13.1315	13.2427
3	836	12.3903	11.9664	12.3084	12.2743
4	836	11.4204	10.9451	11.4255	11.5257
5	836	11.6658	11.1959	11.4729	11.8278
6	836	12.1031	11.5863	11.9537	12.1718
7	835	11.3936	10.8911	11.2587	11.3696
8	835	11.8692	11.2281	12.0592	12.1587
9	835	12.5354	12.3415	12.3263	12.5452
Average		12.1396	11.7203	12.0229	12.1729
(Std. Dev)		(0.5567)	(0.6862)	(0.5414)	(0.5300)

Note: From the paper "Predicting the helpfulness of online customer reviews across different product types" by Park, Y. J. (2018). *Sustainability, 10(6), 1735*. https://www.mdpi.com/2071-1050/10/6/1735

Nguyen et al. (2020) performed a study on the online review popularity and their influence on customer decisions by combining XGBoost and skip-gram models. Extracted features included textual (sentiment, relevance) and non-textual (reviewer demographics, product details). The models evaluated by this study were XGBoost and Ridge Regression for linear relationship modeling. XGBoost, being an ensemble algorithm that works on gradient-boosted trees, can integrate weak classifiers to achieve strong ones. Parallel computing and overfitting prevention are critical differentiators. Ridge regression is crucial for analyzing datasets in various domains as it can help to model linear relationships between different variables. XGBoost, with around 50,000 predictors, performed significantly better than Ridge Regression when using root mean squared logarithmic error as the metric. XGBoost achieved an RMSLE of 1.168 vs 1.471 as compared to Ridge Regression. The value of non-linear feature combinations and ensemble tree methods for popularity prediction for textual reviews was demonstrated effectively by the accuracy rating for XGBoost. This study shows that the skip-gram and XGBoost approach provides a framework with a high degree of accuracy for using text-based data for predicting review helpfulness.

The research paper by Kim et al. (2006) focuses on the helpfulness of reviews for MP3 players and digital cameras on Amazon. The study primarily evaluated the SVM (Support Vector Regression) model with a variety of feature extraction techniques. The study extracted a varied set of features like star rating and structural attributes like review length, sentence count, average length, syntactic patterns, and formatting. Using TF-IDF, lexical n-gram features like unigrams and bigrams were prepared. Natural language processing techniques like sentence breaking, part-of-speech tagging, parser tokenization, and syntactic parsing were used to extract these diverse features. Using SVM and the extracted features, the study was able to derive promising results

with rank correlations of 0.66. Review length, product rating, and unigrams proved to be the most critical features. They experimented with different kernels within the SVM framework and identified that the Radial Basis Function (RBF) had the best performance. Extensive parameter tuning was performed and was measured using the Spearman correlation coefficient with scores 0.656 +/- 0.033 for the MP3 players dataset and 0.595 +/- 0.028 for the digital cameras dataset, as shown in Figure 4 (Kim et al. 2006). This study shows that SVM performs really well with data with a high number of features and can be a critical model for this use case. Effective feature modeling can help achieve a high degree of accuracy.

Figure 4

Evaluation of the feature combinations that make up our best-performing system.

Feature Combinations	*MP3 Players*		*Digital Cameras*	
	Spearman†	*Pearson*†	*Spearman*†	*Pearson*†
LEN	0.575 ± 0.037	0.391 ± 0.038	0.521 ± 0.029	0.357 ± 0.029
UGR	0.593 ± 0.036	0.398 ± 0.038	0.499 ± 0.025	0.328 ± 0.029
STR1	0.589 ± 0.034	0.326 ± 0.038	0.507 ± 0.029	0.266 ± 0.030
UGR+STR1	0.644 ± 0.033	0.436 ± 0.038	0.490 ± 0.032	0.324 ± 0.032
LEN+UGR	0.582 ± 0.036	0.401 ± 0.038	0.553 ± 0.028	0.394 ± 0.029
LEN+STR1	0.652 ± 0.033	0.470 ± 0.038	0.577 ± 0.029	0.423 ± 0.031
LEN+UGR+STR1	**0.656 ± 0.033**	**0.476 ± 0.038**	**0.595 ± 0.028**	**0.442 ± 0.031**

LEN=*Length*; UGR=*Unigram*; STR=*Stars*

†95% confidence bounds are calculated using 10-fold cross-validation.

Note: From paper "Automatically Assessing Review Helpfulness" by Kim, S. M., Pantel, P., Chklovski, T., & Pennacchiotti, M. (2006, July). In Proceedings of the 2006 Conference on empirical methods in natural language processing (pp. 423-430). https://aclanthology.org/W06-1650.pdf

Based on the technology survey conducted, it is evident that review data can have a lot of variations. The product category, sentiment, and textual length hold a strong significance when it

comes to evaluating machine learning models. It is, therefore, crucial to have a clear understanding of the product categories for which the reviews will be analyzed, as the data cleansing and pre-processing techniques might vary from category to category. We will be performing data cleaning and preprocessing using deduplication, null identification, removing emojis or URLs and non-ASCII characters, and lower-case conversion. Post-cleaning, we will utilize Natural language processing techniques like stemming, lemmatization, BOW (Bag of Words), and TF-IDF for feature extraction, and long tail consolidation using PCA. We will be evaluating multiple algorithms like Logistic Regression, SVM, Random Forest, and XGBoost to predict the helpfulness of the reviews and compare how each model performs as compared to others with similar set of features. We will perform this with multiple categories of reviews and evaluate how each model's performance varies with different product categories.

1.5. Literature Survey of Existing Research

The impact of electronic word-of-mouth (eWOM) on consumer purchase intentions is examined in the paper by Chen et al. (2021). It highlights the functions of professionalism, helpfulness, credibility, and high-quality information in electronic word-of-mouth. Using an ensemble technique and feature filtering algorithms, the study aims at accurately predicting eWOM's impact on consumer's intention to purchase with various machine learning models. The Support Vector Machines show the highest accuracy at 72.17%, among other machine learning models which include Random Forest, Logistic Regression, RepTree, and Multilayer Perceptron. This study helps both consumers and businesses in understanding online reviews better by showing how complex the effect of electronic word of mouth (eWOM) is on what people buy and how effective it is to include different factors in a model that predicts behavior. It gives valuable information to businesses looking to use eWOM to improve their marketing

strategies and also to consumers who are navigating through a lot of information on the internet (pp. 1-19).

The study by Pranckevicius et al. (2017) goes deep with significance on classification of data, particularly analyzing global social networks, detecting antisocial behavior, and understanding sentiments in various domains. It stresses on the role of sentiment analysis in interpreting customer feedback across industries and is focused on sentiment classification using machine learning techniques like Naive Bayes, Random Forest, Decision Tree, Support Vector Machines (SVM), and Logistic Regression. The study also includes natural language processing techniques for text corpus preparation, which includes bag-of-words, segmentation, stemming, term frequency, and word embedding. This paper is focused on executing large-scale data classification tasks with the MLlib library in Apache Spark and also delves into the difficulties of classification based on sentiment. It seeks to determine the most efficient classifier among the above mentioned methods. Using a dataset of 20,000 chosen online reviews, the study highlights the feature extraction from textual content to train the classifiers. Logistic Regression achieves the highest accuracy ranging from 32.43% to 58.50%, while Decision Tree has the lowest accuracy. Although Naive Bayes is little better when compared to Random Forest and Support Vector Machine in average classification accuracy, the difference is not very significant. Additionally, incorporating uni/bi/tri-gram models marginally improve overall classification accuracy but remains insignificant compared to the unigram model. The study concludes by highlighting the effectiveness of Logistic Regression in sentiment-based classification of online product reviews and emphasizes the importance of selecting appropriate classifiers for optimizing sentiment analysis outcomes (pp. 221-232).

Rathor et al. (2018) focused research on using natural language processing analysis to classify and analyze online reviews into positive, neutral, and negative using Sentiment Analysis. Online reviews influence e-commerce customer behavior, and the review system is currently unstructured and unorganized. The study aims to automate the review classification process through Sentiment analysis classification, which has three levels - Document level, Sentence level, and Aspect level. This analysis process is conducted in five steps (p. 1553). Different machine learning classifiers were used to classify Amazon's reviews, such as Support Vector Machines (SVM), Naive Bayes (NB), and Maximum Entropy (ME). The dataset was extracted using the Amazon API. Unigrams and weighted unigrams were used to train the classifiers. A total of 21000 reviews were used in training, of which both positive and negative reviews were 9500 each and 2500 reviews were neutral. To test the model, 3000 randomly selected reviews were considered. The model is then used to compare the classifiers of above mentioned three machine learning techniques using unigram features and weighted unigram features. It was found that the classifiers performed average and comparable for the three techniques for unigrams. However, with the weighted unigrams, the SVM method yielded the highest accuracy (81.2%) (p. 1557). It was concluded that product reviews are essential for both buyers and sellers. Textual product reviews are unorganized, and they need to be classified appropriately into positive, negative, and neutral. User reviews need to be more effective for accurate predictions**.**

The existing study on consumer behavior and sales primarily focuses on summary statistics of reviews, such as ratings, without considering the impact of the sequence of reviews. Kaushik et al. (2018) introduce that besides review statistics and content, the sequence in which these reviews are presented plays a significant role in molding consumer perceptions and their decision-making, highlighting the complexity of online e-commerce transaction behavior. The

proposed framework outlines Message Persuasiveness, Informativeness, and Valence of reviews' helpfulness and variables contributing to online product sales (p. 24). The author has introduced seven hypotheses on product sales for testing the effect of persuasiveness, informativeness, and sequence of review. Hypothesis 1 - Number of helpful reviews that influence positively. Hypothesis 2 - Average informativeness of helpful reviews. Hypothesis 3 - Products with higher average informativeness reviews will likely have lower sales than products with lower average informativeness. Hypothesis 4- Percentage of verified reviews. Hypothesis 5 - Helpfulness Count. Hypothesis 6 - Balance of helpful reviews. Hypothesis 7- Sales are likely to be lower for products with more negative reviews. The data for the testing was collected from Amazon for 57 smartphones over nine months, and the hypotheses proposed using a linear regression model were tested. The model included eight input variables, five control variables, and one dependent variable (p. 26). Model 1 checks how sales are impacted by control variables (p. 27). Model predicts that rating mean, savings offer, and number of reviews affect positively on sales of a product whereas the age of the product and amount of sales have a negative effect. Model 2 helps test hypotheses 1 to 7, and the hypotheses proved to be supported. This study concluded that negative feedback plays a critical role in products that have a large number of negative feedbacks in the first half of reviews which results in less product sales. To avoid cognitive overload, the study suggests reordering less information-dense reviews at the beginning and reorganizing reviews to emphasize positive feedback and mitigate negative reviews.

Chatterjee et al. (2021) studied a Statistics-Based Outlier Detection and Correction Method (SODCM) that targets sentiment analysis algorithms' performance by correcting star ratings without data loss. The proposed research provides methods to eliminate this discrepancy and provide accurate, valuable sentiment analysis, natural language processing, and data

analytics. The SODCM is a new approach that combines statistical and distance-based techniques to detect and correct outliers in Amazon's customer reviews. The research uses Amazon's publicly available datasets collected from 2008 to 2020 of these reviews spanning various products from books, entertainment, electronics, personal care, pharmaceuticals, and health care categories and contains 35000 reviews. SODCM contains two categories: identification of outliers and rectification of identified anomalies. The method has a 6-step analysis process (p. 9). The output of this algorithm has the reviews with the rectified nature of star rating, which means a positively-natured review will have a rating as one, and the negatively-natured review will have -1. SOCDM is further used with datasets and Excel's existing state-of-art discrepancy detection algorithms in accuracy and recall. The research paper findings are instrumental in improving sentiment analysis tools and methodologies, helping businesses and researchers arrive at more accurate insights from customer feedback**.**

Malik et al. (2018) research focuses on e-commerce review helpfulness based on the review contents, detailed analysis and variables of the reviewer. The prediction model uses six machine learning models on three categories of Amazon datasets (Crawled, Multi-domain sentiment analysis, and snap Stanford Amazon dataset). This study identifies review content and reviewer variables. It classifies content variables related to linguistic, psychological, summary language, and text complexity aspects, as well as reviewer variables such as productivity score and helpfulness per day. The significance of the number of syllables in review texts and the productivity scores of reviewers suggests major helpfulness predictors. The study uses several features classified as reviewer variables, visibility variables, readability variables, and linguistic variables. Out of the six machine learning techniques, Stochastic Gradient Boosted was identified as the most effective. The predictive performance is measured in mean-squared error,

root-mean-squared error, and root-relative squared error. As mentioned earlier, for the three datasets, Stochastic GB MSE is as follows: 0.059, 0.069, and 0.078. Adopting Stochastic GB helps in identifying the most significant predictors of review helpfulness**.**

Ireland et al. (2018) studied the importance of data analytics in product design through analysis of sentiment for Amazon products. They considered Amazon product "Coleman Oversized Quad Chair with Cooler" reviews for the study. This study analyzed the top 10 "most helpful" reviews, resulting in 50 reviews being examined for the analyses. This study proposed a framework combining machine learning with natural language processing (NLP) techniques to study reviews. This model highlights the strengths and weaknesses of the product. The machine learning model produced reasonably meaningful results even though the dataset was small. To provide insights to the designers so they could make informed decisions and cater to customers' needs, the qualitative data were converted into quantitative. This framework uses design theory with data analytics. NLP techniques like WordNet and part-of-speech and ML algorithms TFIDF, Support Vector Machine, Naive Bayes Classifier, Maximum Entropy, and Apriori algorithm (pp. 131-134). Word tagging types, division of reviews into sentences, model training to read the sentiment, and generating sentiment pairs were also used for sentiment analysis**.**

The paper by Subhashini et al. (2021) emphasizes on how the natural language processing (NLP) techniques and machine learning algorithms are applied to customer review analysis, which is a crucial area for the companies which are trying to improve their services. There are different kinds of machine learning models which are used, including Neural Networks, Support Vector machines, Decision Trees, and Naive Bayes. The evaluation is done based on their performance in processing and classifying large-scale unstructured feedback from consumer datasets using various platforms. This dataset, which has a lot of reviews and also

related information like ratings, product specifications, and user demographics, which enables a detailed assessment of the performance of the model, accounting for precision, recall, and F1-score criteria. This comparison is essential for determining the pros and cons of each machine learning approach when managing various kinds of customer feedback, from direct positive or negative evaluations to more complex, conflicting opinions. The paper states that most models are supervised learning at 92%; the remaining are unsupervised learning at 5% and semi-supervised learning at 3%, respectively. These could significantly improve company tactics by getting precise understanding of customer sentiment by combining these models. To uncover underlying patterns in consumer feedback, it promotes additional investigation into unsupervised methodologies and deep learning techniques, underscoring the dynamic potential of this study field for well-informed business decision-making and product improvement (pp. 6343-6389).

The study by Poomka et al. (2021) compares the efficacy of machine learning and deep learning techniques in sentiment analysis of Amazon.com book reviews. The dataset comprises 2000 records, that are split evenly between 1000 positive and 1000 negative reviews, with a training set of 70% and a testing set of 30%. Machine learning techniques include Logistic Regression, Naive Bayes, Support Vector Machines (SVM), and Neural Networks with Bag of Words (BoW) preprocessing, and deep learning approaches like word embedding followed by Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models. The BoW with the combination of neural networks achieved the highest accuracy at 82%, with a training time of one-minute. The next best was BoW with logistic regression, reaching a 76% accuracy in under a second of training time. Word embedding with deep learning algorithms stands third, showing 74% accuracy and a two-minute training time. This paper highlights BoW combined with neural networks as a powerful method for improving sentiment analysis accuracy in product reviews.

This study emphasizes the importance of appropriate data processing and model development technique selection in sentiment analysis, showcasing the strengths of machine learning and deep learning in textual data classification for sentiment assessment (pp. 103-109).

The research paper by Ngo-Ye et al. (2012) presents a study on assessing the helpfulness of online reviews using a sophisticated text mining approach. It employs the Regressional ReliefF algorithm, which is an advanced version of the traditional ReliefF algorithm, to predict consumer review helpfulness by identifying key features of reviews. The study combines the Regressional ReliefF algorithm with conventional machine learning models to enhance predictive accuracy by selecting relevant features from online product reviews. The dataset includes text, ratings, and helpfulness scores, which are highly useful among other features. A comparative analysis reveals that Regressional ReliefF significantly boosts the model's effectiveness in predicting review helpfulness. The study concludes by highlighting the potential of this advanced text mining method to improve online review systems for both consumers and retailers by providing users with more informative reviews and offering insights for quality enhancement (pp. 1002-1020).

The paper by Wassana et al. (2021) delves into analyzing Amazon product reviews through machine learning to categorize sentiments as positive, negative, or neutral. This study employs Logistic Regression, Random Forest, and Gradient Boosting Machines on a dataset which consists of 50,000 Amazon reviews. It was found that Gradient Boosting Machines demonstrated superior performance, by comparing these models based on accuracy, precision, F1-score, and recall. The study concludes by highlighting the importance of machine learning in structuring product reviews for better consumer insight and which in turn help with better

business decision-making, highlighting the effectiveness of Gradient Boosting Machines in sentiment analysis among the tested models (pp. 695-703).

The paper by Fan et al. (2019) investigates online product reviews impact on consumer purchasing decisions, by focusing on incorporating product-related information to predict review helpfulness. This approach recognizes that the value of reviews vary, and integrating the product context can enhance prediction accuracy. The study employs machine learning models, which include Linear Regression, Support Vector Machines (SVM), and Deep Neural Networks (DNN), to assess their effectiveness in this predictive task. Among these, DNN models perform better in handling the complexities of review texts and associated product information. The dataset consists of more than 100,000 reviews from a central e-commerce platform, detailed with review texts, metadata, ratings, and product specifications. A thorough comparison of model performances is conducted, examining accuracy, precision, recall, and F1 score metrics. The findings reveal that DNNs excel in predicting review helpfulness, highlighting the advantage of deep learning techniques in analyzing online reviews within the product context. The paper concludes that product-aware analysis significantly boosts the predictive accuracy of review helpfulness models, with deep learning models, especially DNNs, proving to be highly effective. This insight paves the way for future research in improving online review usefulness for consumers and businesses by factoring in product-specific information (pp. 2715-2721).

The paper by Hudgins et al. (2023) focuses on predicting the review helpfulness of Amazon products using natural language processing methods. The study employs machine learning models like Naive Bayes and BERT for analysis and also highlights the importance of removing duplicate reviews, considering review word count, and including lexical elements in analysis to improve quality of review helpfulness. The accuracy is highest for the BERT model

at 83% and next accurate model is Naive Bayes whose accuracy is 78%. The model achieved a good F1-Score of 0.83 in predicting the helpfulness of Amazon reviews, and the fine-tuning of data has improved the F1-Score in both BERT and Naive Bayes models used in the research. This indicates that applying data modification methods consistently enhance the accuracy of helpful review predictions. The study concludes that the BERT model, fine-tuned with strategies such as imputation of word count, removal of duplicates, and lexical enhancements, achieved an F1-Score of 0.83 in predicting helpfulness on Amazon product reviews. The research also suggests implementing features like minimum word count requirements and spam review removal as backend processes to improve the overall user review experience, highlighting the importance of such measures in making reviews more beneficial for consumers (pp. 1-24).

The study by Pranckevicius et al. (2018) finds logistic regression to be the most effective classifier, highlighting the importance of selecting suitable algorithms for optimizing sentiment analysis outcomes in textual content. Whereas, the study by Poomka et al. (2021) compares the efficacy of machine learning and deep learning techniques in sentiment analysis of Amazon.com book reviews, having BoW with the combination of neural networks achieved the highest accuracy at 82%, followed by BoW with logistic regression, reaching a 76% accuracy, and then word embedding with deep learning algorithms showing 74% accuracy. Similarly, Ireland et al. (2018) combined machine learning and neural language processing techniques for sentiment analysis. Rathor et al. (2018) used sentiment analysis, where machine learning classifiers worked the best for the classification of reviews into positive, neutral, and negative sentiment analysis. The study by Chen et al. (2021) emphasizes the functions of helpfulness, credibility, and high-quality information in electronic word-of-mouth, having Support Vector Machines with the highest accuracy at 72.17%, whereas the paper by Fan et al. (2019) focuses on the online product

reviews impact on consumer purchasing decisions, with a novel focus on incorporating product-related information to predict review helpfulness. From Fan et al. (2019) paper, it is understood that DNNs excel in predicting review helpfulness, compared with other machine learning models like Linear Regression, Support Vector Machines (SVM), and Deep Neural Networks (DNN). Chatterjee et al. (2021) used SODCM to enhance sentiment analysis by addressing the discrepancies between star ratings and text sentiments. Malik et al. (2018), the Stochastic Gradient boost was the most effective prediction model. He explained the importance of considering various linguistic, psychological, and text complexity variables for accurate helpfulness prediction. Kaushik et al. (2018) demonstrated how review sequence influences the consumer's perceptions and purchase decisions and proposed seven hypotheses that impact the review characteristics on sales of products. It's evident that SVM provides accurate results over large datasets.

2. Data & Project Management Plan

2.1. Data Management Plan

Amazon is the primary destination for people looking to make online purchases across a wide spectrum of product categories making it the leading e-commerce platform in the United States. Amazon reviews play an integral part in the purchasing decisions made by customers as the listing for similar products can be quite overwhelming. Reviews help customers make informed decisions thereby improving customer confidence while purchasing online. The dataset used in this study was originally compiled and published by Julian McAuley, UCSD (2014). The dataset originated from Amazon.com and spanned multiple years from May 1996 to October 2014. An updated version of the dataset was published by Jianmo Ni, UCSD (2018); it listed the reviews till Oct 2018. The wealth of information available in this dataset is extremely crucial for our research objectives. The dataset includes multiple notable enhancements in addition to the critical features like rating, review text, helpfulness votes, metadata and URLs. It provides more comprehensive coverage across all product categories with a staggering 233.1 million reviews as compared to 142.8 million reviews in the original 2014 dataset. Transaction metadata like product details, size, package type and post-purchase product images for each review are some notable inclusions that have enriched the dataset for various use cases. Jianmo Ni, Jiacheng Li, and Julian McAuley addressed the topic of justifying product recommendations based on product reviews by constructing an annotated dataset and extracting justifications from extensive reviews. This laid a strong foundation for future research studies and their research and the published dataset aligns closely with the requirements of our study.

The Amazon dataset is extremely relevant for predicting review helpfulness. It allows for extensive model training across a wide variety of research use cases. It is made possible due to

the availability of extensive reviews labeled across a plethora of product categories and products. This dataset offers reviews across an extensive range of product categories and products that can cater to a wide variety of research use cases. The annotations included in this dataset provide insights into the different factors influencing review helpfulness and aid in the development of predictive models. This is advantageous for studies focusing on personalized justification generation as it contains annotated data specifically for this task. Access to community resources, tools and support may be readily available as this dataset is widely used across multiple studies on this topic. The continued relevance of product reviews over time aids in the improvement of any existing research done on this area and enhances the development process.

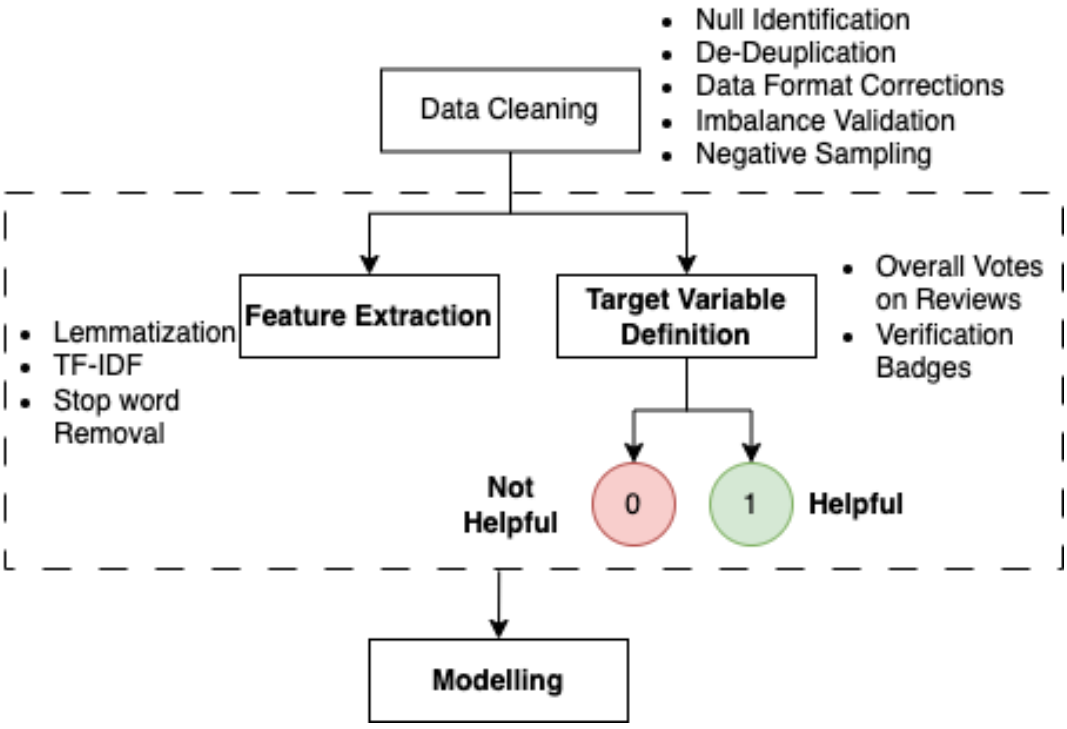
We will focus on data from 2010 to 2018 and restrict our analysis to specific objective categories, namely appliances, automotive, cell phones and accessories, and tools and home improvement. The dataset is available with multiple variations; there is an option of using the complete raw dataset, which is a collection of 233.1 million reviews. There is also an option to use a 5-core subset that is a collection of 75.26 million reviews. In the 5-core dataset, all the users have contributed at least 5 reviews each for each of the items under each category. We will use the raw dataset for the appliance category while K-core dataset for the categories automotive, cell phones & accessories, tools, and home improvement.

We have gigabytes of data that include images, URLs, and textual content with multiple rows and columns. Data management will be extremely crucial here, as we will need to effectively handle and manage large volumes of data. As with any raw dataset, this dataset requires a lot of cleaning and preprocessing to ensure sanity and efficiency when working with different machine-learning algorithms. Feature extraction and modeling are extremely critical on text data to convert it into a machine-readable format. Data quality plays a pivotal role in training

effective machine learning models and as such pre-processing plays an important role in quality assurance.

Figure 5

Data pre-processing flow



Note: Data pre-processing flow illustrating different steps before modeling

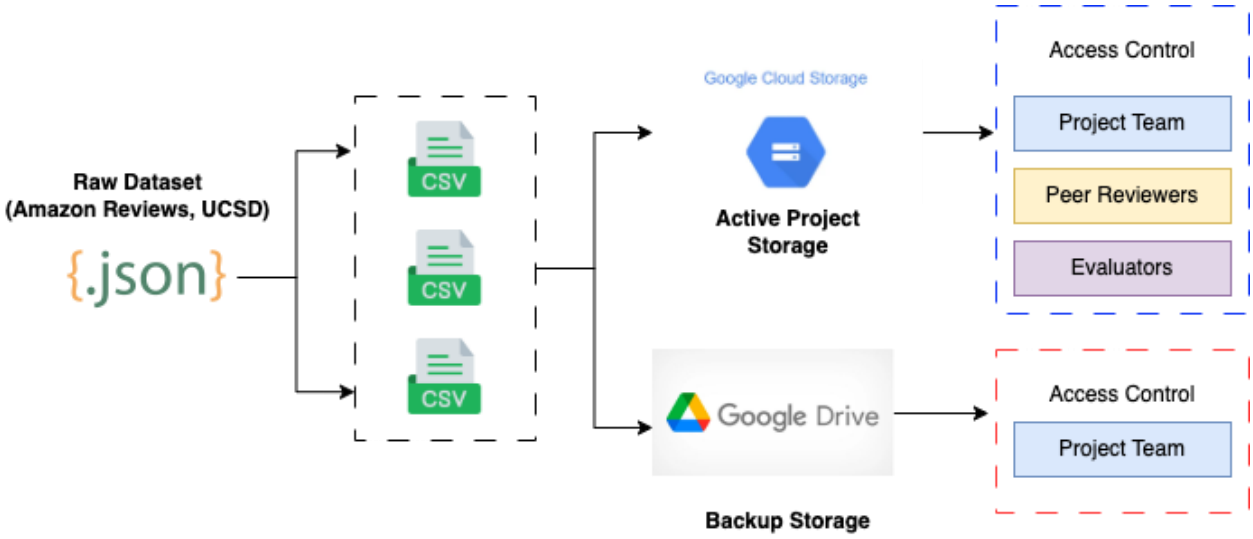
Review helpfulness, our target variable is defined by using overall votes and verification badges on the user reviews. When customers mark a review as helpful, it increments the number of helpful votes on a review, which will be a crucial feature used in our review. Amazon tags reviewers as verified if the reviewer actually purchased the product before writing a review, this increases our confidence in using this for defining our target variable. Reviews that align to both the requirements outlined above are tagged as 1 (Helpful), while others are tagged as 0 (Not Helpful), Refer Figure 5. We will extract features from the review data consisting of review text

and metadata by using processing techniques like TF-IDF, lemmatization, and stopword removal. Using the extracted features, we will train different models for helpfulness prediction.

We will review our dataset for imbalances within the categories selected. After performing deduplication and cleaning, we will review the cleaned data to validate imbalance ratios. A not helpful:helpful ratio of 90:10 or similar ballpark value is desirable for our use case, as perfectly balanced data will defeat the overall objective of predicting review helpfulness. We may employ negative sampling if we encounter a 99:1 ratio or similar skewed ratios.

Figure 6

Data Management Flow



Note: Figure illustrating data management flow, storage solutions and access management

Local processing for such large volumes of data is impractical and inefficient. Amazon dataset has gigabytes worth of data, and as such, usage of cloud storage makes more sense for ease of use and distribution among peer reviewers or evaluators. The raw dataset is available in JSON format, however, we plan to convert it into a CSV format (Refer Figure 6) and store the converted files on a cloud storage solution. We have decided to use the Cloud Storage in Google Cloud Platform for our study as it will facilitate ease of access and retrieval. We will partition the

data by categories we aim to use in our study, this will help us in organizing the data properly and also improve readability. The shared access to the GCS platform will allow individual team members to focus on their project tasks with minimal downtime. We will also create copies of the cleaned dataset on Google Drive and GitHub for backup purposes. Shared access will be maintained at all times across all team members. The access to the backup dataset will be restricted and only project team members will have access. Google Cloud Storage has a 5 GB data limit per month for free tier. However, given our data volume we will extend the same by purchasing required storage.

The Amazon dataset is a publicly available dataset and we will provide the relevant links and information about the metadata in a simple text file in our project repository. The data usage is governed by the rules laid out by the original publishers and usage of this data requires explicit citation for the paper *Justifying recommendations using distantly-labeled reviews and fine-grained aspects (2019), by Jianmo Ni, Jiacheng Li, Julian McAuley*. The data is originally available in JSON format and we will convert them into CSV format for facilitating convenient parsing and pre-processing. The converted data files will be placed on labeled directories on Google Cloud Storage (GCS) and the access will be limited to project team members. On a need basis, the access privileges will be extended for project evaluations and peer reviews using Access Management features under GCS. As the data is publicly available on the original publisher's webpage, we will not be opening access to this data on GCS. The security and sanity of the data will be protected by standard security measures put in place by Google. We will store the data only until the project completion on GCS and post completion the data will be purged as it will not be cost effective to store high volumes of data. We will continue to store the data on our private Google drives for future reference. Any access privileges extended for peer reviews

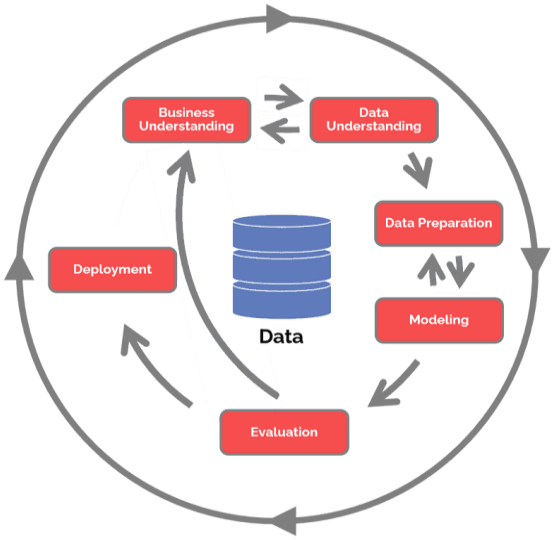
will be revoked from GCS. We will provide proper references and links to the original data set, this will help researchers or students to review the original dataset and train their own models.

2.2. Project Development Methodology

There are two approaches that we will be using for our project - Agile methodology and CRISP-DM (Cross-Industry Standard Process for Data Mining) methodology. Agile is about making quick changes, working together, and improving step by step, making it great for fast-paced projects. CRISP-DM is a step-by-step approach as shown in Figure 7, well-suited for machine learning projects where we work with data, like looking at Amazon reviews to make the shopping experience better. So, we will be using these methodologies for our project. We will segment our project into phases using a work breakdown structure. GANTT and PERT charts will be used to monitor the project and effort timeline. We will monitor the progress and record any bugs throughout testing and development.

Figure 7

CRISP-DM Diagram



Note. This is a CRISP-DM diagram which includes six phases required for our project

Preprocessing and data cleaning, which form the crux of machine learning, involve a number of tasks. Initial data cleaning and exploration, including looking for outliers and missing values, are done in the EDA section. To improve model performance, features are created and transformed using feature engineering. The project uses a variety of machine learning algorithms, which include Random Forest, XGBoost, Support Vector Machine, and Logistic Regression. For data preprocessing, feature engineering, and model development, the project uses a number of Python libraries and frameworks. Our project also utilizes TF-IDF, which stands for Term Frequency-Inverse Document Frequency. It calculates how vital a word or term is in a large text corpus. TF-IDF will be used on the review text feature from the raw data (the actual review text) to generate features, which is the crux of our modeling efforts. We will also use metadata features about the reviews and the review title. Together, these resources help in highlighting the best reviews on Amazon and make shopping easier and more trustworthy for everyone.

Table 2

Responsibility and Resource Allocation for Data Management Plan (DMP)

DMP Phase	Resource	Responsibility
Data collection, documentation, and metadata	Eshita	Download and review Amazon Reviews Dataset (2018) from the provided source
	Veena	Collect detailed Amazon reviews from many products, including what people said, how helpful others found it, the ratings, and if the buy was confirmed
	Sneha	Keep track of how and when we got the data, where it came from, any problems we faced, and how we solved them to make our work clear and repeatable
	Monica	Make and keep details about our data, like what each part means, where it is from, and how we have cleaned it, to help us and others use it better
Ethics and legal compliance	Sneha	Ensure that the team complies with the agreements' terms and conditions and that all software licenses have been obtained
Storage and backup	Monica	Update the files and make backup and storage copies of the information
Data Sharing	Veena	Update the data that can be shared publicly to the GitHub repository
Data preservation	Eshita	Decide which data to keep, and control access as well as the cost of archiving and storing it

2.2.1. Business Understanding

During this phase, we will focus on the project's objective and its deliverables. Our goal is to improve the helpfulness of reviews on e-commerce sites such as Amazon.com where users post product reviews. These reviews assist buyers in making well-informed decisions and also assist sellers in obtaining feedback from customers regarding the products they have listed. With

a 38% market share, Amazon leads the US e-commerce business. In 2023, it generated $574.8 billion in net sales, 12% more than $514.0 billion in 2022. Amazon is an online marketplace that allows merchants to list their goods in a variety of categories and allows customers to purchase these listed goods from the vendors of their choosing. Based on the product reviews, buyers may make well-informed selections about which things to purchase and which sellers to select. For this reason, reviews are important for both buyers and sellers in the marketplace. According to recent studies, 77% of consumers specifically look for websites with ratings and reviews. According to 98% of respondents in the same poll, customer evaluations are an essential tool for them when making decisions about what to buy. Amazon has backed up this assertion on its official blog. Therefore, our goal is to enhance the decision-making process by obtaining the underlying characteristics of helpful reviews. We concentrate on product areas where reviews are deemed 'objective'. To be more specific, we are purposefully ignoring areas like literature, fashion, beauty items, etc., where reviews may contain a wide range of viewpoints and a great deal of subjectivity. This has two advantages: information regarding review helpfulness can be advantageous to buyers/customers and sellers, who are the two main players in the marketplace. Prompting customers with relevant terms or keywords can help them write reviews that are beneficial. By incorporating helpful attributes into their product listings, sellers can improve customer satisfaction and experience. Finally, when buyers and sellers are actively engaged, the marketplace, or Amazon, gets more lively and the experience improves.

2.2.2. Data Understanding

This phase mainly focuses on exploring the data and understanding the dataset thoroughly. As mentioned earlier, we are selecting only those categories whose reviews can be considered as objective. We obtain the raw data in JSON format from the Amazon Reviews

dataset as described earlier for the categories: Appliances, Tools and Home Improvements, Automotive, Cellphone, and Accessories. Then, we convert the raw JSON data into CSV for ease of exploration and analysis. Each category's dataset is deduplicated to remove redundant records and this ensures that the data we are analyzing is unique. For relevance and recency, we chose to only focus on reviews posted between 2010 and 2018 since this captures the experience of customers who purchased products after the mass adoption of the internet and e-commerce. To start with, we analyze different kinds of information like type of data (numeric, characters, nominal, ordinal, dates, etc.), distributions (using box plots, histograms, percentiles, etc.), and look for any patterns or connections. Our goal here is to remove any noise from the data and ensure that we have clean and sanitized data, which will in turn help in building models defined later on. We also keep a close eye out for any issues in the data – occurrences like missing information or duplicates that could negatively influence our analysis. Next, we also look at the distributions of ratings across categories to get a better understanding of the star rating accompanying the review. Furthermore, we can also study the volume of reviews across time to understand time patterns like seasonality which will help in identifying peak buying periods. Once we have a fundamental understanding of the data distribution at a high level, we can begin with the data preparation.

2.2.3. Data Preparation

In the data preparation phase, our first goal is to create the binary dependent variable. The dependent or target variable is a combination of two columns available from the raw review data, i.e., helpful votes and verified buyers. Helpful votes are the number of times other customers have answered 'Yes' to the question 'Is this review helpful?' under the reviews. Verified buyers are those that Amazon has verified as having bought the products for which they have provided

reviews - this helps reduce the instances of fake or malicious reviews and informs the review's integrity. If a review is verified and has a meaningful volume of helpful votes, it will be labeled as 1, i.e., helpful, else 0, implying not helpful. With this, we are able to convert our use case into a binary classification problem, with our outcome being to predict review helpfulness. There are three categories of features that we will be using to predict classification - raw features, metadata features and tokenized review features. The raw features include features such as ratings, the metadata features include length of review text, count of words, unique word count, number of links, etc. and finally the tokenized review features includes text processing using techniques like TF-IDF, Lemmatization, Stopword removal, etc., on the raw review text to generate features across multiple dimensions. We will also perform feature engineering to reduce the occurrence of redundant and correlated features and will also remove features that are sparse or ones that are too homogeneous (i.e. most values are nearly the same). Feature engineering is the most critical part of the data preparation stage as it will directly influence the performance and explainability of the models we're planning on using with these features. To maintain the veracity of the review features, we will keep imputation to a bare minimum. Once we have our analytical dataset ready, the next stage is to reduce dimensions wherever appropriate. Since we're converting review text into a large TF-IDF matrix, we are bound to have a high dimensionality in the data - we will have to make a judgement call on how to treat this. Dimensionality reduction using techniques like PCA, t-SNE, etc. will result in reduced feature volume at the cost of explainability. Once we finetune the features, our dataset is ready for modeling.

2.2.4. Modeling

During the modeling phase, we will build supervised classification machine learning models using various available Python libraries such as numpy, pandas, NLTK, sklearn,

textblob, wordcloud, and xgboost. As explained in the data understanding phase, we do not have labeled data readily available. Thus, we generate the binary labels using a combination of votes, and verified buyers based on predefined rules. This will transform our use case into a classification problem. Once we have the target variable, we will first randomly split the data into training and test sets, with the training set being 70% while the test set is 15% and validation set is 15%. Once the data is split, we can apply various models for classification, such as Logistic Regression, Support Vector Machine (SVM), as well as tree-based ensemble models like Random Forest and XGBoost on the training dataset. It is to be noted that though the name contains regression, Linear Regression is used for classification. While SVM can be a linear classifier, the kernel for SVM will be decided based on the outcomes of hyperparameter tuning.

Our approach involves building baseline models for all the algorithms for each of the chosen product categories. The baseline models will be built with default parameters to establish an initial benchmark, which will be improved in the next stage i.e. fine-tuned models. We will employ various methods like grid search to find the optimal hyperparameters for each model. Furthermore, we will fine-tune the model based on the optimal hyperparameters obtained. Once we have our models, we can use our trained models to predict the target variable on the test data – this is done since we can compare the predicted and actual values of the target variable in the test dataset, which will help us compute the confusion matrix and various model performance metrics called accuracy, precision, recall, etc. Next, we will compare the results of the baseline and tuned model and analyze model performance improvements. This will help us understand the incremental improvement in model performance enabled by using optimal parameters.

Every individual in our project will implement at least one model for all four selected objective product categories. We anticipate the XGBoost model will perform better than all the

other models since it is an ensemble modeling technique that uses boosting to improve performance while reducing the possibility of overfitting. Finally, the best model will be chosen for each product category and analyzed, and the results will be compared with all the other models for each category.

2.2.5. Evaluation

Our project has the following evaluation metrics: Receiver Operating Characteristics curve (ROC), Area Under the Curve (AUC), Accuracy, Precision, Recall, and confusion matrix. For a binary classifier, ROC is a graph that acts as a performance indicator. Every point on the curve is a specific decision threshold with a respective true positive rate and false positive rate. The area under the curve captures the area under the ROC. Confusion matrix is a visualization in the form of the actual labels versus the model's predictions. Other metrics that will be used in our project to evaluate the performance of the model are accuracy, precision, recall and F1 scores; these are the values that can be obtained from the confusion matrix. Accuracy is a measure of the overall correctness of the model. Recall is a true positive rate. Precision is a positive predicted value. F1 score is the harmonic mean of recall and precision which gives us a general

These are key measures of classification. Along with this, we will also calculate the feature importance, which is a rank ordering of each feature based on its predictive power. Feature importance can be computed only for Logistic Regression, Random Forest and XGBoost. In Logistic Regression, the dependent variable for the model would be the log-odds of helpfulness, and hence the coefficients of the features can be interpreted as their importance. In tree-based models like Random Forest and XGBoost, we can obtain a relative importance of variables using Gini importance. Our project has five phases for modeling: computing the performance of each model, comparing each model based on performance, selecting the

best model by category, feature importance, and insights.

2.2.6. Deployment

In the deployment phase, we will take critical steps to implement our project. We will begin by uploading all the code, designed to evaluate the helpfulness of Amazon reviews, into a private GitHub repository. This will enable us to document our work in a centralized manner and have further enhancements and collaboration by team members. Once our code repository is in place, we will test it in a simulated test environment to ensure it integrates smoothly without causing any disruptions. Following testing, we will document the performance of the codebase and conduct a demonstration of the final models. We will monitor the deployment to quickly address any issues that arise and to fine-tune the model based on user feedback. This is a phased implementation for ongoing checks and enhancements which makes sure our project performs smoothly. By planning carefully and executing adaptively, we aim to deliver an enhanced shopping experience for customers and sellers that values and highlights the most helpful product reviews.

2.3. Project Organization Plan

The Cross-Industry Standard Process for Data Mining (CRISP-DM) used in our project is a highly beneficial framework focusing on improving Amazon's review experience. As shown in Figure 7 is the Work Breakdown Structure (WBS) of our project. In the first phase, business requirements, we will define the project scope and deliverables. We have also conducted literature surveys on previous research papers to understand the approaches and their underlying models and methodologies used in similar use cases. In order to find the appropriate dataset, we gather all the data sources for our dataset collection along with the necessary tools required to carry out our project. We create a project plan to execute the project, which establishes the

project's purpose, deliverables, and key metrics that can be used to evaluate the project parameters.

In data understanding, our aim is to examine and understand our Amazon dataset thoroughly. The four product categories are selected and then converted from raw JSON to CSV using Python. For each category, we perform extract transform load. We then explore the data to understand the feature distribution, variable types, data types, and their dependencies. We also check for data quality issues and define ways to address these issues in order to prepare data for the next phase.

The third phase is the crucial phase, where we make our data ready for predictive modeling. Tasks include cleaning the data by removing duplicate records, standardizing, representing missing data, and creating a dependent variable based on votes and verified features. We perform feature engineering on raw data which includes reviewing metadata, extracting useful features from text, and creating graphs for data understanding. The TF-IDF vectorizer is created so that our dataset, i.e., a particular category, is ready for modeling.

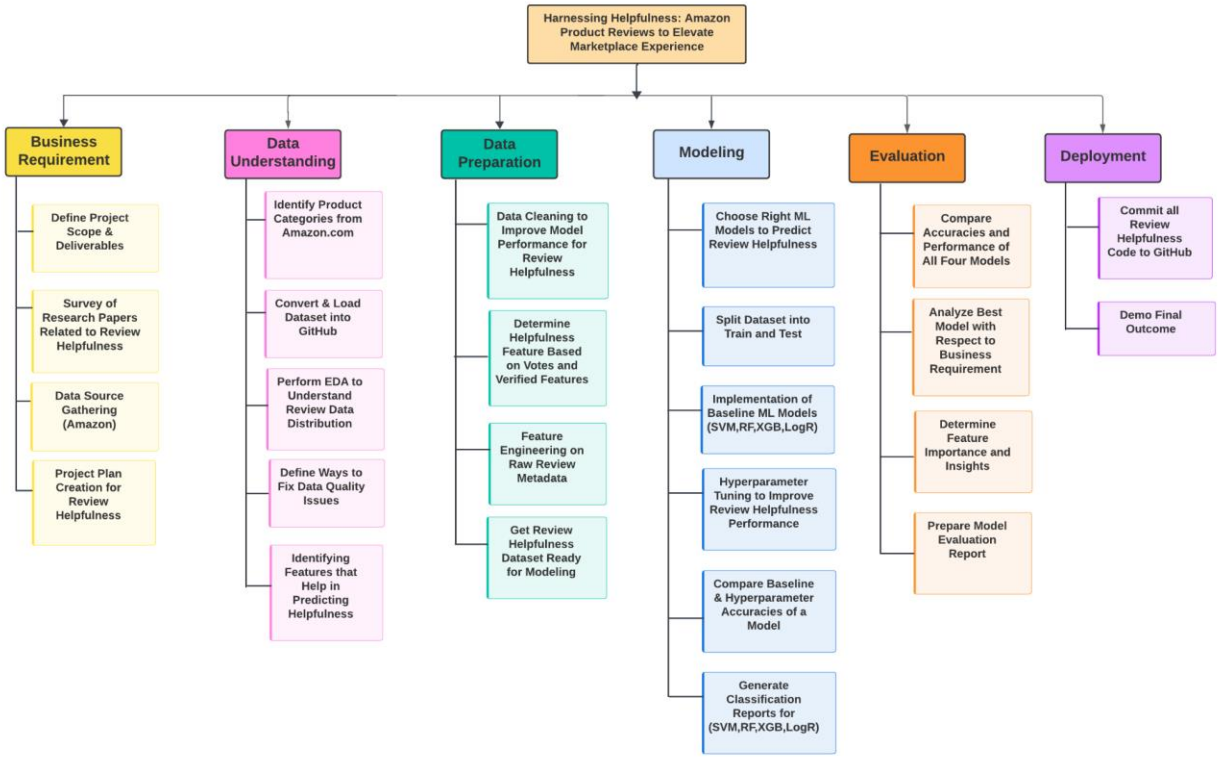
In the Modeling phase, we choose appropriate machine learning techniques and models to predict review helpfulness. We split the dataset into test and train data, which will implement the models by importing the required libraries. We calculate accuracy and other relevant evaluation metrics and display classification reports. Everybody in the team is responsible for a model of each product category of the Amazon dataset. We will find optimal hyperparameters and perform tuning to improve the performance of our models.

In the evaluation phase, we analyze each machine learning model for the same category and compare its accuracies for all classification models. Highest-accuracy model is identified along with the justification for choosing it.

In the final deployment phase, the codes are pushed to a version control system like GitHub, and we present a demo of our final results, findings, and project methodology.

Figure 8

Work Breakdown Structure for Harnessing Helpfulness for Amazon Product Review.



2.4. Project Resource Requirements and Plan

To implement our project, we will use the Python Jupyter Notebook in combination with visualization libraries such as Matplotlib and Seaborn. We will need a Windows machine with 16 GB RAM and a 64-bit processor for this project. The raw project files will be stored in GitHub and Google Drive, while Cloud storage by Google Cloud Platform can be utilized to store the data after converting the raw JSON to CSV. Specifically, we will utilize Python Jupyter Notebook version 6.5.4 for data preprocessing, cleaning, labeling data, feature engineering, and modeling. In addition, we aim to create a dashboard using a visualization tool

i.e., looker in Google Cloud Platform. We will also perform text tokenization, lemmatization, and classification reports for our dataset. Furthermore, our project will handle the entire data from data preprocessing, data cleaning, and predictive modeling, which includes training, hyperparameter tuning, and optimization. We will use the HPC lab refer to Table 3 for complete hardware requirements for our project. Table 4 below shows the software requirements for our project we will utilize Machine learning frameworks like sklearn, wordcloud, nltk, and xgboost libraries. Additionally, we will use MS Office Professional Plus, Draw.io, GitHub, ClickUp, and Lucid software. We will be using various resources locally and cloud services refer to below Table 5 which shows our project's resources and cost estimation in detail. We have estimated a total cost of $1000 for our project.

Table 3

Project Hardware Requirements

Hardware	Configuration	Purpose
Cloud Storage	2 GB	Store the CSV files
Google Drive	2 GB	Store the CSV files, Raw JSON Files
HPC Lab	AMD Ryzen 9 7950X (CPUs:16, RAM:128)	Model training
Local Windows Machine	16 GB RAM, 64-bit processor	Preprocessing, Model Development and Testing

Note. This hardware is specific for our project

Table 4

Project Software Requirements

Resource	Version	Purpose
Python Jupyter Notebook	6.5.4	Project Development
NumPy	1.24.3	Preprocessing, cleaning
Pandas	2.0.3	Preprocessing, cleaning
scikit-learn or sklearn	1.3.0	Model Development
nltk	3.8.1	Preprocessing, cleaning
wordcloud	1.9.3	Preprocessing
xgboost	2.0.2	Model Development
GitHub	--	Project code management
ClickUp	--	Project Management Tool
MSOfficeProfessional Plus	2021	Creating and editing reports
Draw.io	--	Creating Flow diagrams
Lucid Chart	--	Creating a work breakdown structure

Note. This configuration is specific for our project.

Table 5

Resources and Cost Estimation

Utility	Resource Type	Tool/Application	Duration	Cost Estimation
Data Storage	Hardware	Cloud Storage	2 months	Free (Student Credits)
Data Preprocessing	Software	Python Jupyter notebook	2 months	Free
Machine Learning Frameworks	Software	sklearn, wordcloud, nltk, xgboost	2 months	Free
GitHub	Software	Data Files and Project Files	2 months	Free
Local Machine	Hardware	64-bit Version	2 months	$1000
Visualization Tool	Software	Python Jupyter notebook	2 months	Free

Note. All resources, cost estimations, and cloud services are as per project requirements.

2.5. Project Schedule

We are using a Gantt chart to track tasks and the status of the project, which is displayed in the form of a bar chart. We wanted the flexibility of moving back and forth between phases, due to which we've structured our Gantt chart to match the CRISP-DM framework, along with an agile methodology. Each phase is depicted vertically, ensuring progression towards deliverables at the end of each phase. Using this approach, we can understand the task dependencies, duration, and individual responsibility. The x-axis represents the project's timeline, and the y axis represents the tasks and subtasks. We read the graph from left to right to keep track of the task's status. We have broken down each phase into multiple sprints, which last 2 weeks, and will be delivered at the end of each sprint. Each sprint has tasks and sub tasks allocated to group/individual team members. There are seven sprints in total. Individuals will be working on tasks assigned every week, including weekends. A subtask's duration is estimated by the difficulty level and amount of work involved. At any point, we can go back to the previous phase /tasks/ subtasks to change or modify the existing information. After each sprint, we have a retrospective meeting to understand the challenges faced and areas of improvement for upcoming sprints.

In the business understanding as shown below in a Gantt chart Figure 8**,** we have six tasks, i.e., define project scope & deliverables, preparation of project abstract report, survey of research papers related to review helpfulness, data source, project plan creation for review helpfulness and create project introduction report. It took a sprint to define the scope of the project, create an abstract for the use case, and gather data sources. In addition, the research survey, project plan, and introduction report were also completed within two weeks. The creation of the introduction report has dependencies on four tasks which makes it one of the most

important milestones of the project. The details of tasks and subtasks can be seen in a Gantt chart below. The business understanding phase took two sprints to complete, during which the abstract and introduction reports were presented. Table 6 gives the details of tasks and their dependencies.

Figure 9

Business Understanding Gantt Chart

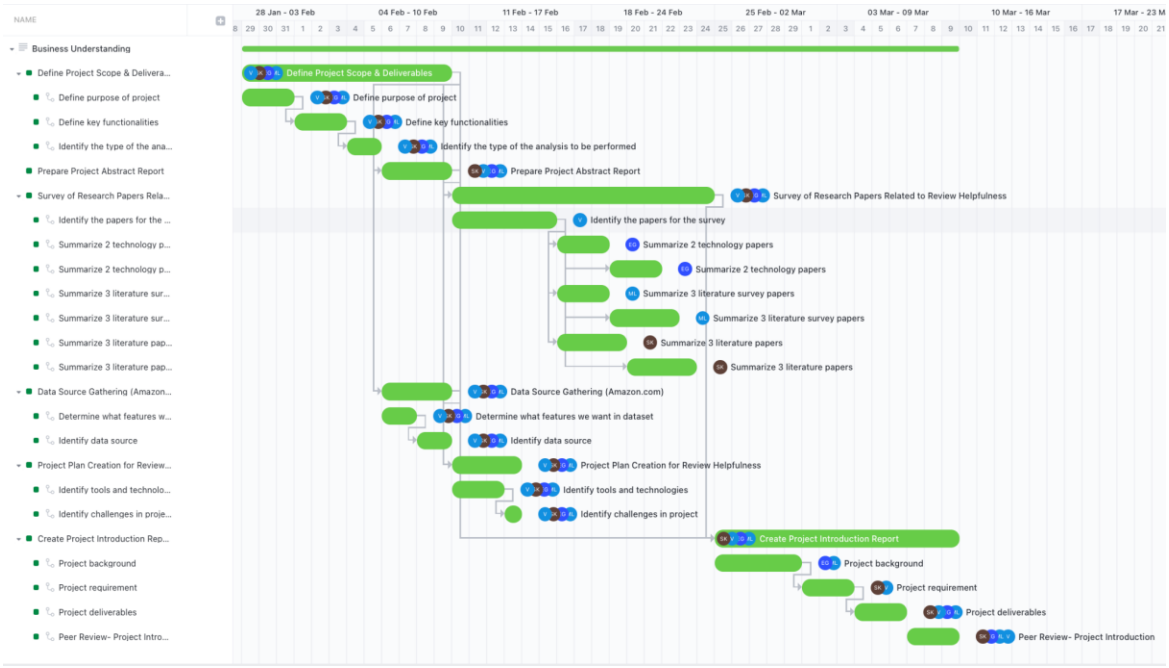


Table 6

Task List for Business Requirement

Task Number	Task Name	Depends On	Start Date	End date	Duration
TS1	Define Project Scope & Deliverables	--	Jan 29	Feb 9	12
TS2	Prepare Project Abstract Report	TS1	Feb 6	Feb 9	4
TS3	Survey of Research Papers Related to Review Helpfulness	TS2	Feb 10	Feb 24	15
TS4	Data Source Gathering (Amazon)	TS1	Feb 6	Feb 9	4
TS5	Project Plan Creation for Review Helpfulness	TS4	Feb 10	Feb 14	5
TS6	Create Project Introduction Report	TS1, TS2, TS3	Feb 25	Mar 9	14

Sprint three mainly focuses on understanding how data is distributed in the dataset for each product category as shown in Figure 9, the Gantt chart of data understanding which consists of six tasks. The Sprint starts from March 12 to March 23rd. Most tasks and subtasks are completed by collaborating with the entire team. We start by identifying the product categories that aren't subjective in the Amazon dataset. We convert this data from JSON to CSV and store it in the GitHub repository. Perform data exploration, i.e., identify the missing values, duplicates, outliers, and null values. We will label the dependent variable that is helpfulness and also identify features that would help in predicting helpfulness. Because the dataset is large, there are data quality issues that need more attention, and deciding how to resolve them is critical. Work Breakdown Structure, Gantt, and PERT charts are prepared as part of the data deliverables and project management plan report. Table 7 gives the details of data understanding tasks and their dependencies.

Figure 10

Data Understanding Gantt Chart

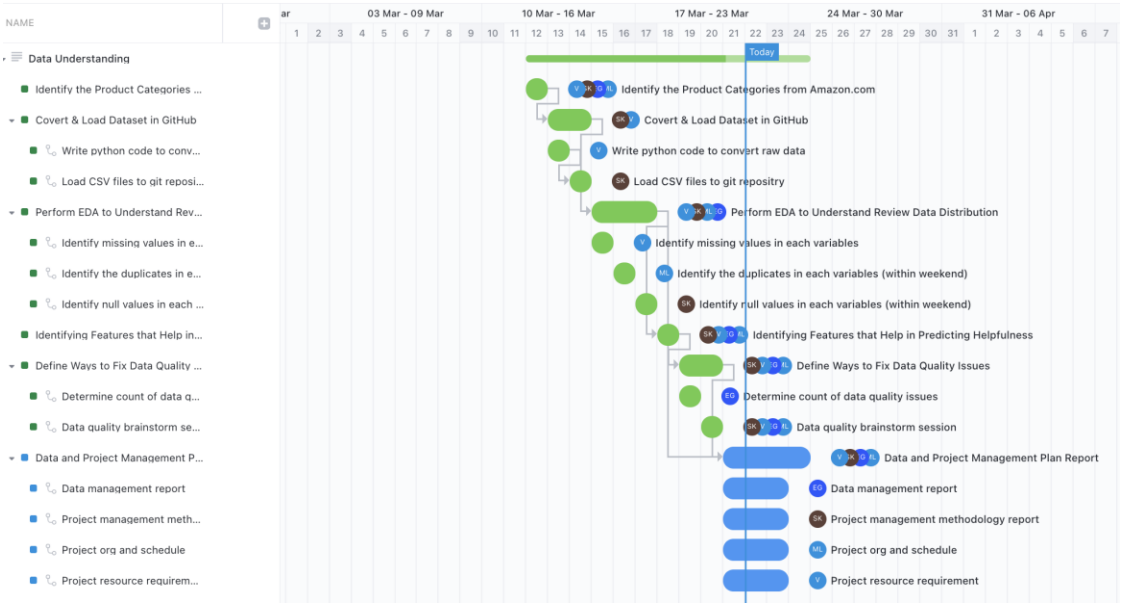


Table 7

Task List for Data Understanding

Task Number	Task Name	Depends On	Start Date	End date	Duration
TS1	Identify the Product Categories from Amazon	--	Mar 12	Mar 12	1
TS2	Convert & Load Dataset in GitHub	TS1	Mar 13	Mar 14	1
TS3	Perform EDA to Understand Review Data Distribution	TS2	Mar 15	Mar 17	3
TS4	Identifying Features that Help in Predicting Helpfulness	TS3	Mar 18	Mar 18	1
TS5	Define Ways to Fix Data Quality Issues	TS4	Mar 19	Mar 20	2
TS6	Data and Project Management Plan Report	TS3, TS5	Mar 21	Mar 24	4

Data preparation is a critical phase, pivotal for achieving high accuracy. Sprint five is a critical and most important task in this sprint, as cleaning and preprocessing data that lays the foundation for successful predictive modeling. This phase comprises numerous subtasks, primarily focused on cleaning techniques, such as lemmatization, stop words, and tokenization are applied to extract meaningful features from review/text metadata. Histograms and box plots are plotted to understand the distribution of features after cleaning and preprocessing the dataset. Here each task is dependent on its prior task. This phase involves both individual and team collaboration. In this phase, we are preparing the dataset for applying machine learning models. Each piece of work is peer-reviewed and merged into one file. Figure 10 shows details of the timeline, tasks, subtasks, and dependencies of data preparation through the Gantt Chart. Table 8 gives the details of data preparation tasks and their dependencies.

Figure 11

Data Preparation Gantt Chart

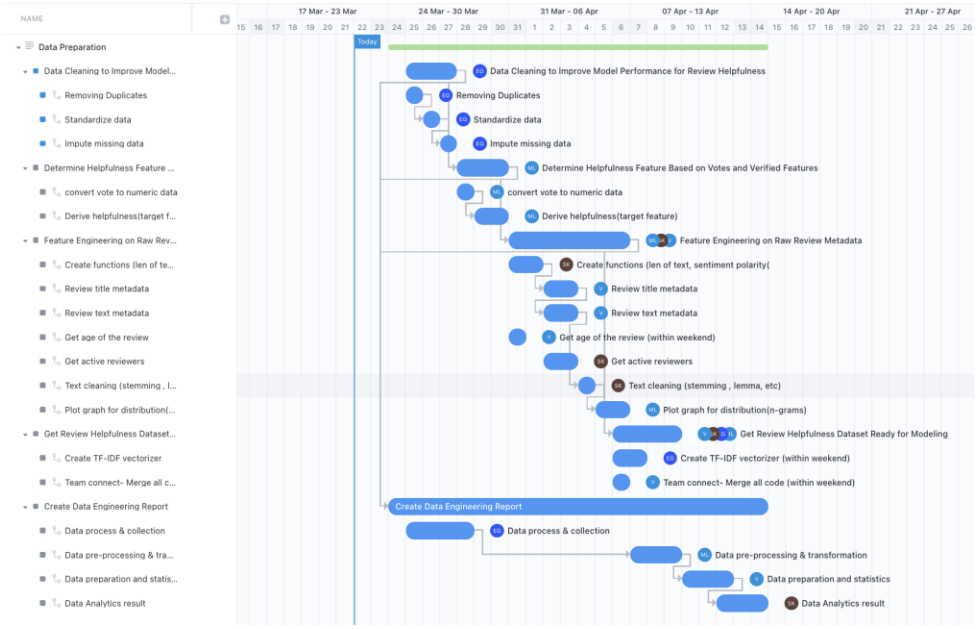


Table 8

Task List for Data Preparation

Task Number	Task Name	Depends On	Start Date	End date	Duration
TS1	Data Cleaning to Improve Model Performance for Review Helpfulness	--	Mar 25	Mar 27	3
TS2	Determine Helpfulness Feature Based on Votes and Verified Features	TS1	Mar 28	Mar 30	3
TS3	Feature Engineering on Raw Review Metadata	TS2	Mar 31	Apr 6	7
TS4	Get Review Helpfulness Dataset ready for Modeling	TS3	Apr 6	Apr 7	2
TS5	Create Data Engineering Report	TS1, TS2, TS3, TS4	Mar 24	Apr 14	22

During the modeling phase, as shown in Figure 11, we apply machine learning techniques to predict review helpfulness by selecting appropriate models. The dataset is divided into training and testing subsets, and we apply four classification machine learning algorithms i.e., Logistic Regression, Support Vector Machine, Random Forest, and XGBoost, to the objective Amazon categories. We also perform hyperparameter tuning to improve the performance of the model. Once we have identified the hyperparameters, we will build a tuned model, compare the performance between the baseline and tuned model, and iterate if necessary. Each individual member of the team will build one model for all four categories, and the best-performing model for each category will be chosen based on the model's performance. Table 9 gives the details of modeling tasks and their dependencies. In this phase, we also write the individual report on each model. Most of the tasks are individual tasks, as each team member will be responsible for one model for all four categories of the Amazon dataset.

Figure 12

Modeling Gantt Chart

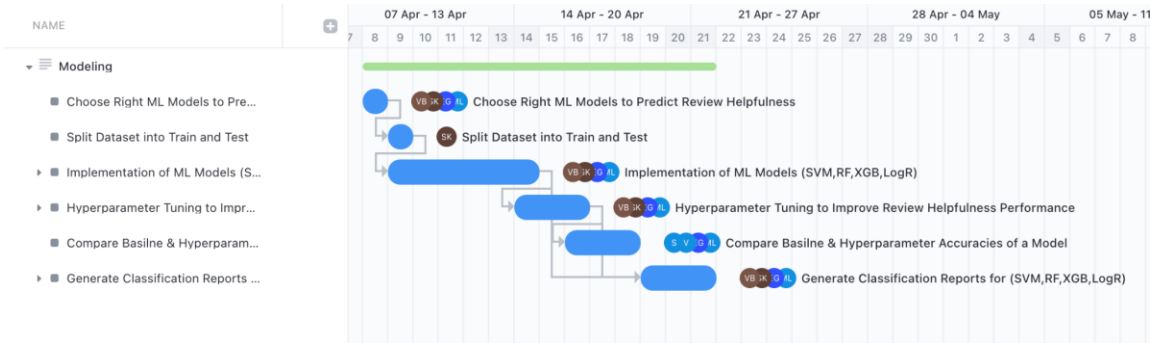


Table 9

Task List for Modeling

Task Number	Task Name	Depends On	Start Date	End date	Duration
TS1	Choose Right ML Models to Predict Review Helpfulness	--	Apr 8	Apr 8	1
TS2	Split Dataset into Train and Test	TS1	Apr 9	Apr 9	1
TS3	Implementation of ML Models (SVM,RF,XGB,LogR)	TS2	Apr 9	Apr 14	6
TS4	Hyperparameter Tuning to Improve review Helpfulness Performance	TS2, TS3	Apr 14	Apr 16	3
TS5	Compare Baseline & Hyperparameter Accuracies of a Model	TS4	Apr 16	Apr 18	3
TS6	Generate Classification Reports for (SVM,RF,XGB,LogR)	TS3, TS4, TS5	Apr 16	Apr 21	6

Evaluation and deployment are scheduled for a single sprint between April 22nd and May 5th. Each phase lasts one week and involves team collaboration tasks. We will evaluate the models based on metrics such as receiver operating characteristics curve (ROC), Area under the

curve, Recall, Precision, Variable importance, and confusion matrix. Also, we will compare the accuracy and analyze the best model among the four classification models for each category. Table 10 and Table 11 give the details of evaluation and deployment tasks and their dependencies respectively. The aim is to meet the business requirements stated in phase one. Finally, we push all the code to Github, merge all individual work into one report, and present the final demo. Figure 12 and Figure 13 is the Gantt chart of Evaluation and Deployment respectively.

Figure 13

Evaluation Gantt Chart

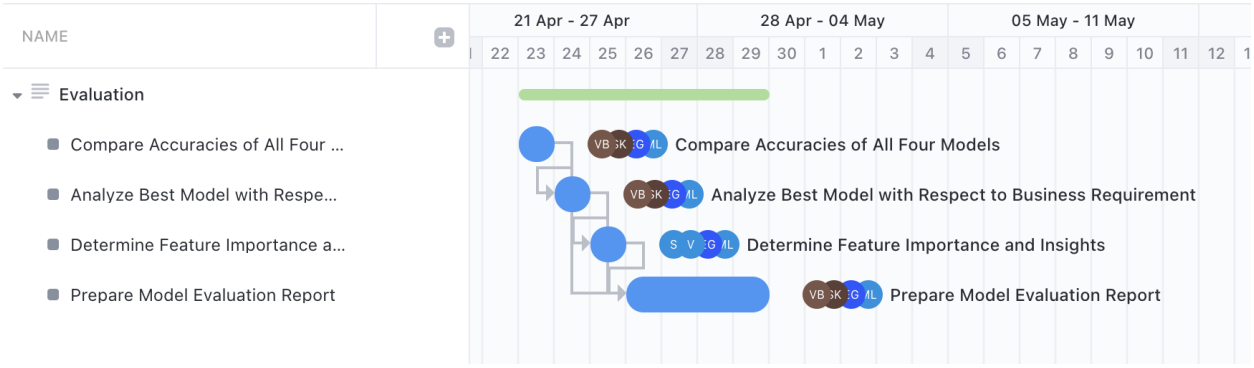


Figure 14

Deployment Gantt Chart

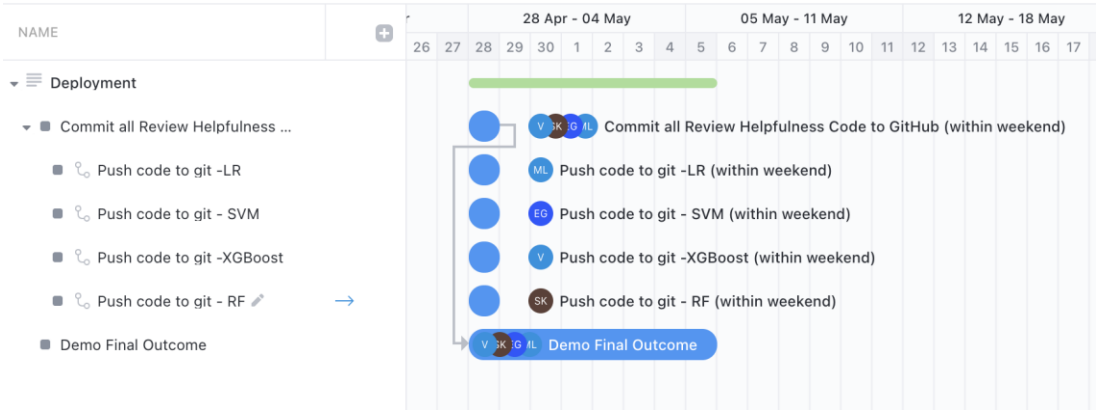


Table 10

Task List for Evaluation

Task Number	Task Name	Depends On	Start Date	End date	Duration
TS1	Compare Accuracies of All Four Models	--	Apr 23	Apr 23	1
TS2	Analyze Best Model with Respect to Business Requirements	TS1	Apr 24	Apr 24	1
TS3	Determine Feature Importance and Insights	TS2	Apr 25	Apr 25	1
TS4	Prepare Model Evaluation Report	TS1, TS2, TS3	Apr 25	Apr 28	4

Table 11

Task List for Deployment

Task Number	Task Name	Depends On	Start Date	End date	Duration
TS1	Commit all Review Helpfulness Code to GitHub	--	Apr 28	Apr 28	1
TS2	Demo Final Outcome	TS1	Apr 28	May 5	8

PERT CHART

PERT chart is used widely in organizations to analyze the project tasks and time required to complete the project. Here we calculate the minimum amount of time needed to finish the project. It gives a visual representation of events in the project timeline. Figure 14 illustrates the PERT chart for Helpfulness of Amazon Reviews. Each component has a task label, task ID, estimated days, start date and end date. The task ids are sequential. The critical task labels have been highlighted while noncritical tasks don't have a highlight in the labels. Most of the report generation tasks have more estimated number of days than the other tasks. In the first phase more time was spent on defining project scope followed by literature and technology survey which is 12 and 15 days respectively. This phase start date is January 29th and end date is March 9th

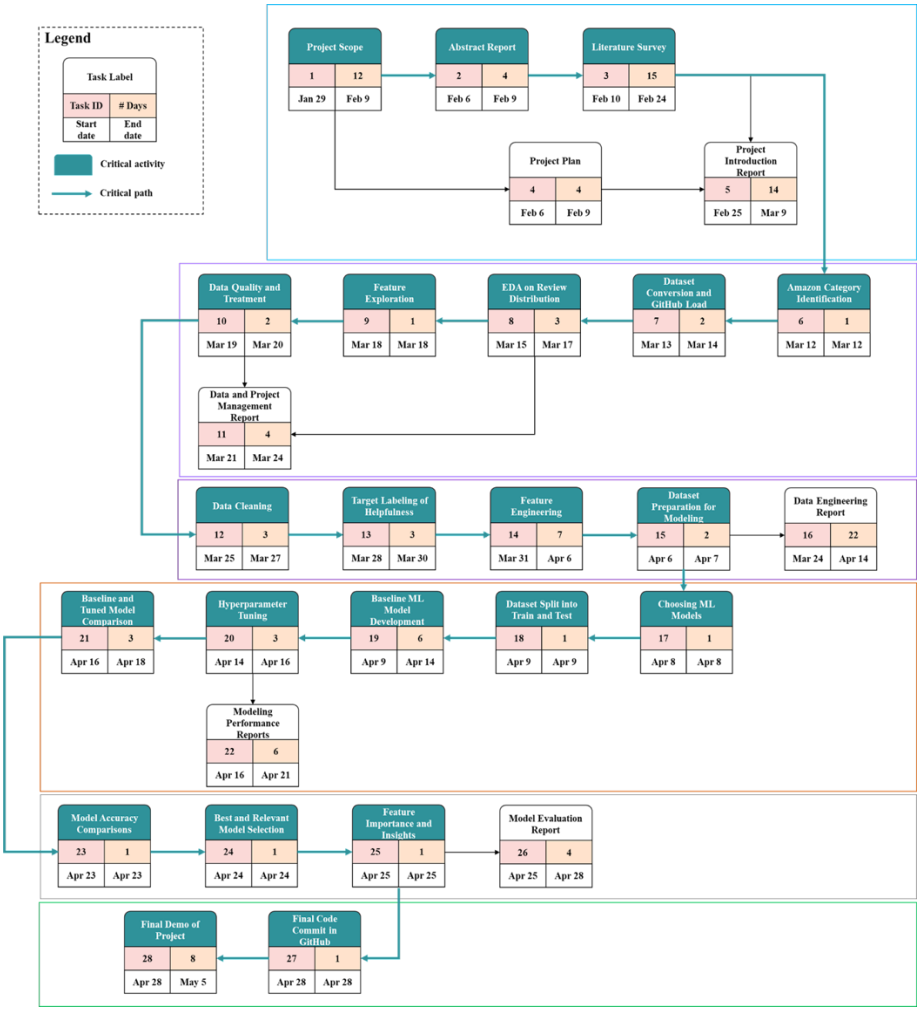
which is two sprints a total of 4 weeks. In the data understanding phase, the tasks estimated days were between 1 to 4 depending on the complexity. It took one sprint to complete this phase with a start date of March 12th to March 24th. Since we started generating the data engineering report on day one of sprint 5 for the data preparation phase, it has the highest estimated days, which is March 24 to April 14. In the modeling phase from April 8th to April 21st, Baseline model takes 6 days followed by other tasks estimated at 3 days and 6 days. It takes a week for evaluation and another week for deployment during the last sprint. Tasks are given 1 day each except for the report and final demo.

The first step in the project is to define the scope of the project based on which we draft the project abstract followed by a literature and technology survey. Amazon category identification in the data understanding phase can be started as soon as the research survey is done. It doesn't really depend on the project plan and project introduction reports which is why these are marked as non-critical tasks. Upon completion of category identification, we convert the raw files and upload them to GitHub. On the review dataset, EDA is performed and the significance of each feature is analyzed thoroughly, which aids in identifying the target feature. Also, we identify data quality issues and define methods for resolving them. All these steps mentioned are critical in order to move to the next phase which is data preparation. Hence, project and data management plan is marked as a non critical task. Data is cleaned by identifying the missing, null and duplicate values. The target variable is created based on the vote and verified feature of the dataset. Feature engineering is performed on the review metadata. Finally, the dataset is made ready for the modeling phase by creating a TF-IDF vectorizer. This phase involves preparing a report on data engineering, which is not critical. Moving to the modeling, we identify the right machine learning model, split the cleaned dataset into train and test,

baseline model for hyperparameter tuning, and compare the baseline model with a tuned model for accuracy comparison in the evaluation phase. Model performance report is the non-critical task hence upon model comparison we choose the best model and determine the insights for evaluation. Model evaluation report is a non-critical task here due to which we push the code to GitHub as soon as insights are drawn from the best model. Lastly, we submit our final report and presentation as a completion of the project.

Figure 15

PERT chart for Harnessing Helpfulness for Amazon Product Review



Data Engineering

3.1. Data Process

To collect data, we explored various eCommerce platforms such as Sephora, Yelp, Walmart, and Amazon. But based on factors like popularity and versatility, we decided to use the Amazon dataset, which was compiled and published by Julian McAuley, UCSD (2014) and Jianmo Ni, UCSD (2018). Though the dataset consists of a collection of reviews from 1996 onwards. We will be focusing on using data ranging from 2010 to 2018. During this period, there was a shift in consumer behavior where consumers relied on the reviews before buying the products. Also, consumers started giving feedback on the products. This timeframe shows the change in trend in consumers in online purchasing. Our project objective can be achieved based on this subset of the dataset.

We restricted to product categories that were objective like appliances, automotive, cell phones and accessories, and tools and home improvement. After converting the raw JSON to csv, we performed data cleaning by removing duplicates, standardizing date formats, imputing missing values, checking non-numeric values, plotting graphs to understand the data distribution, and creating helpfulness binary.

For transformation, we used stratified samples of 50,000 whose distribution is representative of overall data for each category. We performed transformation on textual contents from reviewTitle (summary) and reviewText. The steps involved determining number of words in the review, review length, analysis of sentiment polarity, identifying active reviewers, calculating age of the review. Additionally, we also performed tokenization, stop word removal, and lemmatization. TF-IDF vectorizer was created and distribution of top n-gram was visualized. Data processing and transformation helps us improve the quality of the data

which then can be used in modeling. We have prepared our data for modeling by dividing our cleaned and transformed dataset into 70 % training dataset and 15 % for testing, and 15% for validation.

3.2. Data Collection

For our project, we are using Amazon Review Data (2018). The dataset contains a total of 233.3 million reviews, and the categories we will be using for our project include Appliances, Automotive, Cell Phones and Accessories, and Tools and Home Improvement. These categories are selected to provide a broad range of insights from different types of consumer products available on Amazon. We can use this data that will help sellers to improve content of the listing and make informed marketing strategies and improve customer service. Table 12 includes the data collection plan for our project shown below.

Table 12

Data Collection Plan for all the four categories

Questions	Dataset	Reason
Why are we gathering this information?	Appliances category	We are gathering information to gain insights into how satisfied our customers are with our products. We want to address any common issues they may be facing and get a sense of how they feel about our appliances overall. This data will be used to make improvements to our products, enhancing customer service, and creating more effective marketing strategies.
	Automotive category	We are collecting information to analyze customer opinions on automotive category available for purchase on Amazon. This involves evaluating customer satisfaction, recognizing typical problems or flaws, and understanding market trends. Our main objective is to improve product quality, customer support, and to inform potential buyers.

Questions	Dataset	Reason
	Cell Phones and Accessories category	We are gathering information to understand how customers feel about cell phones and accessories. Our goal is to identify trends, issues, and features that are most important to customers when it comes to these products.
	Tools and Home Improvement category	We are collecting information to better understand customer feedback on tools and home improvement goods available at Amazon. This will help us in identifying trends, features, customer issues, and overall satisfaction with these products.
What should we do after the data has been gathered?	All the four categories	Once data is collected, it is important to clean and prepare the data for analysis in order to maintain its quality. This involves eliminating duplicate values, fixing errors, and normalizing text formats. The next step is to analyze the data for patterns, emotions, and helpfulness of reviews. After studying the data, the findings should be communicated to the appropriate parties (such as product teams, customer support, and marketing) for further improvements.
Past information	--	Yes, Amazon keeps track of all the historical reviews that customers leave for products. Our goal is to gather feedback from the moment a product is listed all the way, allowing customers to give feedback or opinions of the products.
Is there a definition for operations?	--	We will create operational definitions to make sure the data is consistent. For example, we will determine "review helpfulness" by looking at the number of helpful votes compared to total votes.
Duration	--	8 years of data

3.2.1. Samples of raw data resources from Amazon Product Reviews dataset

The Amazon Reviews Dataset (2018) includes valuable feedback from customers across several product categories such as appliances, automotive, cell phones and accessories, and tools and home improvement. The dataset consists into four main categories which we will be using in our project, each containing its own set of data and records: Appliances category, approximately

170.9 MB of data with 13,677 records, Automotive contains about 483.6 MB with 1,711,519 records, Cell Phones and Accessories include 443.2 MB with 1,128,437 records, and Tools and Home Improvement are represented with 756.5 MB of data consisting of 2,070,831 records. Each of these categories is detailed with 12 key features which includes overall product rating (Overall), whether the purchase was verified (Verified), number of helpful votes (Vote), reviewer's identifier (ReviewerID), review posting date (ReviewTime), Amazon Standard Identification Number of the product (ASIN), product style details (Style), reviewer's name (ReviewerName), detailed commentary (ReviewText), review summary (Summary), and the Unix timestamp for when the review was posted (UnixReviewTime). This dataset offers extensive insights into consumer behavior and preferences across diverse product ranges.

The 'overall' feature or column records the overall rating of the product on a 1 to 5 scale given by the customer. The 'vote' holds the count of helpful votes a review has received, suggesting the helpfulness of the review to other customers. The 'verified' feature indicates a boolean whether the customer actually bought the product, which makes the review more trustworthy. The 'reviewTime' represents the date on which the review was posted. The 'reviewerID' is unique to each reviewer. The 'asin' refers to the Amazon Standard Identification Number, a unique code for each product, and is represented as a string to check if there are any non-numeric characters. The 'style' column contains a string that may include metadata about the product, such as format or edition, in a dictionary-like format with key-value pairs. The 'reviewerName' is the display name of the reviewer, while 'reviewText' contains the comment given in detail by the reviewer on the product. Both the ‘reviewerName’ and ‘reviewText’ are string datatypes. The 'summary', which is a string datatype, refers to the overview of the review,

serving as the review's title. The 'unixReviewTime' provides a timestamp in an integer datatype denoting the number of seconds.

The Figure 16 displays a dataset of reviews for Appliances category. It includes columns for rating, verification status, review date, reviewer ID, product ID (ASIN), product style, reviewer's name, review summary, review time in Unix format, snippet of the review text, votes (which shows up as 'NaN,' meaning 'not available'), and images (which shows up as 'NaN,' meaning no images are included). Every review comes from a verified purchase.

Figure 16

Sample Dataset of Appliances Category

	overall	verified	reviewTime	reviewerID	asin	style	reviewerName	reviewText	summary	unixReviewTime	vote	image
0	5	True	08 22, 2013	A34A1UP40713F8	B00009W3I4	{'Style:': ' Dryer Vent'}	James. Backus	I like this as a vent as well as something tha...	Great product	1377129600	NaN	NaN
1	5	True	02 8, 2016	A1AHW6I678O6F2	B00009W3PA	{'Size:': ' 6-Foot'}	kevin.	good item	Five Stars	1454889600	NaN	NaN
2	5	True	08 5, 2015	A8R48NKTGCJDQ	B00009W3PA	{'Size:': ' 6-Foot'}	CDBrannom	Fit my new LG dryer perfectly.	Five Stars	1438732800	NaN	NaN
3	5	True	04 24, 2015	AR3OHHHW01A8E	B00009W3PA	{'Size:': ' 6-Foot'}	Calvin E Reames	Good value for electric dryers	Perfect size	1429833600	NaN	NaN
4	5	True	03 21, 2015	A2CIEGHZ7L1WWR	B00009W3PA	{'Size:': ' 6-Foot'}	albert j. kong	Price and delivery was excellent.	Five Stars	1426896000	NaN	NaN

The Figure 17 shows a dataset from the Automotive category with fields for votes and images, both displaying 'NaN' indicating missing data, and columns for ratings, verified status, review date, reviewer ID, product ID (ASIN), product color (style), reviewer name, and a portion of the review text and review summary. Reviews are mix of verified and unverified, and ratings range widely.

Figure 17

Sample Dataset of Automotive Category

	overall	verified	reviewTime	reviewerID	asin	style	reviewerName	reviewText	summary	unixReviewTime	vote	image
0	4	False	05 1, 2015	A8WEXFRWX1ZHH	0209688726	{'Color:': ' AC'}	Goldengate	After I wrote the below review, the manufactur...	Works well if you place phone in horizontally ...	1430438400	NaN	NaN
1	1	True	04 19, 2018	ABCA1A8E4DGV1	0209688726	{'Color:': ' Blue'}	noe	It sucks barely picks up anything definitely n...	sucks	1524096000	NaN	NaN
2	1	True	04 16, 2018	A1NX8HM89FRQ32	0209688726	{'Color:': ' Black'}	Eduard	Well to write a short one, it blew 2 fuses of ...	Defective	1523836800	NaN	NaN
3	3	True	04 13, 2018	A1X77G023NY0KY	0209688726	{'Color:': ' CA'}	Lauren	I have absolutely no memory of buying this but...	Looks cool! Probably works	1523577600	NaN	NaN
4	5	True	04 8, 2018	A3GK37JO2MGW6Q	0209688726	{'Color:': ' Black'}	danny	it ok it does it job	Five Stars	1523145600	NaN	NaN

The Cell Phones and Accessories category dataset is displayed in Figure 18. It contains columns for product ratings, verified purchase status, review date, reviewer ID, product ID (ASIN), product style (one entry displaying "Color: Bling"), reviewer's name, a section of the review text, review summary, and review time in Unix format. Two columns for vote and image have 'NaN' values, indicating no data for these fields. Ratings for the reviews, which range from 1 to 5, are all confirmed.

Figure 18

Sample Dataset of Cell Phones and Accessories Category

	overall	verified	reviewTime	reviewerID	asin	style	reviewerName	reviewText	summary	unixReviewTime	vote	image
0	5	True	08 4, 2014	A24E3SXTC62LJI	7508492919	{'Color:': ' Bling'}	Claudia Valdivia	Looks even better in person. Be careful to not...	Can't stop won't stop looking at it	1407110400	NaN	NaN
1	5	True	02 12, 2014	A269FLZCB4GIPV	7508492919	NaN	sarah ponce	When you don't want to spend a whole lot of ca...	1	1392163200	NaN	NaN
2	3	True	02 8, 2014	AB6CHQWHZW4TV	7508492919	NaN	Kai	so the case came on time, i love the design. I...	Its okay	1391817600	NaN	NaN
3	2	True	02 4, 2014	A1M117A53LEI8	7508492919	NaN	Sharon Williams	DON'T CARE FOR IT. GAVE IT AS A GIFT AND THEY...	CASE	1391472000	NaN	NaN
4	4	True	02 3, 2014	A272DUT8M88ZS8	7508492919	NaN	Bella Rodriguez	I liked it because it was cute, but the studs ...	Cute!	1391385600	NaN	NaN

The Tools and Home Improvement category dataset is shown in Figure 19. It contains reviews with ratings (from 1 to 5) and verification status (all true), review date, reviewer ID, product ID (ASIN), product style details, reviewer name, portions of the review text, review summaries, review times in Unix format, and a column for votes (mostly 'NaN' with one entry having a value of '16'). Additionally, there is a column for images (given by 'NaN') that are

not included.

Figure 19

Sample Dataset of Tools and Home Improvement Category

	overall	verified	reviewTime	reviewerID	asin	style	reviewerName	reviewText	summary	unixReviewTime	vote	image
0	5	True	01 28, 2018	AL19QO4XLBQPU	0982085028	{'Style:': ' 1) IR30 POU (30A/3.4kW/110v)'}	J. Mollenkamp	returned, decided against this product	Five Stars	1517097600	NaN	NaN
1	5	True	11 30, 2017	A1I7CVB7X3T81E	0982085028	{'Style:': ' 3) IR260 POU (30A/6kW/220v)'}	warfam	Awesome heater for the electrical requirements...	Five Stars	1512000000	NaN	NaN
2	5	True	09 12, 2017	A1AQXO4P5U674E	0982085028	{'Style:': ' Style64'}	gbieber2	Keeps the mist of your wood trim and on you. B...	Five Stars	1505174400	NaN	NaN
3	4	True	07 19, 2017	AIRV678P7C4NK	0982085028	NaN	Justin Banner	So far I hooked it up and tested it , filled a...	it is the perfect temp for a shower	1500422400	NaN	NaN
4	1	True	05 25, 2017	A22I5QDNTNECDW	0982085028	{'Style:': ' 3) IR260 POU (30A/6kW/220v)'}	daveparker	i installed this 10 months ago, instructions w...	worked well...for 10 months.	1495670400	16	NaN

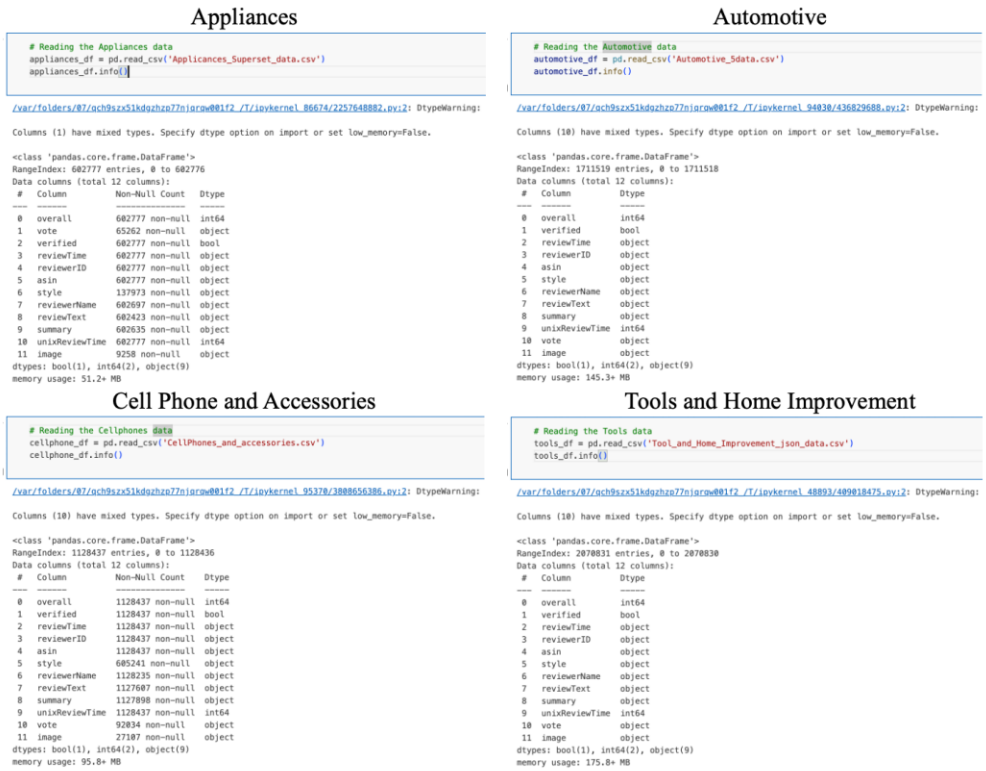
3.3. Data Pre-processing

We sequentially have records for all the above-mentioned categories in CSV format, which is suitable for a quick overview to gain rough insights and for further feeding it to Python for data analysis. Data pre-processing is a very crucial step in any data analysis or machine learning project. We need to prepare data for machines to learn from it and provide us with meaningful results. Data pre-processing involves transforming raw data into a format that is suitable for training and testing models of high quality. The main objectives include removing noise or irrelevant information, removing outliers, cleaning data, handling missing values, and preparing data well for further analysis and modeling to get higher accuracy and correct predictions. There is a line that is always associated with the field of data analysis: “Garbage-In, Garbage-Out”. It clearly means that the quality of the results is linearly related to the quality of the input of data. Therefore, in this section, we will primarily focus on the techniques used to pre-process data for modeling purposes.

We shall start by importing the respective CSV files into Python and importing the necessary libraries relevant to our usage. To have a quick overview of the dataset we used the info () method from the “Pandas” library as it provides valuable insights into the structure and characteristics of a dataset. It provides us the number of rows and columns in the dataframe, shows the data types of each column to understand the nature of data and to determine if any type conversions are required. In the Figure 1 below, we see we have millions of rows for every category with a list of columns having name and data type. The Figure 20 also provides the data shape of each of the category, indicating the number of reviews available in our raw dataset.

Figure 20

Review Data Category characteristics.



Note. Information about the columns and row counts for different categories from the Amazon review dataset.

We clearly see here that the dataset is size-heavy, having review counts in millions. Deduplication is needed on large datasets as it's essential to ensure data quality, consistency, and uniformity. Duplicate values can skew the analysis and introduce unnecessary noise. We move further to check on the missing values and remove duplicates. Post deduplication, we noticed that approx. 4% of data was duplicate for Tools and Home Improvement and Automotive category, 1% for Appliances and 0.3% for Cell Phones and Accessories, refer Table 13. Data shrinkage was negligible after removing duplicates which is understandable given the volume of data we have for each category.

Table 13

Deduplication metrics by category

Category	Deduplication Metrics
Appliances	Before deduplication shape: (602777, 12) After deduplication shape: (591371, 12) Percentage of duplicates removed: 1.8922420729390803 %
Automotive	Before deduplication shape: (1711519, 12) After deduplication shape: (1647280, 12) Percentage of duplicates removed: 3.753332565983784 %
Cell Phones and Accessories	Before deduplication shape: (1128437, 12) After deduplication shape: (1124986, 12) Percentage of duplicates removed: 0.30582123769426206 %
Tools and Home Improvement	Before deduplication shape: (2070831, 12) After deduplication shape: (1982666, 12) Percentage of duplicates removed: 4.257469585881223 %

In addition to reviewing the de-duplication, we also checked for missing values, including the NA, NaN, and NULL/Blank values. Removing null/missing values from a dataset

is very important because many machine learning algorithms and statistical analyses cannot handle missing data efficiently. These values can introduce bias or inaccuracies in the result and can also cause errors or unexpected behavior in computations. We found that certain features such as vote, verified, style, reviewerName, reviewText, summary, and image have missing values in all the categories. The Figure 21 below shows the count of missing values for every column in all the categories.

Figure 21

Category wise NULL/Missing record counts

Appliances

```
# Function to count missing values (including NA, NaNs, and 0)
def count_missing(df):
    missing_counts = df.isna().sum()  # Count missing values (NaN)
    missing_counts = missing_counts.add(df.eq(0).sum(), axis=0)  # Count zeros
    return missing_counts

# Printing the counts of missing values
missing_value_counts = count_missing(appliances_df_working_dedup.copy())
print(missing_value_counts)
```

```
overall              0
vote            529749
verified         37781
reviewTime           0
reviewerID           0
asin                 0
style           455622
reviewerName        80
reviewText         351
summary            138
unixReviewTime       0
image           583097
dtype: int64
```

Automotive

```
# Function to count missing values (including NA, NaNs, and 0)
def count_missing(df):
    missing_counts = df.isna().sum()  # Count missing values (NaN)
    missing_counts = missing_counts.add(df.eq(0).sum(), axis=0)  # Count zeros
    return missing_counts

# Printing the counts of missing values
missing_value_counts = count_missing(automotive_df_working_dedup.copy())
print(missing_value_counts)
```

```
overall              0
verified         77938
reviewTime           0
reviewerID           0
asin                 0
style          1069958
reviewerName       253
reviewText         887
summary            351
unixReviewTime       0
vote           1463684
image          1605545
dtype: int64
```

Cell Phone and Accessories

```
# Function to count missing values (including NA, NaNs, and 0)
def count_missing(df):
    missing_counts = df.isna().sum()  # Count missing values (NaN)
    missing_counts = missing_counts.add(df.eq(0).sum(), axis=0)  # Count zeros
    return missing_counts

# Printing the counts of missing values
missing_value_counts = count_missing(cellphone_df_working_dedup.copy())
print(missing_value_counts)
```

```
overall              0
verified        140453
reviewTime           0
reviewerID           0
asin                 0
style           520614
reviewerName       202
reviewText         829
summary            539
unixReviewTime       0
vote           1033453
image          1097901
dtype: int64
```

Tools and Home Improvement

```
# Function to count missing values (including NA, NaNs, and 0)
def count_missing(df):
    missing_counts = df.isna().sum()  # Count missing values (NaN)
    missing_counts = missing_counts.add(df.eq(0).sum(), axis=0)  # Count zeros
    return missing_counts

# Printing the counts of missing values
missing_value_counts = count_missing(tools_df_working_dedup.copy())
print(missing_value_counts)
```

```
overall              0
verified        162483
reviewTime           0
reviewerID           0
asin                 0
style           959406
reviewerName       264
reviewText         610
summary            301
unixReviewTime       0
vote           1697705
image          1939789
dtype: int64
```

Note. Summarized counts of missing or blank records per category for each feature

Before dealing with the missing values for each category, we had to understand the data in different columns to decide on the treatment. Table 14 below is the list explaining each column from the dataset along with its data type.

Table 14

List of common columns for all categories

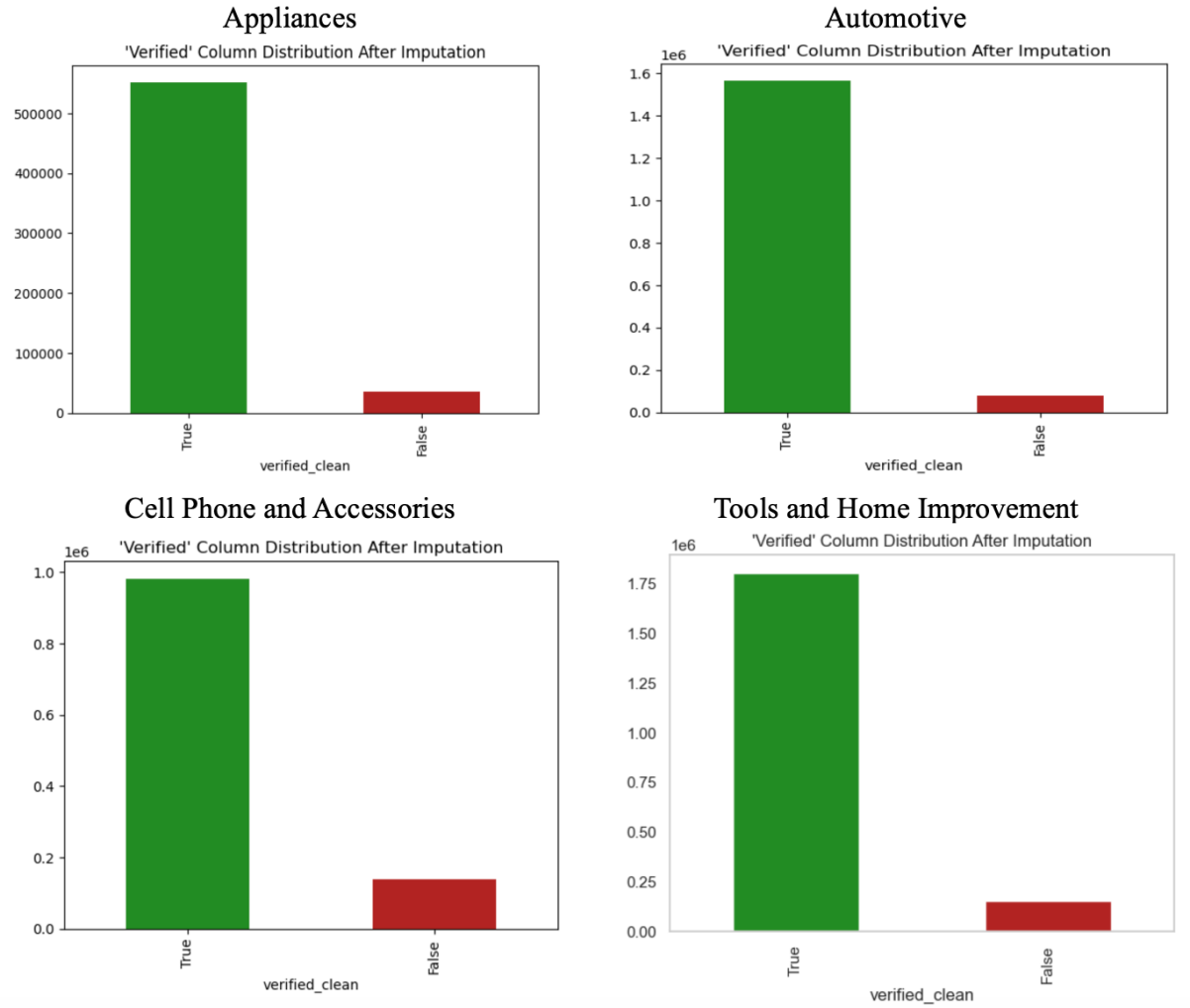
Column	Summary	Original Datatype
Overall	Rating for the Product	Int64
verified	If the purchase is verified or not. Also, useful for getting helpfulness dependent variable	Bool
reviewTime	Time at which the review was posted in "mm dd, yyyy" format	Object
reviewerID	ID of the reviewer who posted the review	Object
asin	Amazon Standard Identification Number	Object
style	Describes the SKU or variant	Object
reviewName	Name of the reviewer	Object
reviewText	Actual review text	Object
summary	Review title	Object
vote	Number of votes for helpfulness	Object
image	URL of review image if available	Object
unixReviewTime	Time of the review (Unix format)	Int64

Starting with the "reviewTime", which was of the type "object", we standardized it in standard datetime format. We also created separate columns for the date feature to split day, month and year, the resulting columns were named "review_day", "review_month" and "review_year" respectively. Subsequently, as we had discussed that will be focusing on the reviews from the year 2010 to 2018, we applied a filter on the "review_year" column to only consider relevant years. However, we found that majority of the reviews already fell under this date range, and the excluded record count was negligible. After formatting the review dates, we looked at the "verified" column which was an important feature for our target variable "helpfulness". This column had a high number of missing values, and it had only two distinct Boolean values, true or false; so, we imputed the missing values with false and created a new

feature “verified_clean”. As this feature was very important to derive the helpfulness of a review, we considered that the missing values were not verified purchases. The Figure 22 below shows the data distribution after the imputation.

Figure 22

Distribution for feature “verified_clean” post Imputation



Note. Plots above show the distribution for the column “verified_clean” across categories.

We next focused on the column “vote” which was another feature that was critical for our target variable, and it also had a lot of missing values. Missing values for this column generally signifies that no one voted the review as helpful. We imputed missing values with zero and

created a new feature “vote_clean”. As for the “Style” feature, an empty value was replaced by an empty string. For the “reviewName” feature, we found that there were a few instances having missing values. If the reviewer’s name is missing, we don’t know who the reviewer is, and hence, we have imputed the missing values with “Unknown”. Our most important column “reviewText” that was crucial for feature extraction had very few instances of missing or NaN values, so we decided to exclude them from the overall dataset as the volume was negligible. On similar lines, we handled the missing values for the column “summary” by dropping missing values as the volumes were insignificant. Finally, for the “image” feature, we checked if the image URL existed or not, using a regular expression matching and we created a new feature called “image_available”. This new feature was created to hold only Boolean values, either true or false, as we wanted to use the metadata but not the image itself for our analysis. Post treatment of missing values, we were successfully able to get a dataframe with treated values and these new features were free from missing values, duplicates, or NaN values. Figure 23 below provides a quick snapshot of how the newly created columns looked in the dataframe.

Figure 23

Snapshot of the newly created features

reviewTime_cleaned	review_year	review_month	review_day	verified_clean	vote_clean	image_available
2017-03-07	2017	3	7	True	0	False
2017-03-02	2017	3	2	True	0	False
2017-01-07	2017	1	7	True	0	False
2018-04-17	2018	4	17	True	0	False
2017-12-04	2017	12	4	True	0	False

Note. Snapshot of the newly created columns in our data frames post initial cleaning.

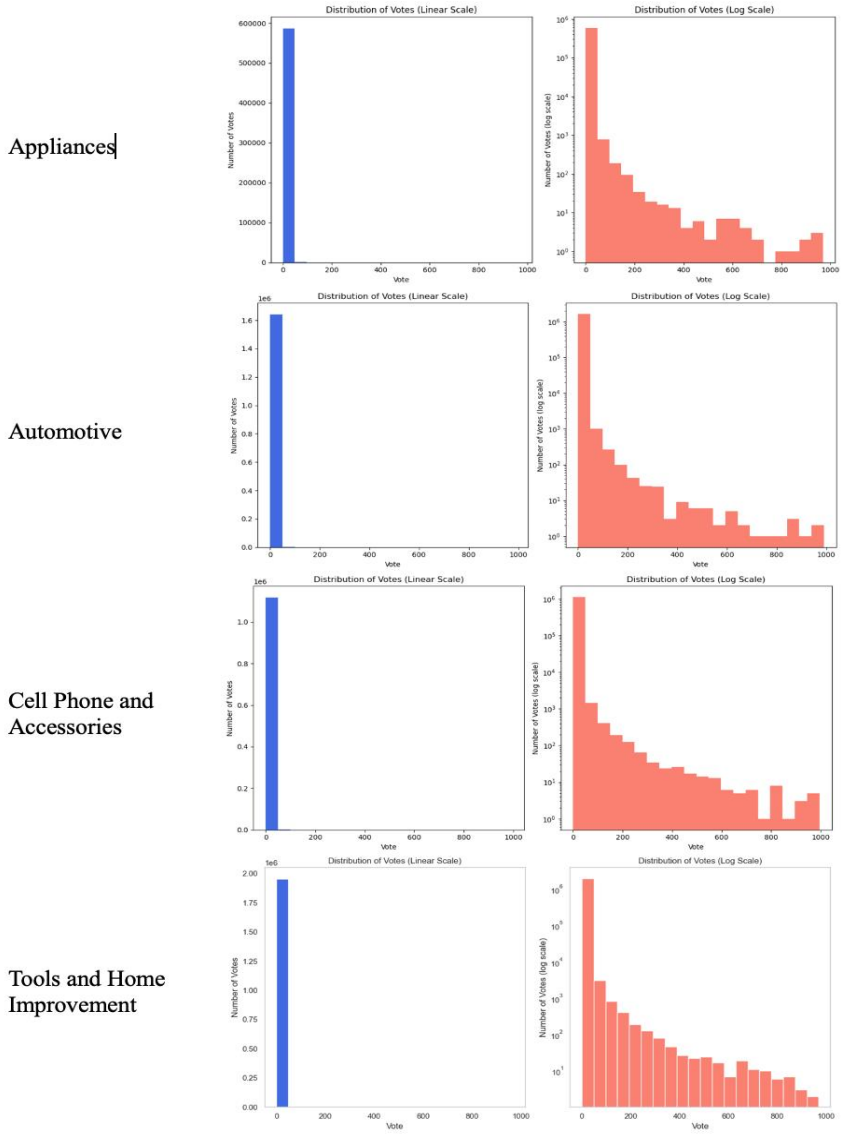
Our next point of focus was to create the dependent variable “helpfulness”. Helpfulness is a target variable that is determined by a combination of “number of votes (vote_clean)” and “verified purchases (verified_clean)” features. When we created the “vote_clean” feature, its parent column “vote” was of the datatype “object”. Since, votes are always numeric, we converted the newly created feature “vote_clean” to numeric format. Post conversion, we verified for any redundant values that could not convert into numeric. There were very few instances like that, and we dropped those occurrences from our dataset. We also performed the same checks for “summary” and “reviewText” columns for NaN values. To avoid biased results, errors or unexpected behavior, we simply removed them from our dataset. These amounted to a very negligible volume of approximately 0.05 – 0.1% of the overall dataset, so it was safe to remove them.

We plotted the spread of the vote counts using the feature “vote_clean” in the dataset to see how they are spread out. We saw that values of vote counts mostly spanned from 0 to 1000, meaning for a single review, the number of votes can be 1000, while for another review, they can be 0 within the same category. Major chunk of the reviews were the ones with no votes while there was a big spike from 0-100 range in majority of the categories. This means, that certain products were much better received as compared to others and had a higher reach in terms of buyers within a single category. This is often the result of brand recognition, the number of positive reviews and vote count related to the helpfulness of the reviews. We tried to plot the distribution of votes on a box plot to see the percentile split of votes. However, it was not intuitive as the number of records with no votes was high for all categories. We then decided to visualize the distribution with simple histograms with linear and logarithmic scales. The Figure 24 below shows both the plots side by side for each category. As you can see from the figure, the

linear scale is not able to provide an accurate visual of the split. So, we decided to use logarithmic scale as it compresses larger values and expands smaller values, allowing us to better visualize a feature with such type of distributions.

Figure 24

Vote Distribution, Linear vs Logarithmic



Note. Distribution of votes on both linear and logarithmic scale

After the initial cleaning, we were ready to create our target feature called "helpfulness _binary" using the "vote_clean" and "verified_clean" features with certain conditions. Helpfulness was tagged with a value of 1 only if the number of votes was greater than 0 and if the review was verified i.e. “verified_clean” is True, for all other combinations, it was tagged with a value of 0. Therefore, our target variable can be interpreted as “1” being helpful, and “0” being non-helpful. The Table 15 below shows the counts of feature “helpfulness_binary” for each of the categories, along with the percentage of helpful/non-helpful reviews.

Table 15

Percentage Distribution of Target Feature

Category	Helpfulness	Percentage
Appliances	0: 540331	0: 92%
	1: 47005	1: 8%
Automotive	0: 1474453	0: 90%
	1: 166434	1: 10%
Cell Phones and Accessories	0: 1050626	0: 94%
	1: 68505	1: 6%
Tools and Home Improvement	0: 1726619	0: 88%
	1: 231963	1: 12%

Note. Distribution for the feature “helpfulness_binary” across all categories showing an approx. 90:10 ratio for all categories.

The table above clearly shows the count for non-helpful reviews is much higher than helpful reviews. Such distributions are usually considered as imbalanced data. However, if we review all the categories individually, it roughly follows a ratio of 90:10. This ratio is acceptable for our use case as having a 50:50 ratio would defeat the overall goal of our use case, which is to find the review helpfulness. We observe that the dataset for all the categories is volume intensive. Working with a large dataset in a Jupyter notebook on a local computer poses many challenges like high CPU/RAM usage with simple transformations, slow/long-running computations, data transformation challenges and challenges with data visualizations. We

encountered this with majority of our datasets, and it was becoming challenging to complete basic transformations for feature engineering. Keeping this in mind, we decided to do a stratified sampling on our datasets for further feature engineering and modelling. We carefully considered several factors to have an appropriate and unbiased sample drawn from the large dataset, and stratified sampling was the best fit for our use case as we had imbalanced data. With stratified sampling, the population is divided into non-overlapping subgroups called “strata” based on the characteristics of dataset. Unlike random sampling, which treats the population as homogeneous, stratified sampling ensures adequate representation of each stratum in the sample. This technique can effectively reduce sampling variance and enable separate estimates and analyses for specific subgroups of interest in the overall population. We decided to take 50,000 records as the total sample volume for all the categories.

While specifying the strata features, we have considered “helpfulness_binary” as strata, which is our dependent feature, to ensure that the class balance remains the same. We also considered “review_year” and “review_month” to have a sample representative of seasonality and review age. To ensure that the same is representative of the overall data, we have considered the “image_available” feature. The Table 16 below shows the distribution of reviews as helpful/not helpful under the sample taken for each of the categories.

Table 16

Percentage Distribution of Target Feature after Sampling

Category	Helpfulness	Percentage
Appliances	0: 45995	0: 92%
	1: 4000	1: 8%
Automotive	0: 44928	0: 90%
	1: 5074	1: 10%
Cell Phones and Accessories	0: 46942	0: 94%
	1: 3064	1: 6%
Tools and Home Improvement	0: 44075	0: 88%
	1: 5921	1: 12%

Note. Distribution of target feature on sampled data after applying stratified sampling.

We see that after stratified sampling, the percentage distribution of helpfulness/not helpfulness is the same as of the population. The Figures 25, 26, 27, 28 below show the percentage distribution of the overall vs stratified sample data across different categories.

Figure 25

Helpfulness distribution, Overall vs Sample, Appliances category

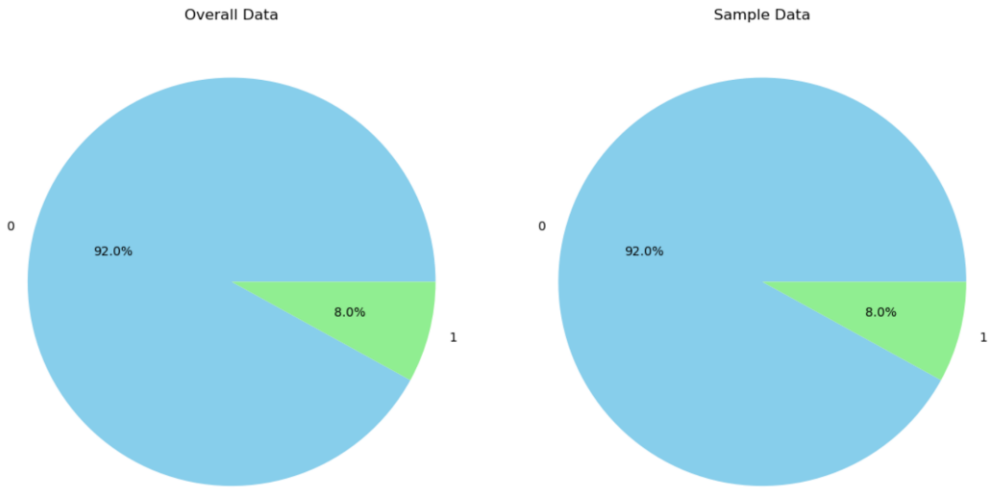


Figure 26

Helpfulness distribution, Overall vs Sample, Automotive category

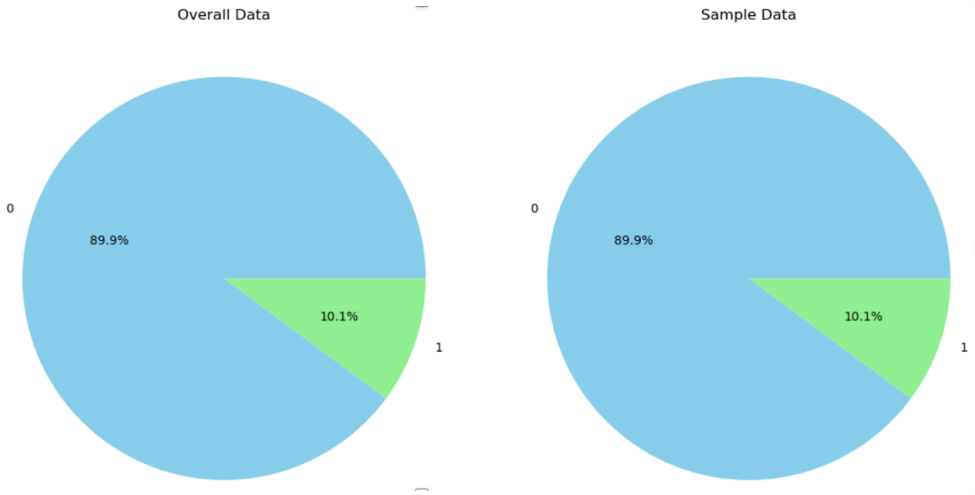


Figure 27

Helpfulness distribution, Overall vs Sample, Tools & Home Improvement category

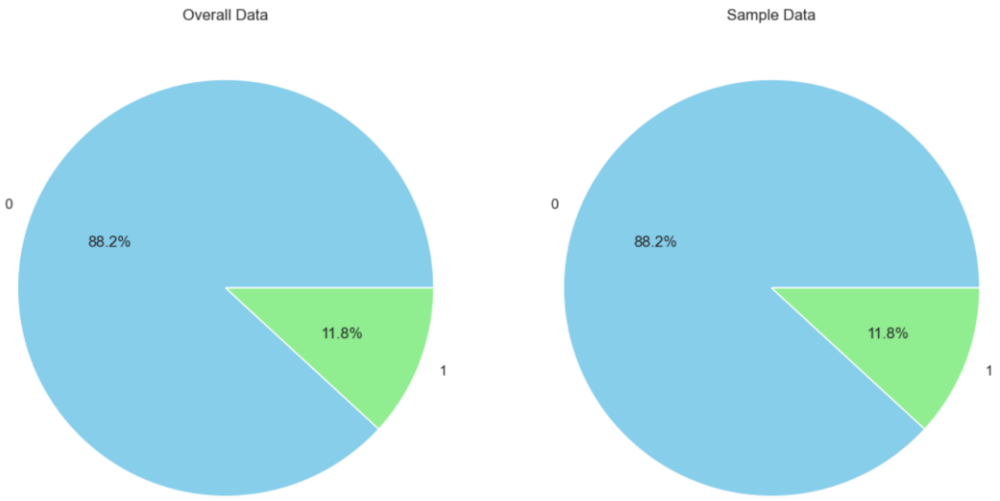
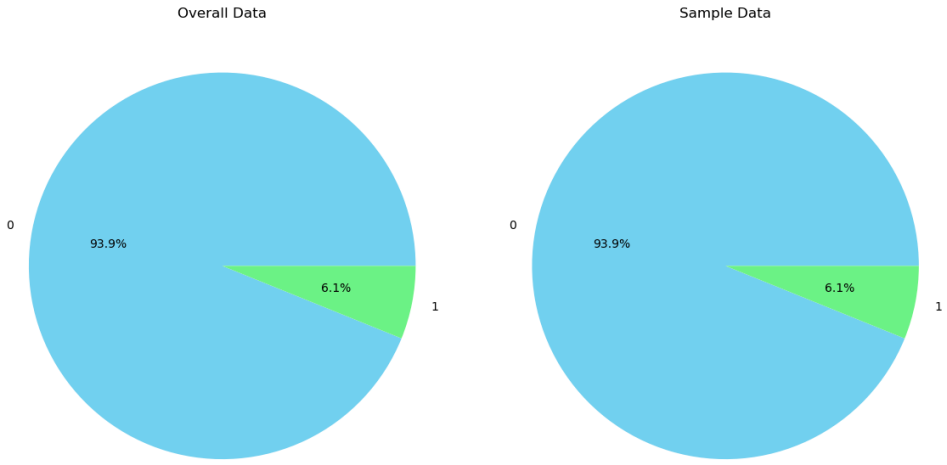


Figure 28

Helpfulness distribution, Overall vs Sample, Cell Phones & Accessories category



This proves that sample data is a true representative of the overall population and further analysis can be carried out on this sample data effectively. As the percentage split after sampling matches with the original dataset, reducing volume did not introduce any unnecessary bias into our analysis. The feature extraction on the sample would be equally effective as with the original dataset and would help eliminate the resource constraints we had due to large volumes.

3.4. Data Transformation

Data transformation involves cleaning, integration, organizing, aggregation, and enriching the data for modeling. This step is crucial in the Machine Learning lifecycle, as building the models will be easy with transformed data. It will also help improve the models' performance. Amazon is a global marketplace leader among the various e-commerce platforms. Leveraging the rich repository of Amazon reviews through text mining techniques allows us to extract many insights into customer preferences, product sentiments, and market trends.

Given the vast amount of the review data from the source, we decided to use stratified sampling. This method involves selecting a representative sample from each category, ensuring

that the sample reflects the same proportions of the overall data. We chose to sample 50K records for each category and checked to ensure that the proportion of helpful versus non-helpful reviews was equal. We then proceeded to transform the stratified sample, starting with the review title feature. We extracted latent features of the text data, which we refer to as metadata features and there are two primary columns with the text data – review title and review text. This was done since the title and review text can have varying degrees of information. The title is intended to be a simple summary, while the review text is a more elaborate description. For the title, we started by extracting the number of words in the title, the length of the original title, the sentiment associated with the title, unique word count, emoji, and non-ASCII character count. The sentiment of the review title is essential to understand the sentiments and they take three values such as positive, negative, and neutral. Once the customer makes a purchase, any sentiment can be associated with the product.

If the sentiment is positive, the customer can purchase the product again, and if the sentiment is negative , the customer is most unlikely to buy the product or recommend the product. This insight can help the sellers to improve the listing content. We have created functions to create each of the metadata features mentioned above. The length of the title, the number of words in the title and unique word count can help in understanding the relationship between long and verbose titles on helpfulness. Emoji and non-ASCII character count can help in understanding whether titles with a lot of unprintable characters are helpful or not. Insights from analyzing the metadata features of titles can be crucial in predicting helpfulness and can also be used by sellers to improve customer experience.

After completing the review title features, we proceeded to work with the review text feature. In the review text feature on the sample, apart from the metadata features mentioned

above for title, which have similar utility for review text as stated earlier, we created additional metadata features like the presence of URL and stop word count with custom functions. URL presence may be an interesting feature since we can understand if reviews, wherein customers share links to other products or websites, are helpful or not. Stop word count may also be an interesting feature since we will be removing stop words from the review text. Stop words are words that are often used for grammatical purposes like “a”, “the”, “is”, “an”, etc. but do not add value to the underlying meaning of the text and can be excluded from text processing since they do not carry a lot of information. Identifying the count of stop words helps understand the ratio of signal versus noise in a given text.

We also calculated the review age by considering the time difference in days between a review being posted and the end of the data, that is, 31st Dec 2018. The review age feature brings temporal data directly into the dataset so that more recent reviews are represented by smaller numbers, which can be more informative to assessments of recent product purchases.

We perform text cleaning on the review text feature to create the review_text_clean feature. Text cleaning is done in three stages. The first stage is converting all the review text to lowercase to offset the effect of different textual cases on the raw data. This allows algorithms to identify repetitions of the same words regardless of whether these words are originally used in upper, lower, or mixed cases in reviews. In the second stage, we expand contractions to make the text clearer for machines to read by retaining the actual semantics and structure. Contractions in text processing refer to words that are shortened by replacing letters with apostrophes like “I’m” and “they’re” which would be resolved to “I am” and “they are” respectively. In the third stage, we used some functions to clean up the review text: removing non-ASCII characters and special characters such as emojis from the review text feature. Since these are unimportant, we are also

removing special characters, stop words, and digits from the review text. Another function expands contractions and colloquial abbreviations, turning shorthand notations into complete descriptions like “tbh” to “to be honest” so that the algorithms understand the contextual usage. All of these text transformations are applied to the ‘review text’ feature. Finally, we get feature review_text_clean after all the transformations as shown in Figure 29 below for the Appliances category. Similar patterns were observed in other categories as well.

Figure 29

Review_text_clean feature for the Appliances category.

```
26599     would drive miles pick filter one arrived seve...
115267    writing review counter current star rating rev...
96843     john fact think neighbors posted product sever...
108945    replaced year old dead quiet white maytag stai...
301       gas dryers give carbon monoxide using device o...
Name: review_text_clean, dtype: object
```

We performed lemmatization on the cleaned text. Lemmatization is the process of reducing different inflected forms or variants of the same word in text data to a common root word. For instance, changing, changed, and change can be mapped to change. This is a kind of text normalization technique. After lemmatization, we get a feature 'review_text_lemmatized'. Finally, we have a dataset with all the features that can be used to prepare the data for modeling. Figure 30 shows the final metadata columns and snapshot of the dataset after all the transformations have been applied.

Figure 30

Final metadata columns of the dataset and sample transformed data

```
Index(['overall', 'vote', 'verified', 'reviewTime', 'reviewerID', 'asin',
       'style', 'reviewerName', 'reviewText', 'summary', 'unixReviewTime',
       'image', 'reviewTime_cleaned', 'review_year', 'review_month',
       'review_day', 'verified_clean', 'vote_clean', 'image_available',
       'helpfulness_binary', 'num_words_review_title', 'title_length',
       'title_sentiment', 'unique_word_count_title',
       'emoji_non_ascii_count_title', 'num_words_review_text', 'review_length',
       'review_sentiment', 'unique_word_count_review', 'url_count_review',
       'stop_word_count_review', 'review_age_days', 'review_text_clean',
       'review_text_clean_lemmatized'],
      dtype='object')
```

num_words_review_text	review_length	review_sentiment	unique_word_count_review	url_count_review	stop_word_count_review	review_age_days	review_te
172	880	0.186111	95	0	50	3280	cartridges work c work
60	300	-0.011111	48	0	29	3280	repl produc glove how
789	3841	0.121429	339	0	296	3261	heirarchy induc althou
310	1612	-0.057292	190	0	103	3282	bosch sh reputation en
23	95	0.377778	20	0	11	3279	power length qua ea

Once we have processed data, we convert the data into a TF-IDF matrix. TF-IDF stands for Term Frequency-Inverse Document Frequency and is a method of measuring the relevance of a word in a document and a collection of documents. The TF part stands for Term Frequency, which is just the count of word occurrences. The IDF part stands for Inverse Document Frequency and is the log of the ratio of the total number of documents to the number of documents in which a term appears. The final metric is a product of TF and IDF. This method ensures that words that occur too frequently get a low score and rare or relevant terms get a higher score. This ensures that we score terms based on their relative importance and not just the number of times they are used. To apply the TF-IDF, we use the TfidfVectorizer method from sklearn and restrict the number of features to 1000 (using the max_features parameter) since we

want to avoid having a sparse matrix that may be difficult to interpret, resulting in very high dimensionality and can affect the modeling.

We have visualized the review text clean feature by creating n-gram distributions across all the categories. Figure 31, Figure 32, Figure 33 and Figure 34 show unigram for all categories.

Figure 31

Distribution of unigram for Appliances, Automotive, Cellphones and Accessories,

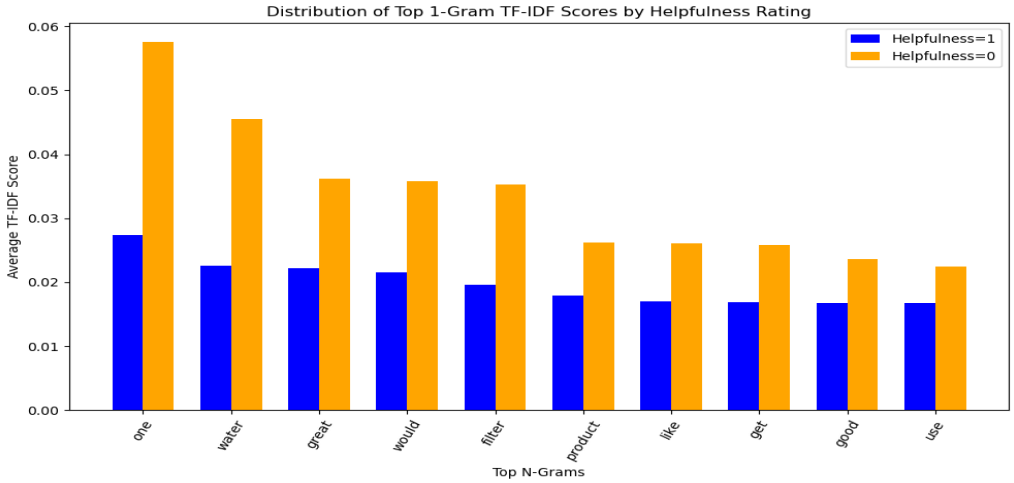


Figure 32

Distribution of unigram for Automotive category

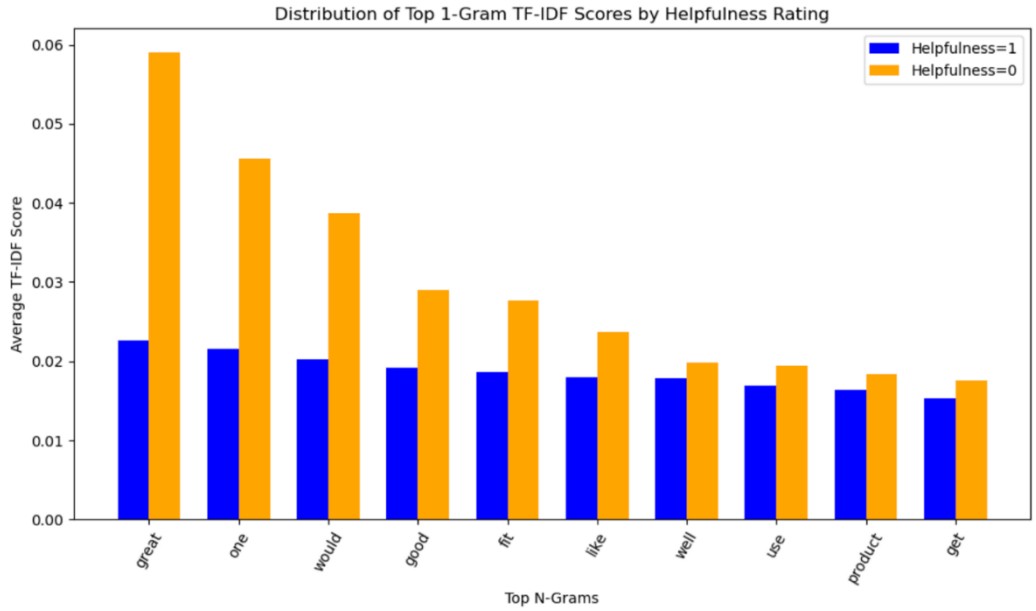


Figure 33

Distribution of unigram for Cellphones and Accessories category

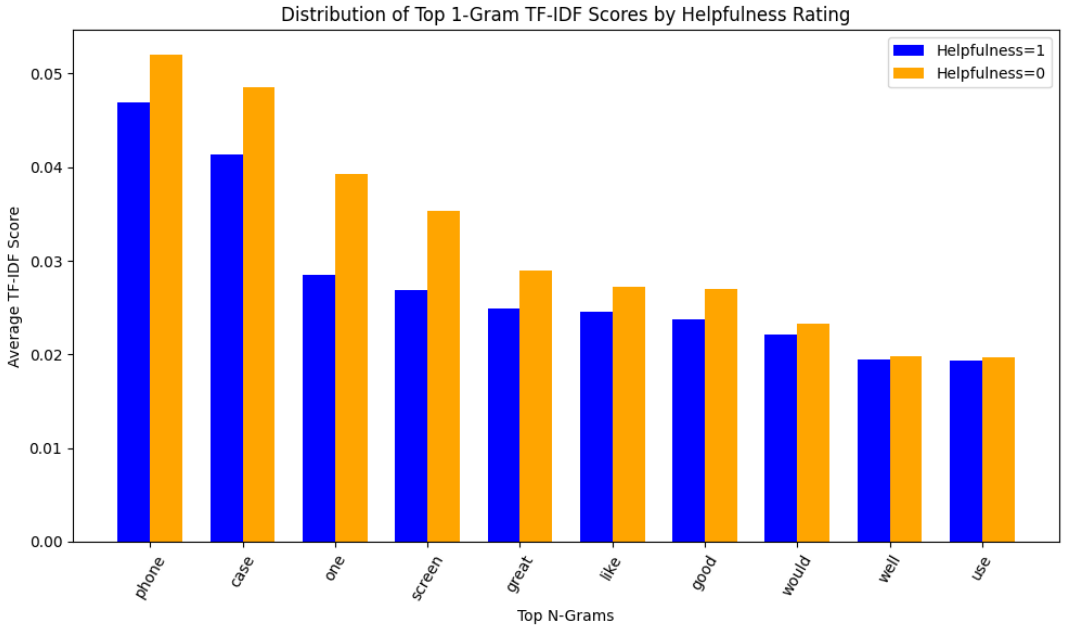
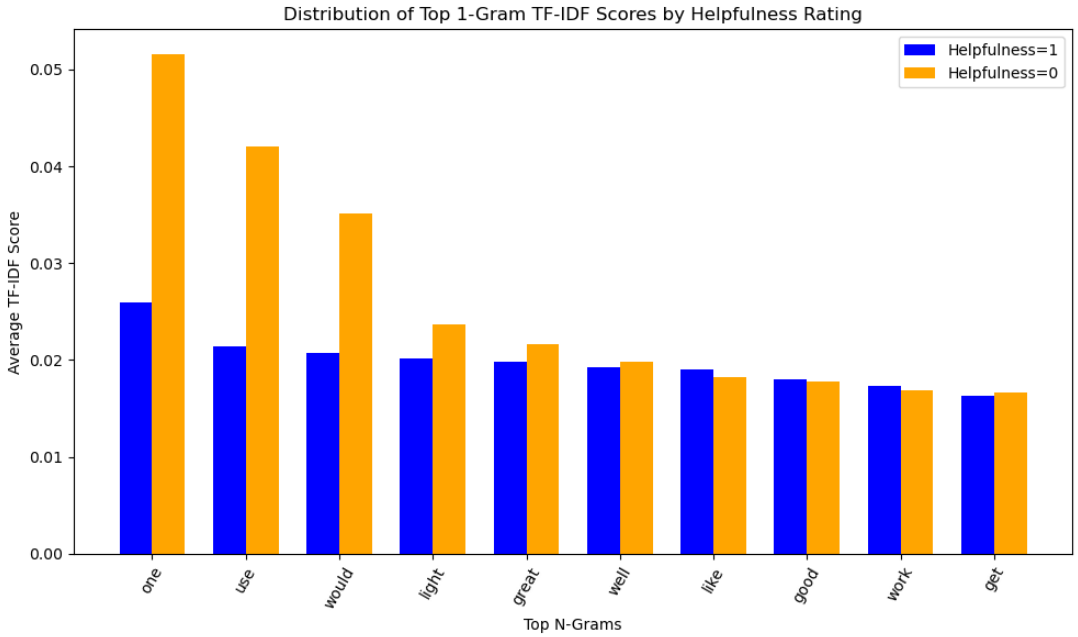


Figure 34

Distribution of unigram for and Tools and Home Improvements category

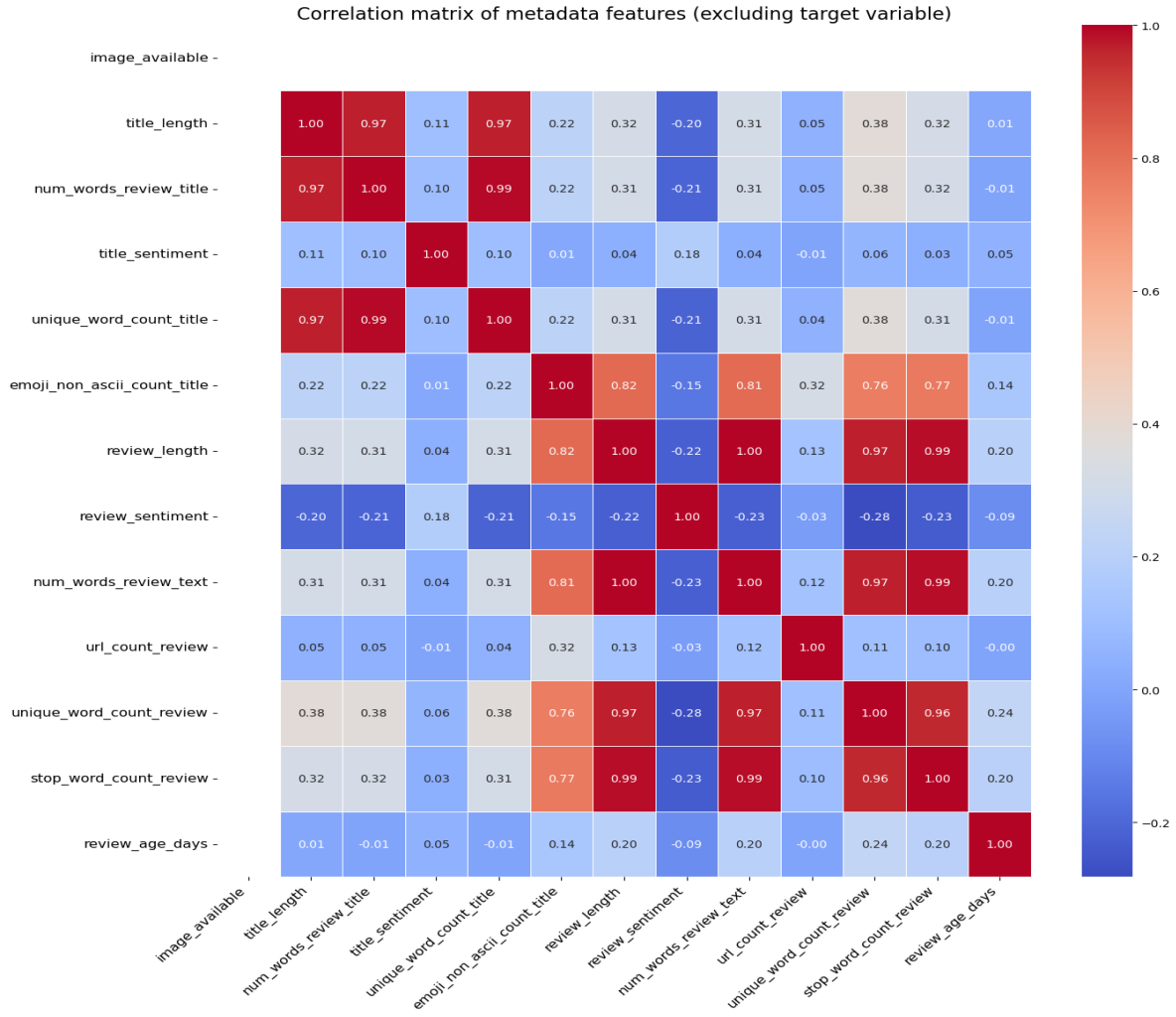


3.5. Data Preparation

Once the data is preprocessed and transformed, it is ready for modeling. The modeling dataset combines the processed metadata features and the TF-IDF matrix. The modeling dataset may have several independent features that are correlated, and addressing these is crucial. In a scenario where two features are highly correlated, only one can be taken and the other removed since one feature is a linearly scaled version of the other. The choice of which correlated features to remove is heuristic and depends on our assessment of the relative importance of variables. Since it is not feasible to look for correlated features among the TF-IDF matrix, we only restricted the correlation checks to the metadata features. We have chosen an absolute correlation cut-off of 0.85, which means that any two features that have a positive correlation greater than 0.85 or lesser than -0.85 are said to be highly correlated. The correlation matrix of the metadata features for the Appliances category is shown in Figure 35, and we can observe that several features are correlated which need to be removed.

Figure 35

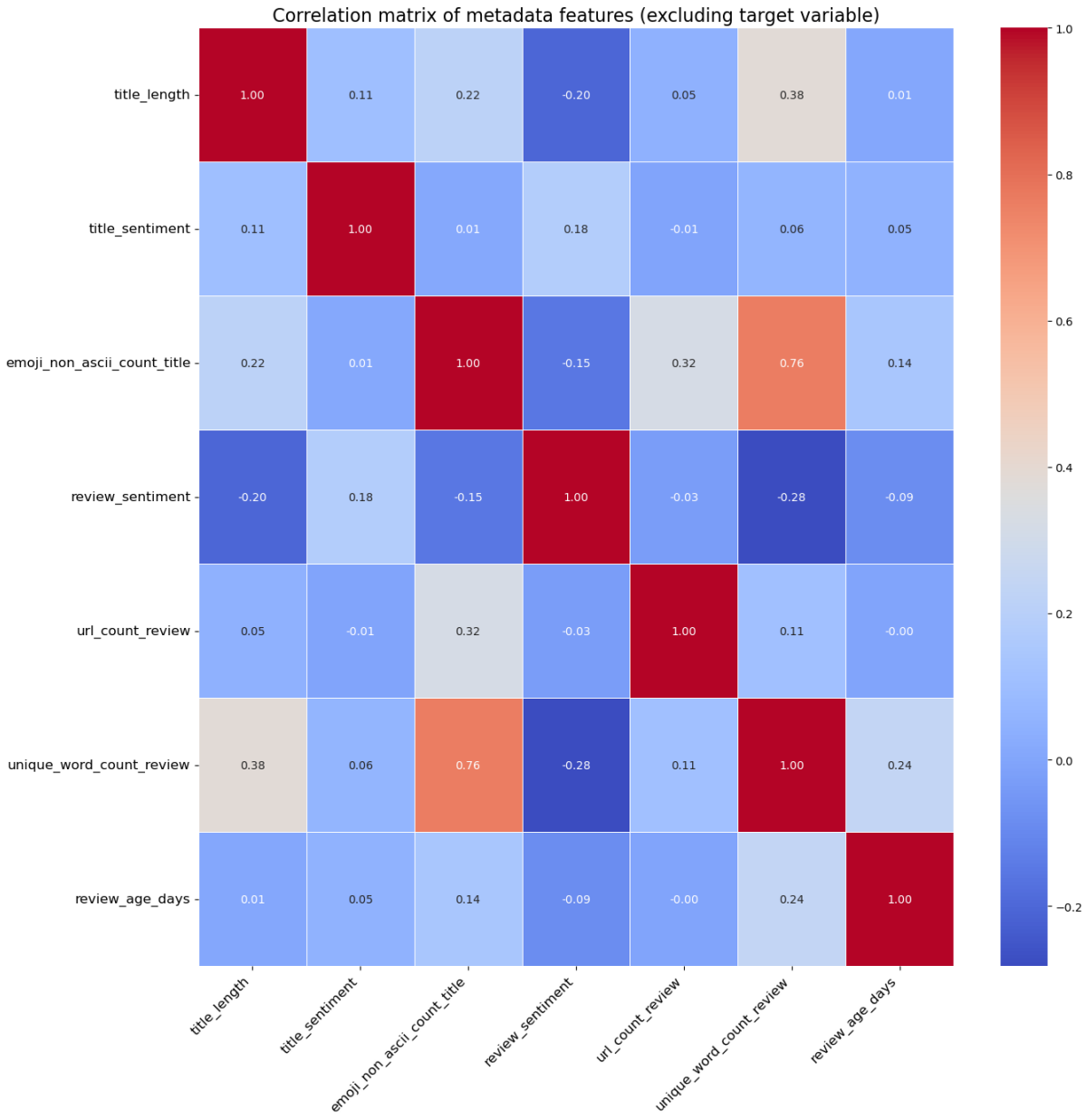
Initial correlation heatmap of metadata features



After studying the matrix and assessing the relative importance of features, we decided to retain only the following metadata features: 'title_length', 'title_sentiment', 'emoji_non_ascii_count_title', 'review_sentiment', 'url_count_review', 'unique_word_count_review' and 'review_age_days'. The final correlation heatmap is shown in Figure 36.

Figure 36

Final correlation heatmap of selected metadata features



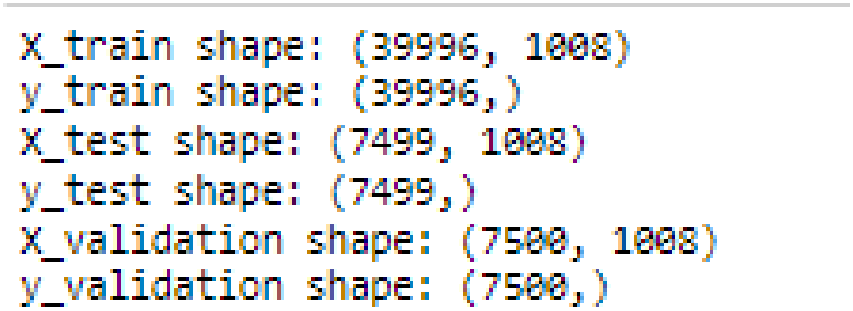
Once we have decided on the final modeling dataset, we split it into X and y. X contains all the features, while y contains the target. Preparing the data for modeling is essential, and we are creating three sets: a training dataset, a testing dataset, and a validation dataset. We split the transformed data into 70:15:15 for training, testing, and validation, respectively. This split will

help us balance training, testing, and validation for modeling below Figure 37 shows the shape of each of the datasets.

As mentioned earlier, we have taken a stratified sample of the overall data as the data is massive for each category, and we have considered 50K representative samples and 1000 top words for TF-IDF. This sample contains the same distribution as the overall sample, as shown in Figure 38. The sample has substantial data required for our baseline model to be trained on and fit. The validation dataset is used to tune the hyperparameters to predict the helpfulness. We can calculate metrics like accuracy, precision, recall, and F1-score for our tuned model and compare it to the baseline model. Once the model is tuned, we can evaluate it on the testing dataset.

Figure 37

Shape of the training, testing, and validation dataset



Note. This is on the stratified sample of the data.

Figure 38

Train, test, and validation sets with 1000 features

```
X_train.head(2)
```

	able	absolutely	access	accurate	across	actual	actually	adapter	add	added	...	yet	youtube	title_length	title_sentiment	emoji_non_ascii_co
31984	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	10	1.0	
47286	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	63	0.7	

2 rows × 1008 columns

```
X_test.head(2)
```

	able	absolutely	access	accurate	across	actual	actually	adapter	add	added	...	yet	youtube	title_length	title_sentiment	emoji_non
1659	0.116317	0.0	0.0	0.0	0.163125	0.0	0.0	0.0	0.0	0.162023	...	0.0	0.0	26	0.166667	
23287	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	...	0.0	0.0	56	0.000000	

2 rows × 1008 columns

```
X_validation.head(2)
```

	able	absolutely	access	accurate	across	actual	actually	adapter	add	added	...	yet	youtube	title_length	title_sentiment	emoji_non_ascii_co
37352	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	4	0.7	
9052	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	10	0.0	

2 rows × 1008 columns

3.6. Data Statistics

Table 17 illustrates the transformation of the datasets through different procedures that are followed in each stage of our project. The raw data set that was gathered included 13,677 records from the Appliances category, 1,711,519 records from the Automotive category, 1,128,437 records from the Cell Phones and Accessories category, and 2,070,831 records from the Tools and Home Improvement category.

Table 17

Summary of data sizes after every step of data processing

Stage	Process	Category	Rows x Columns
Raw data collection	--	Appliances dataset	602777 x 12
		Automotive dataset	1711519 x 12
		Cell Phones and Accessories	1128437 x 12
		Tools and Home Improvement	2070831 x 12
Data Preprocess	After Deduplication	Appliances	591371 x 12
		Automotive	1647280 x 12
		Tools and Home Improvement	1982666 x 12
		Cell Phones and Accessories	1124986 x 12
Data Transformation	Feature extraction using TF-IDF	Appliances	49995 x 1000
		Automotive	49872 x 1000
		Cell Phones and Accessories	50006 x 1000
		Tools and Home Improvement	49996 x 1000
Data Preparation	Training set	Appliances	39996 x 1008
		Automotive	34910 x 1008
		Cell Phones and Accessories	35004 x 1008
		Tools and Home Improvement	39996 x 1008
	Validation set	Appliances	7499 x 1008
		Automotive	7481 x 1008
		Cell Phones and Accessories	7501 x 1008
		Tools and Home Improvement	7499 x 1008
	Testing set	Appliances	7500 x 1008
		Automotive	7481 x 1008
		Cell Phones and Accessories	7501 x 1008

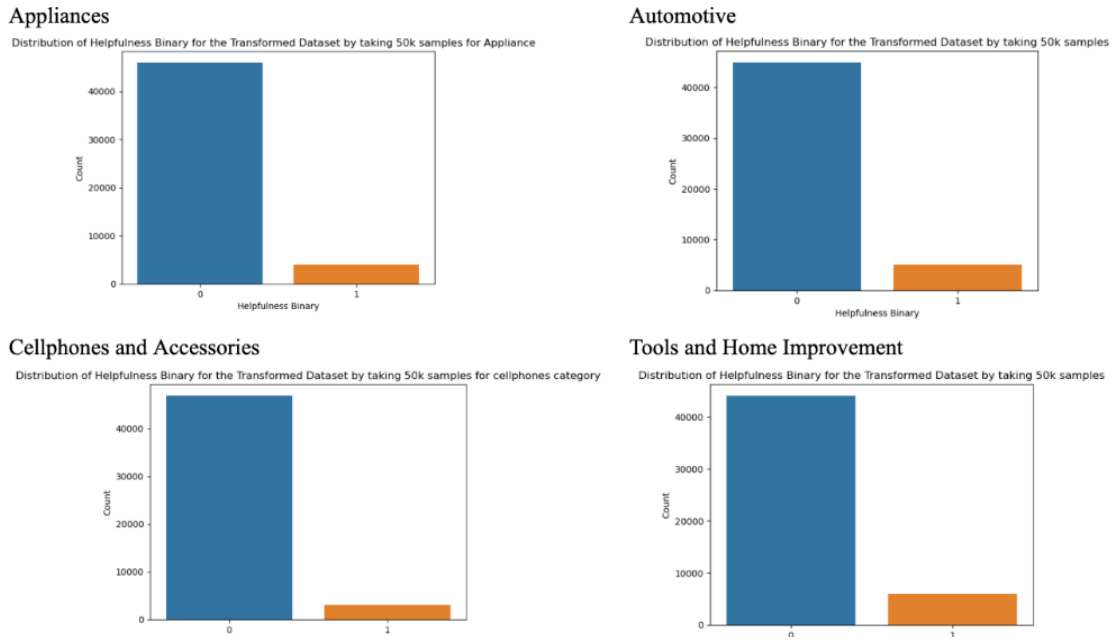
Tools and Home Improvement	7500 x 1008

First, unprocessed data is gathered, where the number of rows and columns indicates the amount of the dataset for each category. Data preparation modifies the dataset dimensions by removing duplicates, lemmatizing terms to their basic forms, etc. After that, TF-IDF feature extraction is used to convert the data, creating a homogeneous feature space with 1000 columns for every category. Ultimately, the data is separated into training with 70%, testing and validation, each with 15% sets, each of which has a distinct function in the creation and evaluation of machine learning models. Notably, the Tools and Home Improvement category has a testing set that is significantly larger than the others. This methodology guarantees a thorough examination of the evaluations with the goal of using machine learning techniques to extract significant insights.

Following the selection of 50,000 samples, the distribution of binary helpfulness labels in the dataset for various categories is displayed in a bar chart as shown in Figure 39. There are more reviews rated as not helpful (0) than as helpful (1), as seen by the left bar which is labeled ‘0’ than the right bar which is labeled as ‘1’. This implies that there is a notable imbalance in the sampled data, with the majority of reviews marked as not unhelpful.

Figure 39

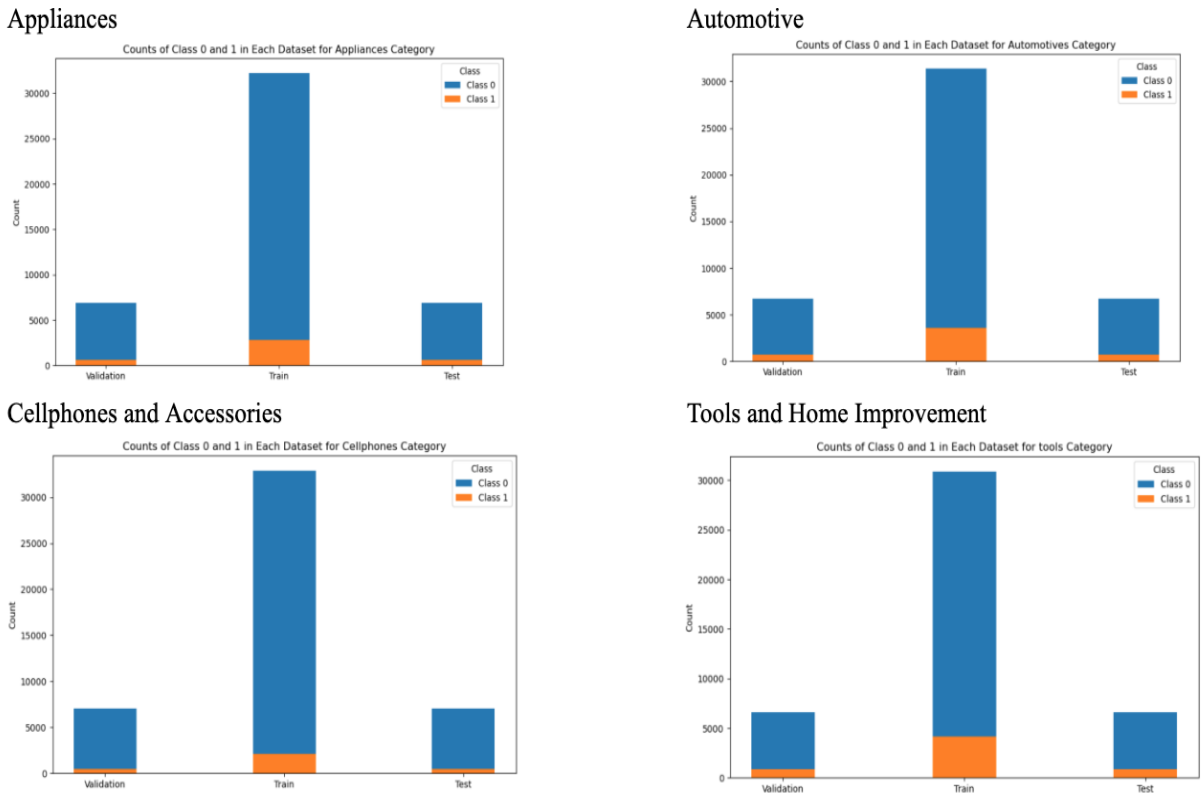
Bar Chart showing the distribution of Helpfulness Binary for the Transformed Dataset by taking 50k samples for various categories



The Figure 40 shows a bar chart of the counts of two classes, labeled class 0 and class 1, across three different datasets, Test, Train, and Validation, for the various categories of Amazon product reviews. The bars indicate the number of instances in each class for each dataset. Class 0 has a much larger count in the training set compared to class 1, which suggests an imbalance in the dataset. The test and validation sets have relatively smaller counts, which is typical in machine learning where the majority of data is used for training, and the rest is divided into validation and test sets.

Figure 40

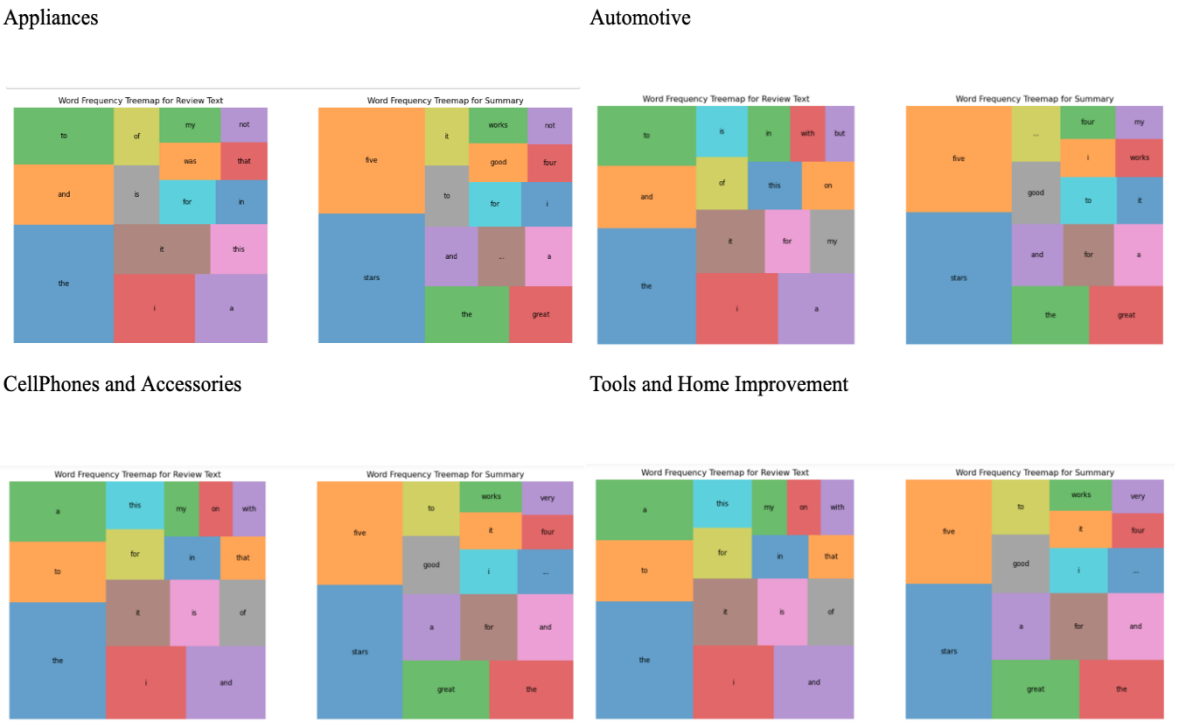
Bar charts showing count of Class 0 and 1 in Train, Valid and Test Dataset for various categories



The Figure 42 represents pie charts displaying the frequency of the top 15 non-stop words in product review texts and summaries for various Amazon categories, following the removal of stop words. Based on the visualizations, we can see these charts provide insight into the most common terms used by customers in their reviews. For instance, words like ‘great’, ‘easy’, ‘perfect’ are frequently mentioned suggesting the areas of customer focus or satisfaction. These terms can highlight what customers value most in their products and can be used to understand areas for product improvement.

Figure 41

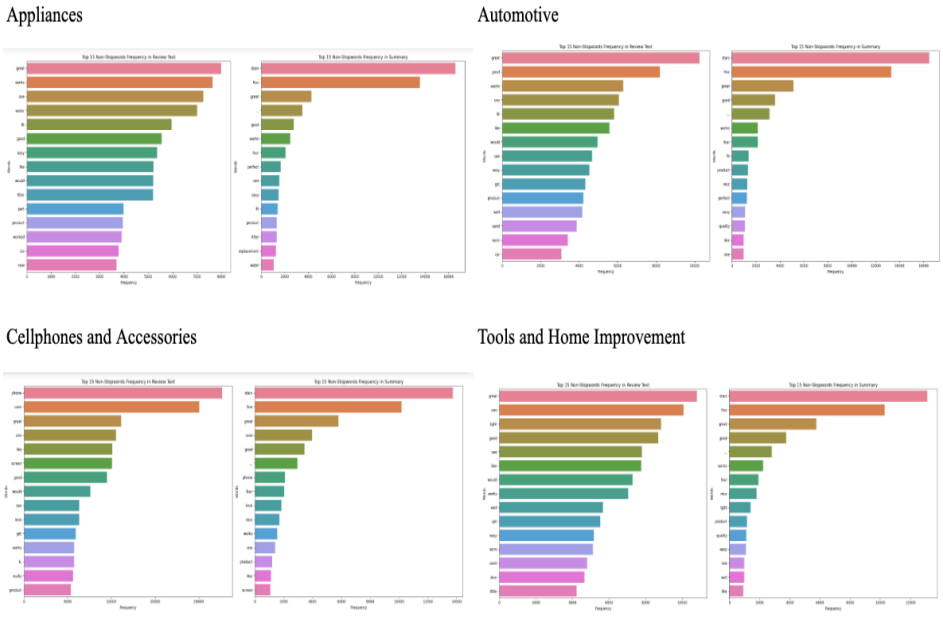
Word Frequency Treemap for review text and summary



The Figure 42 represents pie charts displaying the frequency of the top 15 non-stop words in product review texts and summaries for various Amazon categories, following the removal of stop words. Based on the visualizations, we can see these charts provide insight into the most common terms used by customers in their reviews. For instance, words like ‘great’, ‘easy’, ‘perfect’ are frequently mentioned suggesting the areas of customer focus or satisfaction. These terms can highlight what customers value most in their products and can be used to understand areas for product improvement.

Figure 42

Bar charts showing top 15 non-stopwords frequency in review text and summary

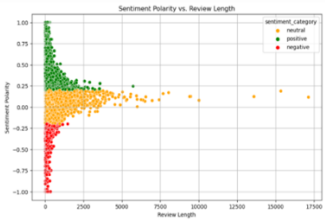


The Figure 43 consists of scatter plots for various Amazon product categories, showing the relationship between review length and sentiment polarity. Positive sentiments are more frequent in shorter reviews, while negative sentiments appear across all lengths, and neutral sentiments are less common overall. The trend suggests that people often write positively in brief reviews, while longer reviews may contain more detail, often associated with negative feedback.

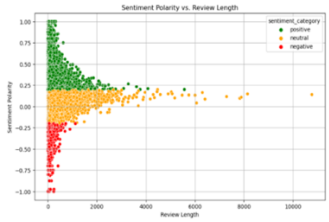
Figure 43

Scatterplot showing Sentiment Polarity vs. Review Length for various categories

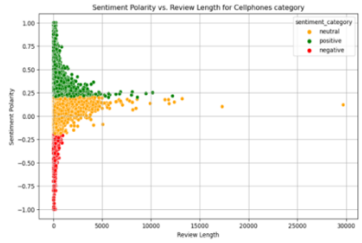
Appliances



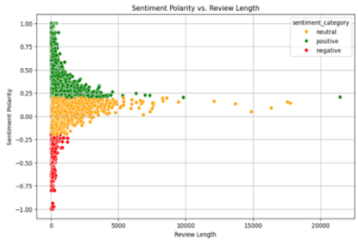
Automotive



CellPhones and Accessories



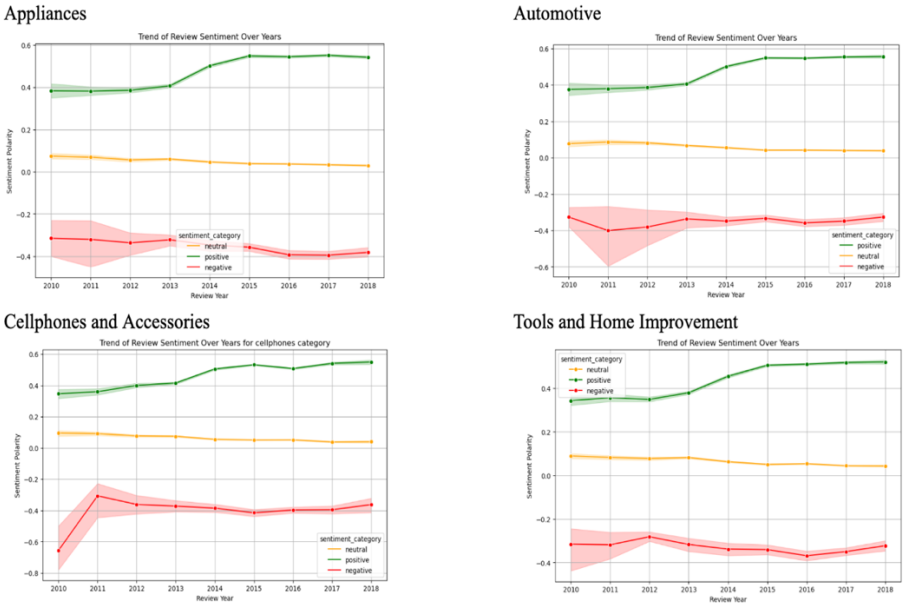
Tools and Home Improvement



Line graphs depicting the sentiment of reviews over a period of years for different product categories on Amazon are displayed in the Figure 44. The sentiment polarity (positive, neutral, and negative) of each graph is plotted against time to show how consumer sentiment has evolved over time. The high percentage of positive ratings indicates that customers are generally satisfied. While the categories for Automotive, Tools and Home Improvement indicate progress, there is a small increase in negative sentiment in the Appliances, Cellphones and Accessories categories, suggesting dissatisfaction in recent years. Neutral sentiment remains relatively unchanged.

Figure 44

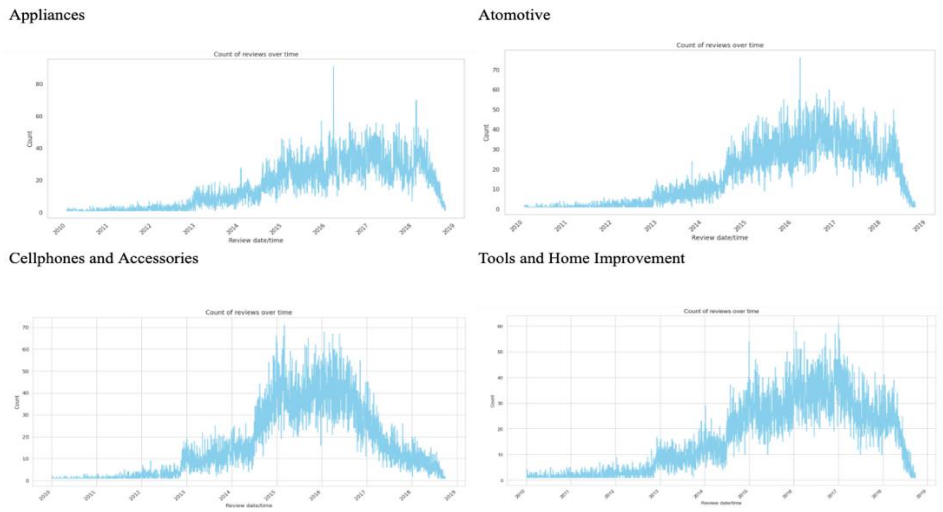
Line graphs showing the Trend of Review Sentiment Over Years



The trend from 2010 to 2019 is represented by each chart in Figure 45, where the y-axis displays the number of reviews and the x-axis displays the review date and time. There is a noticeable increase in reviews in the Appliances category about 2016, which is then followed by a slow decline. The number of reviews in the Automotive category peaked about 2015 and then sharply declined. The market for Cellphones and Accessories peaks sharply in 2016, indicating a time of increased consumer interest or new product launches. The Tools and Home Improvement chart shows a fall in reviews after a high increase in reviews in 2017.

Figure 45

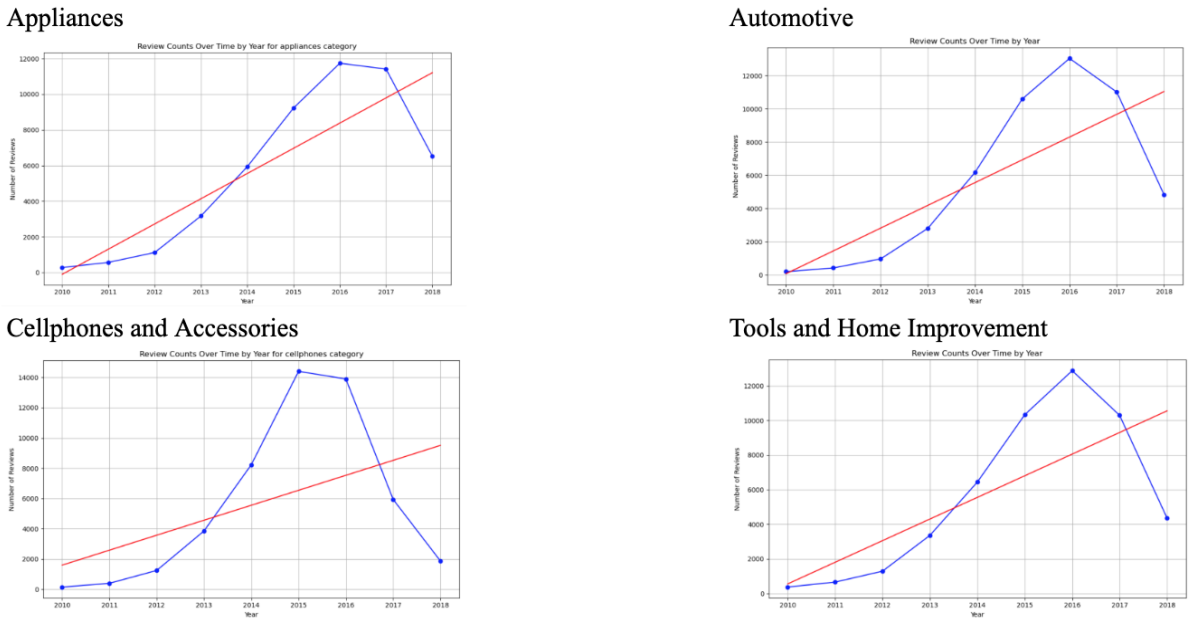
Line charts showing trends in consumer review volumes across various categories



Line graphs representing the number of reviews for several Amazon product categories over time and by year are displayed in Figure 46. Every graph shows the count of reviews between 2010 and 2018, which reflects patterns in consumer buying and reviewing habits. There are notable peaks in a few categories, signifying years with a large spike or drop in the number of reviews. Regardless of the annual variations, the red line most likely depicts a trend line that illustrates the general direction and pattern in review volume over time.

Figure 46

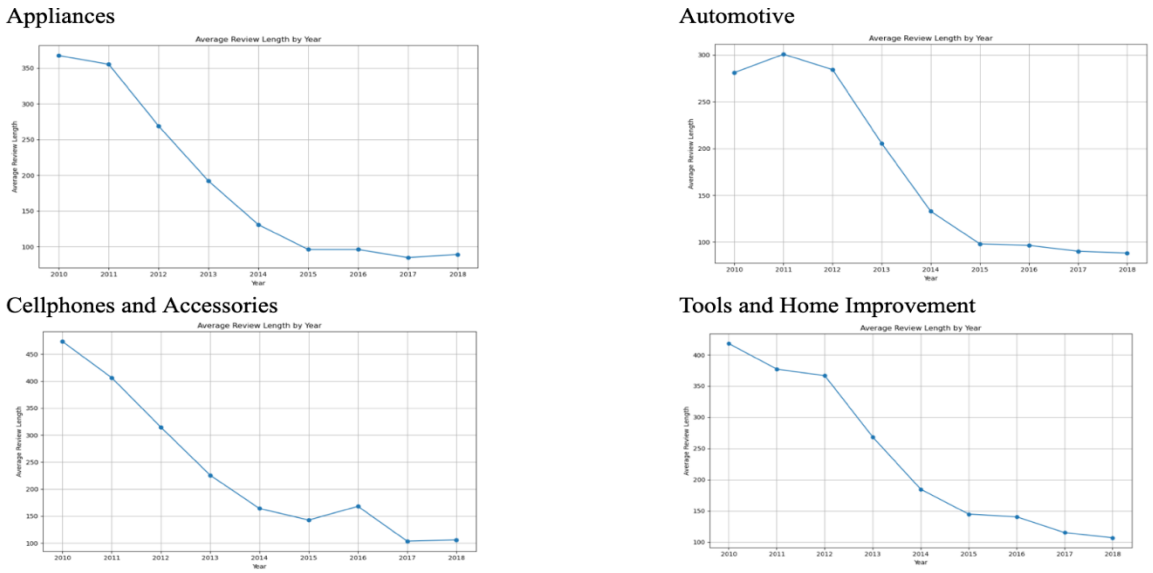
Line graphs showing Review Counts Over Time by Year for various categories



Line graphs illustrating the average review length trend by year for various Amazon product categories are displayed in Figure 47. The line in each graph shows how, between 2010 and 2018, the average number of characters or words per review changed. Every category shows a distinct downward trend, suggesting that customer ratings have been shorter over time. This pattern is evident in all of the categories that are on show, which may indicate a shift in platform guidelines or customer review behavior.

Figure 47

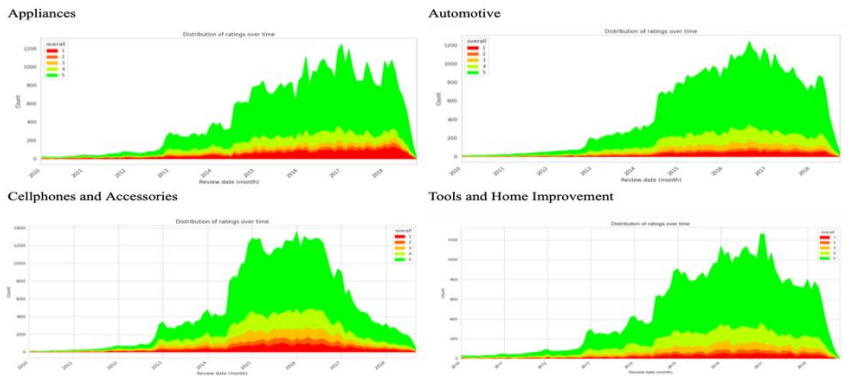
Line graphs showing Average Review Length by Year



The distribution of ratings 1 through 5 over time is illustrated by the stacked area chart presented in Figure 48. Various colors are allocated to each rating level, with red denoting the lowest rating (1) and green denoting the highest rating (5). The green area on the charts, which represents the highest rating, predominates and indicates a general trend of favorable customer feedback in all areas. Interestingly, review volume peaks for each category coincide with major market events, including product launches or holiday sales.

Figure 48

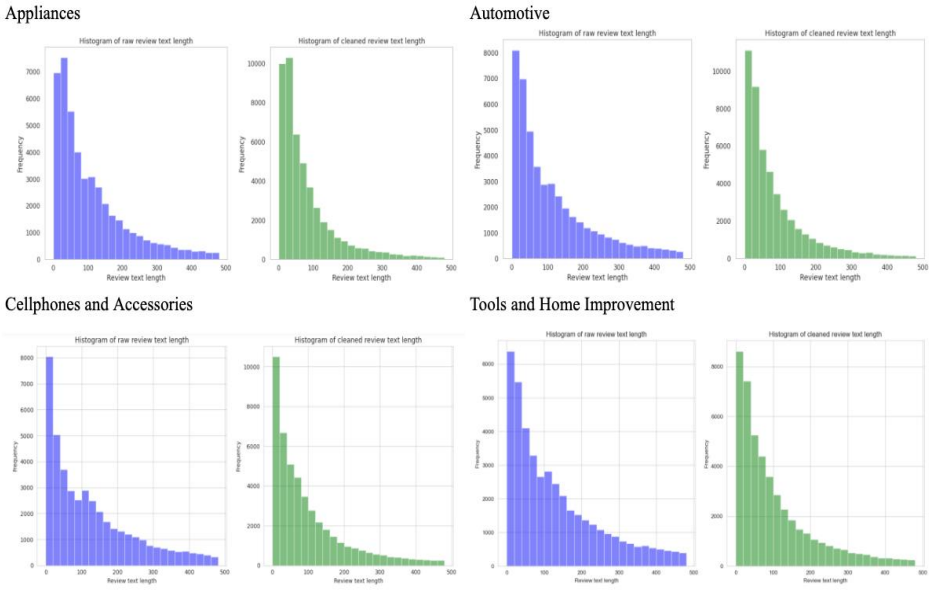
Stacked area charts showing trends in customer ratings over time for various categories



These are color-coded, as seen by the histograms shown in Figure <>, where blue denotes raw text and green, cleaned text. The pre and post histograms effectively compare the length of review texts before and after the cleaning process. Notably, there is an increased concentration of reviews with shorter lengths post-cleaning, indicating that the cleaning steps significantly reduce the text length. This reduction is largely due to the removal of stopwords, URLs, emojis, and other non-essential elements. Both histograms show right-skewed distributions in all categories, with most reviews having fewer characters and these numbers gradually declining as the text lengthens. Longer texts typically exhibit a sharper decline in frequency in the cleaned text histograms, which suggests that irrelevant material was eliminated during the cleaning process. Both histograms are synchronized in terms of their x and y axes, which enhances the visibility of the changes brought about by the cleaning process. This alignment allows for a clear visual comparison, showing just how much content is removed during the cleaning stage.

Figure 49

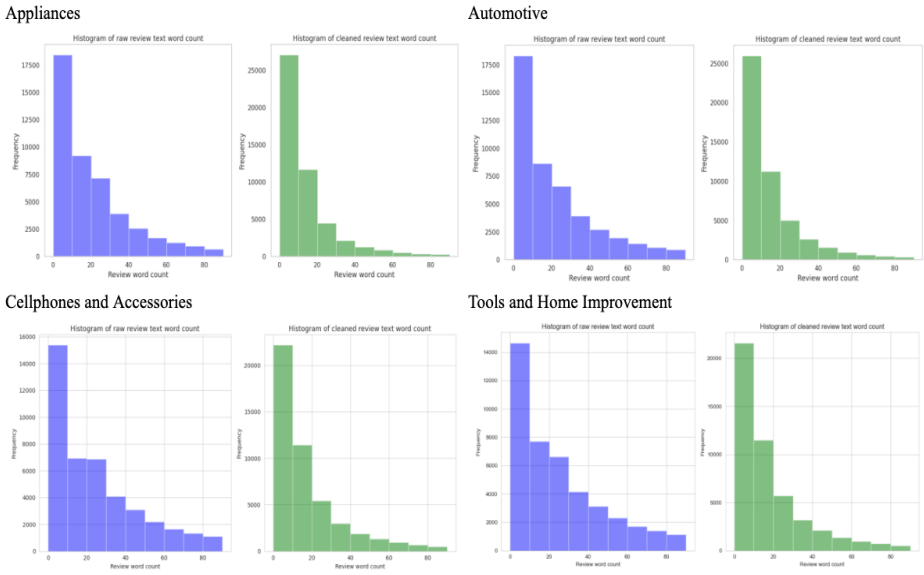
Histograms showing raw and cleaned review text lengths across various categories



Histograms showing the word count distribution in raw and cleaned review texts for different categories are shown in Figure 50. Two histograms are included for each category where the raw review text is shown in blue, and the cleaned review text is shown in green. These histograms, which are divided into bins that span word counts from 0 to more than 80, display the frequency of reviews based on word count. The histograms show that most raw evaluations in each category have a low word count, with frequency dramatically decreasing as word count rises. After cleaning the text, which involves removing irrelevant words or noise, the distribution becomes even more skewed towards shorter reviews.

Figure 50

Histograms showing word count distribution in raw and cleaned reviews across various categories



We have also performed a t-test on a specific metadata feature to determine if the differences between helpful and non-helpful reviews are statistically significant. By comparing the values of a feature between these two types of reviews, the t-test helps establish whether the observed differences in means are not due to random chance. If the resulting p-value is less than

0.05, we can reject the null hypothesis, which states that there is no significant difference between the means. Consequently, a p-value below this threshold indicates a statistically significant difference between the values of the feature in helpful versus non-helpful reviews. Thus, there is a statistically significant difference between helpful and non-helpful reviews for the title_length feature.

Modeling

4.1. Model Proposal

4.1.1. Logistic Regression

Logistic regression is popular in machine learning as it provides probabilities and classifies new samples using discrete measurements. It uses supervised learning techniques for classifying the dependent variable, which is binary. For our project, the dependent variable is helpfulness. As logistic regression outputs are discrete values, that is, if the variable has only two possible outcomes, in our project, it's helpful or not helpful. It makes sense to use logistic regression to predict either 0 or 1 or a probability score between 0 and 1.

The sigmoid or logistic function gives the probability values between 0 and 1. It squashes the outcomes of a linear equation between 0 and 1. This function fits an S-shaped curve, which tells whether the probability of a review is helpful based on the weighted input, which is metadata and TF IDF scores of words. The y-axis on the sigmoid graph is the probability (review helpfulness). This logistic function or sigmoid function is defined by (1)

$$\sigma(z) = \frac{1}{1 + e^{(-z)}} \tag{1}$$

In the equation (1), z is the combination of metadata and transformed text data which are input features along with their respective regression coefficient weights as described in (2).

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{2}$$

Linear hypothesis when used in logistic sigmoid function we get (3), here θ is the vector of parameters (coefficients) of the model and input feature matrix is x. This function returns the probability of y = 1 for given x which is parameterized by θ in (4).

$$h_\theta(x) = \frac{1}{1 + e^{(-\theta^T x)}} \tag{3}$$

$$h(x) = P(y = 1|x; \theta) \quad (4)$$

For h(x), Values that are greater than or equal to 0.5 are predicted as 1, and values that are less than 0.5 are predicted as 0. To explain the relationship among input and target features we use maximum likelihood estimation. This gives the best-fitting model. The cost function of logistic regression is (5).

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}[y^i \, log(h_\theta(x^{(i)})) + (1 - y^i) log(1 - h_\theta(x^{(i)}))] \quad (5)$$

To find optimal parameters, gradient descent is used, and this will minimize the cost function of our model. It uses an iterative method and is given as (6).

$$\theta_{new} = \theta_{old} - \alpha \frac{\partial J(\theta)}{\partial \theta_j} \quad (6)$$

The pseudo algorithm for logistic regression is shown in the figure 51 and the architecture for logistic regression is shown in figure 52.

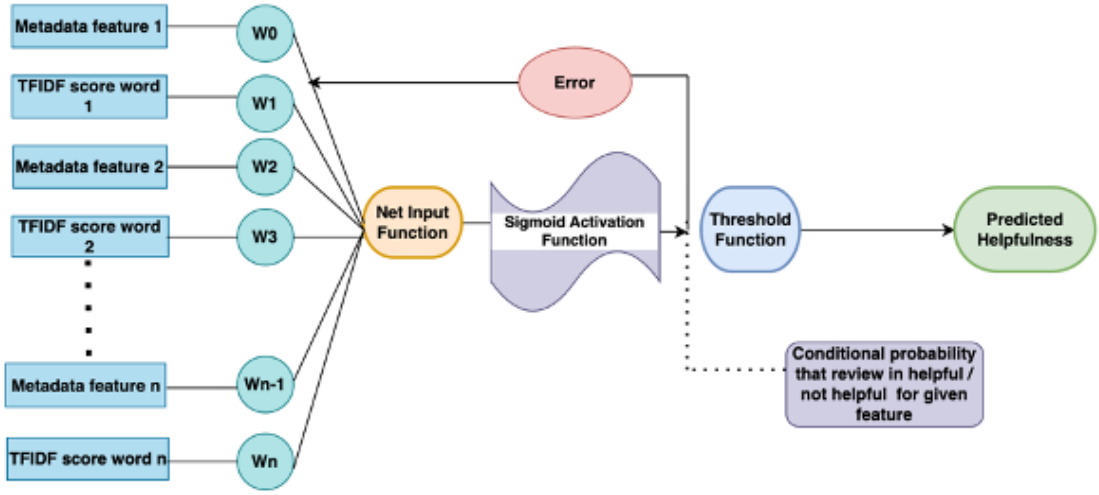
Figure 51

Algorithm of Logistic Regression Model

Input:
$x^{(i)}$: Feature vector of the i^{th} training sample
y^i : Corresponding label (0 or 1)
n: Number of records
Initialization:
Initialize parameter θ to 0 or some small value.
Define the hypothesis function $h_\theta(x)$ using the sigmoid function.
Define cost function J(θ) using the logistic loss/cost function.
Set iteration counter iter = 0
Set convergence criterion (e.g., threshold for change in cost function)
Set maximum number of iterations (optional)
Gradient Descent:
Repeat until convergence or maximum iterations reached:
  For i = 1 to n:
    Compute the hypothesis value for sample i: $h_\theta(x^{(i)})$
    Compute the error (predicted - actual): error = $h_\theta(x^{(i)})$ - y^i
    Update parameters θ using gradient descent: $\frac{\partial J(\theta)}{\partial \theta_j}$ for j = 0 to num_features
  Compute the cost function J(θ) using the updated parameters.
  Increment iteration counter: iter++
Output:
Return the optimized parameters θ.

Figure 52

The architecture of the logistic regression model for review helpfulness



4.1.2. Support Vector Machine (SVM)

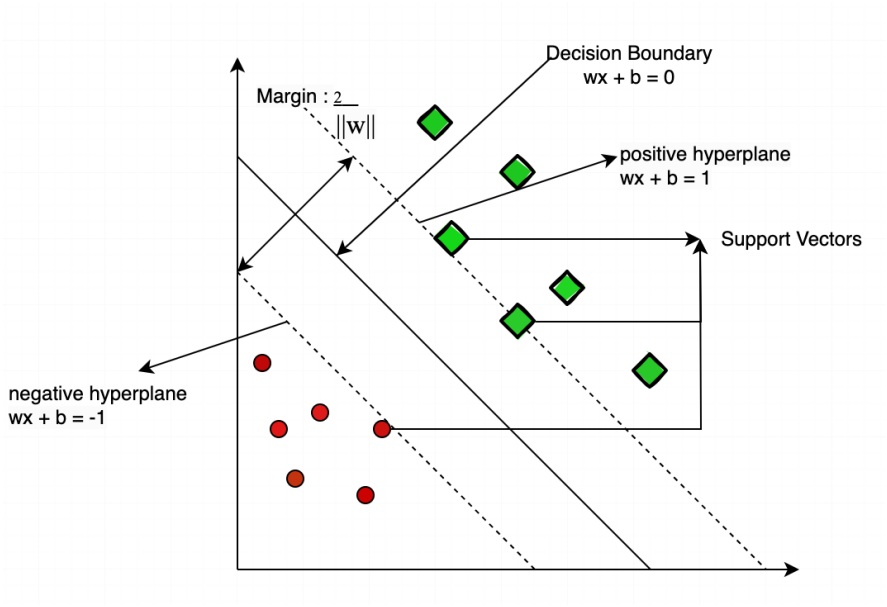
Support Vector Machines (SVM) is the most elegant, popular, and successful method for solving classification problems. Some examples of SVM usage are face recognition, spam filtering, image classification, and stock price prediction. SVM is predominantly utilized for solving classification tasks, although they can also be employed for regression problems.

SVM performs the classification test by creating a decision boundary represented as a hyperplane (i.e., a line in 2D or a plane in 3D) to separate the n-dimensional data points into distinct classes. For instance, distinguishing between true and false tweets involves categorizing new tweets correctly. SVM facilitates this classification by ensuring all points from one category reside on one side of the hyperplane (line for tweet example), and those from the other category occupy the opposite side. While multiple hyperplanes (lines here) may exist that can separate two classes, SVM aims to identify the hyperplane that optimally separates the two classes in such a way that the hyperplane will be as far as possible from the nearest datapoint in both classes. It aims to maximize the margin between the two classes. This distance between the two classes is

called the margin, and the decision boundary is the line dividing the maximum margin into equal halves. We know that every row of the dataset is a vector in space. So, the vectors (points) that fall on the margins are called the supporting vectors. Based on the support vector, we come up with a decision boundary and classify the new data point into the correct category. This is the reason it is called support vector machines. Figure 53 shows the illustration of Linear SVM.

Figure 53

Linear Support Vector Machine Illustration



The goal is to find a hyperplane that maximizes the margin, which is the distance between the hyperplane and the closest point on either side of each class. This is done under the constraint that all data points lie on the correct side of the hyperplane. The margins are written in terms of the linear equation shown above since the data is linearly separable. The distance between the hyperplane is called the margin and the middle hyperplane is the max-margin classifier.

The equation for hyperplanes is shown in (7), (8), (9).

Hyperplane $$w^T.x_i + b = 0 \quad (7)$$

Negative Hyperplane $$w^T.x_i + b = -1 \quad (8)$$

Positive Hyperplane $$w^T.x_i + b = 1 \quad (9)$$

To handle non-linear data, SVM uses a technique called the kernel trick. The kernel trick maps the data into a higher-dimensional feature space where the data becomes linearly separable. This is achieved by using kernel functions, such as the radial basis function (RBF) kernel or the polynomial kernel which finds the similarity between data points in the higher-dimensional space. The SVM then finds the optimal hyperplane in this transformed feature space, allowing it to model non-linear decision boundaries.

4.1.3. Random Forest

Compared to a single decision tree, the primary goal of the Random Forest is to increase prediction accuracy while preserving robustness and lowering the risk of overfitting. In order to do this, it builds a large number of decision trees during training and outputs the class that represents the majority vote of the classes that each tree predicted. By successfully capturing a variety of patterns and correlations in the data, this ensemble method makes use of the strengths of several learners to achieve higher performance on complicated datasets. This method can be used for both regression and classification. The metrics like entropy and Gini impurity are commonly employed in classification to assess the split quality. With the goal of reducing impurity across the network, these metrics calculate the impurity inside each group. For more precise classification, a more homogeneous grouping is indicated by lower impurity levels. In classification, recall, accuracy, and precision are additional metrics that are employed. The formula to calculate the Gini index is shown in (10).

$$Gini(p) = 1 - \sum_{i=1}^{n} p_i^2 \tag{10}$$

where pi is the proportion of samples belonging to class i. The objective is to minimize the Gini impurity across all branches of the decision tree.

The formula used to calculate the entropy is shown in the (11).

$$E(S) = -\sum_{i=1}^{n} p_i \ log_2 p_i \tag{11}$$

where pi represents the proportion of data points in class i within the set S. The summation goes over all classes in the dataset.

The formula used to calculate the accuracy is shown in the (12).

$$Accuracy = \frac{Number\ of\ predictions}{Total\ number\ of\ predictions} \tag{12}$$

The formula to calculate precision is shown in the (13).

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

where, TP is the number of true positives and FP is the number of false positives.

The formula to calculate recall is shown in the (14).

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

where, FN is the number of false negative. The pseudo algorithm for Random Forest is shown in the Figure 54.

Figure 54

Algorithm for Random Forest

Pseudocode for creating Random Forest

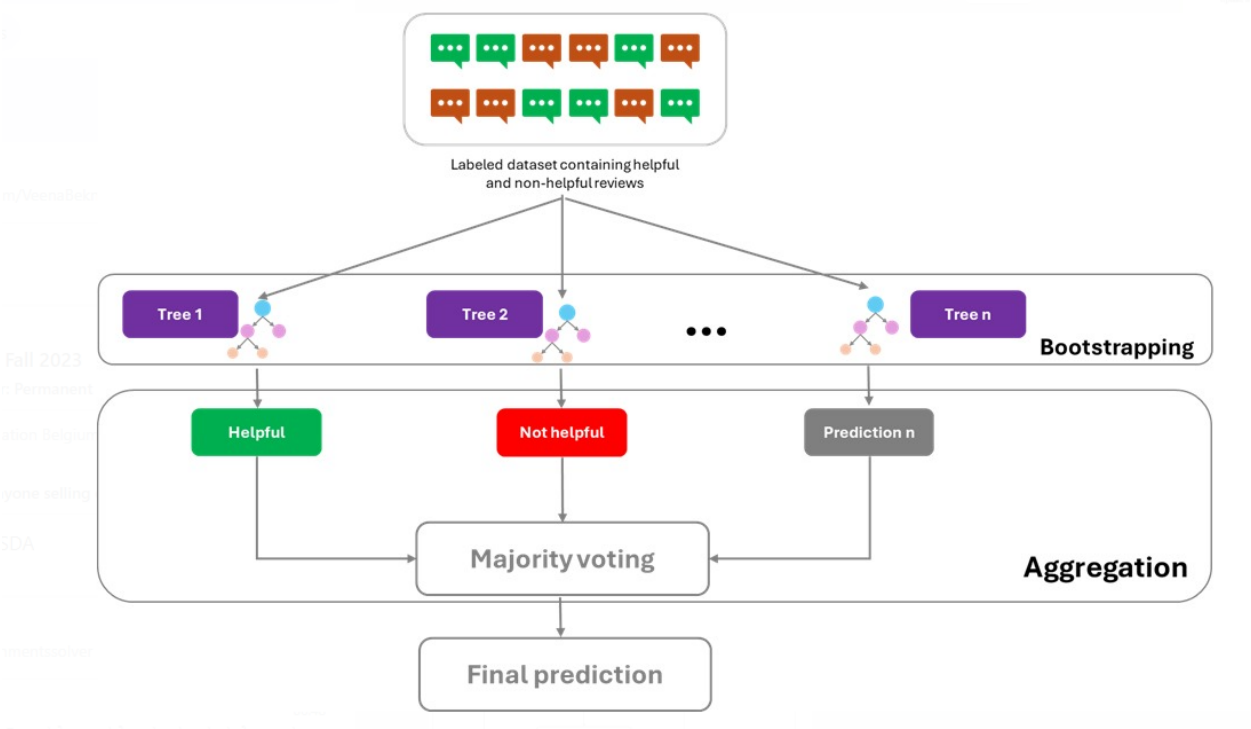
1. Create an empty forest.
In the case of i = 1 to n_estimators:
   2.1 Create a training data bootstrap sample.
   2. Start a decision tree from scratch.
   2.3 For every decision tree node:
      2.3.1 Choose max_features features at random from all of the features.
      2.3.2 Using metrics like entropy or Gini impurity, determine the optimal split based on the chosen features.
      2.3.3 Divide the parent node into its two offspring.
      2.3.4 Continue until max_depth is reached or min_samples_split or min_samples_leaf stops more splitting.
   2.4 Include the built tree in the woodland.
3. To forecast the outcome for a fresh sample:
   3.1 Assign a prediction to every tree in the forest.
   3.2 From all of the trees' predictions, choose the final prediction by majority voting (for classification).
4. Return to the Random Forest model of the forest.

Figure 55 provides a detailed illustration of the Random Forest algorithm, a dependable ensemble learning technique that is frequently employed in machine learning for regression and classification problems. The Random Forest algorithm's structure is depicted in the diagram, which is used to categorize reviews as ‘helpful’ or ‘not helpful’. It starts with a tagged dataset of reviews that have been pre-classified as beneficial or unhelpful. The dataset is bootstrapped, which is a technique whereby several subsets (replacement samples) of the original dataset are made in order to train each forest tree separately. As a result, multiple decision trees—referred to as Tree 1 through Tree n—are produced, each of which is trained using a distinct bootstrap sample. Every one of these trees independently forecasts whether or not the reviews will be beneficial. The ultimate classification for each review is decided by the majority vote across all

trees once these predictions are gathered and combined through a majority voting method. By limiting variance and avoiding overfitting, the ensemble technique improves the model's robustness and accuracy. The final prediction ‘helpful’ or ‘not helpful’, represents the consensus of the forest.

Figure 55

Random Forest Model: Ensemble Learning through Bootstrap Aggregation and Decision Trees



Note. This is a Random Forest model which is an ensemble learning through Bootstrap Aggregation and Decision.

To forecast how useful reviews will be, a Random Forest is used, which makes use of a wide range of features. The model integrates sentiment analysis scores from tools like NLTK, a bag of words model that counts word occurrences, and textual characteristics like TF-IDF, which highlights unique terms in reviews by comparing their frequency in a document to their corpus frequency. Additionally taken into account are the features which include the quantity of reviews

a user has posted, their average rating, past helpful votes. Reviews are contextualized by features like price, brand, category, and historical sales data. To improve generalization, the Random Forest model constructs several decision trees, each from random features and data samples. As mentioned previously, entropy and Gini impurity—which quantify dataset randomness and label homogeneity, respectively—are used to divide the nodes in each tree. Adjusting parameters like max depth and min samples split helps manage complexity and avoid overfitting. The model's effectiveness and efficiency are guaranteed by hyperparameter tuning, which also includes the number of trees and the subset of features taken into account at each split. The Random Forest Classifier can efficiently prioritize and elevate the most insightful evaluations in the Amazon marketplace which improves user experience.

4.1.4. XGBoost (Extreme Gradient Boosting)

XGBoost (Extreme Gradient Boosting) is a tree-based ensemble machine learning algorithm for supervised learning (can be used for regression and classification). In this project, XGBoost was used for predicting helpfulness by clearly differentiating between helpful and non-helpful reviews. XGBoost relies on a technique called boosting; it is a technique in ensemble modeling wherein the output of each decision tree is input into the next tree, which leads to weak learners getting stronger over multiple iterations. The objective function for XGBoost is a sum of a loss function and a regularization term as shown in the (15).

$$Obj = \sum_{i=1}^{n} loss(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k) \quad (15)$$

where Obj is the objective function to be minimized, n is the number of training instances, $loss(y_i, \hat{y}_i)$ the loss function which measures the difference between the actual value y_i and predicted value $\hat{y}_i$, K is the number of trees, $\Omega(f_k)$is the regularization parameter that penalizes

complex models to avoid overfitting. Instead of learning the tree all at once which makes the optimization harder, an additive strategy is applied.

For binary classification, since helpfulness is binary, the logistic loss function can be used:

$$Logistic\ Loss\ (Binary\ classification)\colon loss(y_i, \hat{y}_i) = -[y_i\ log(\hat{y}_i) + (1 - y_i)\ log\ (1 - \hat{y}_i)] \tag{16}$$

where the actual value is y_i and predicted value is $\hat{y}_i$ and $\hat{y}_{ij}$ is the 1 if sample i belongs to class j, 0 otherwise.

Regularization is represented as

$$\Omega(f_k) = \gamma T + \frac{1}{2}\ \lambda\ \|w\|\ \frac{2}{2} \tag{17}$$

Where γ is the regularization parameter for controlling the number of leaves in the decision trees, T is the number of leaves in tree k, λ is the L2 regularization parameter and $\|w\|\ \frac{2}{2}$ is the L2 norm of the weights in the tree.

For the actual decision tree ensemble construction, there are 3 key steps: Gradient calculation, Tree construction, and Ensembling and updating learners. For a given loss function, the gradient of the loss function with respect to the predicted value g_i and the second derivative (Hessian) of the loss function is denoted as h_i. For gradient calculation, the gradient and Hessian are computed for each training instance i using the below formulae.

$$g_i = \frac{\partial L\ (y_i, \hat{y}_i)}{\partial \hat{y}_i^{\ 2}} \tag{18}$$

$$h_i = \frac{\partial^2 L\ (y_i, \hat{y}_i)}{\partial \hat{y}_i^{\ 2}} \tag{19}$$

For the Tree construction, given the gradients g_i and Hessians h_i for each training instance, a new tree is fit to the negative gradient residuals. For each leaf j in the tree, the optimal weight w_j is computed by minimizing the following objective function for that leaf.

$$Obj = \frac{1}{2}\sum_{i\in I_j}\left(\frac{g_i}{h_i}\right)^2 + \lambda \|w\| \frac{2}{2} \tag{20}$$

Where I_j is set of indices of training instances assigned to leaf j, λ is the L2 regularization parameter and the optimal weight w_j is obtained by taking the derivative of the objective function with respect to w_j.

Finally, after constructing each new tree, it is added to the ensemble with a calculated step size (learning rate) to prevent overfitting. The prediction of the ensemble for a given instance is computed by summing the predictions of all trees in the ensemble which is the core boosting element.

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \tag{21}$$

where $\hat{y}_i$ is the predicted value for instance i, $f_k(x_i)$ is the prediction of tree k for instance i and K is the total number of trees in the ensemble. The pseudo algorithm for XGBoost is shown in Figure 56

Figure 56

Algorithm for XGBoost

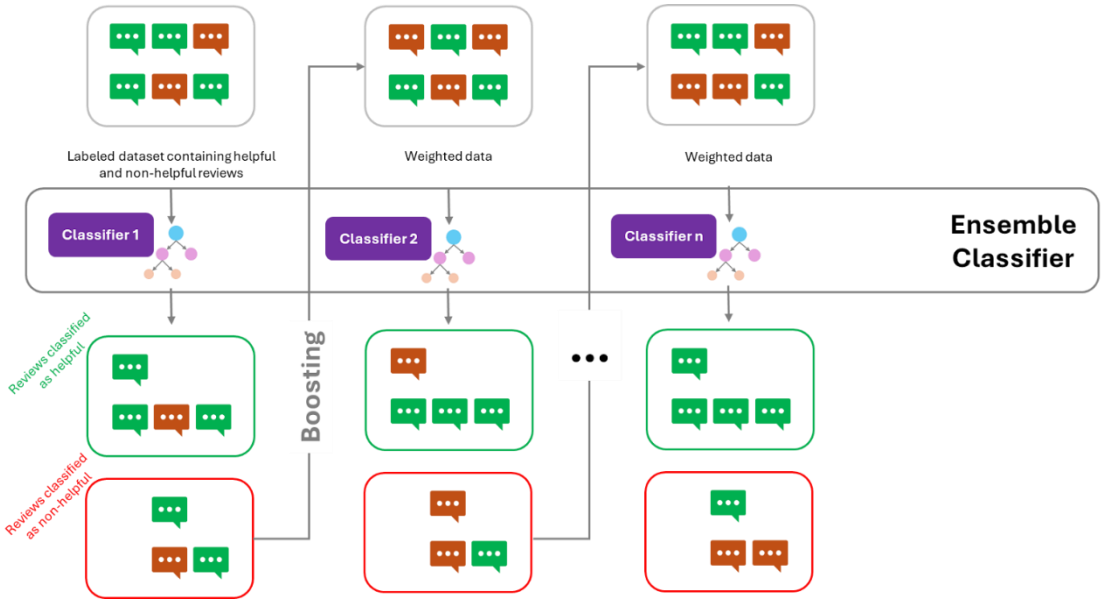
```
Input:
- Training data (X_train, y_train)
- Hyperparameters (e.g., learning_rate, max_depth, n_estimators)
- Loss function (e.g., logistic loss, squared loss)

Algorithm:
1. Initialize an ensemble model as an empty list of trees.
2. Initialize predictions for training data as an array of zeros
3. For each iteration (t = 1 to n_estimators):
   a. Compute the gradient of the loss function for the predictions (gradient):
      gradient = ∂L(y_true, y_pred) / ∂y_pred
   b. Compute the Hessian of the loss function for the predictions (hessian):
      hessian = ∂^2L(y_true, y_pred) / ∂y_pred^2
   c. Fit a decision tree to the negative gradient and Hessian:
      tree_t = fit_tree(-gradient, hessian)
   d. Update predictions by adding the predictions from the current tree multiplied by a learning rate:
      predictions += learning_rate × predict(tree_t, X_train)
   e. Add the fitted tree to the ensemble model.
      ensemble_model.append(tree_t)
4. Output the ensemble model.

Prediction:
- To make predictions for new data:
  a. Initialize prediction as an array of zeros
  b. For each tree in the ensemble model:
     i. Compute the prediction from the tree:
        prediction_t = predict(tree, X_new)
     ii. Add the prediction to the overall prediction
        prediction += prediction_t
  c. Output the final prediction
```

Figure 57

The architecture of the XGBoost algorithm for predicting review helpfulness



4.2. Model Supports

4.2.1 Environment, Platform, and Tools

We have used the Alienware AMD Ryzen 9 7950X 16-core processor, 4501 MHz high-performance computing (HPC) machines at SJSU campus, Google Collab Pro, and local machines for developing and running machine learning algorithms. Due to the convenience, data cleaning and transformation were performed on local machines initially. When we finished the preprocessing, the entire code including modeling was transferred to the HPC for faster execution specifically for the model training and hyperparameter tuning process. As our data was huge, getting a stratified sampling for the data was easy on high compute resources. For our entire project, we have used Python as our main programming language. All the coding was done on Jupyter Notebooks because of its interactive and collaborative development process. Some part of preprocessing was done in Collab pro and execution of the model was done in HPC lab as we needed computational power. We have made use of its wide range of libraries which are specifically designed for machine learning as shown in Table 18.

Table 18

Libraries used for review helpfulness.

Libraries	Usage
pandas	For data manipulation and analysis
NumPy	For numerical computing
matplotlib.pyplot	Data visualization
random	Generating random numbers
plotly	Interactive plots
seaborn	Data visualization
squarify	Tree map plots
Counter	Counting hashable objects
nltk	Natural language processing
contractions	Expanding contractions in text
sklearn.feature_extraction.text.TfidfVectorizer	Text feature extraction
TextBlob	Sentiment analysis
emoji	Working with emojis
sklearn.model_selection.train_test_split	Splitting data into train, validation, and test sets.
XGBClassifier	XGBoost model
sklearn.metrics.classification_report	Classification Metrics
sklearn.metrics.accuracy_score,	Model evaluation

Note. These are specific libraries used for our project.

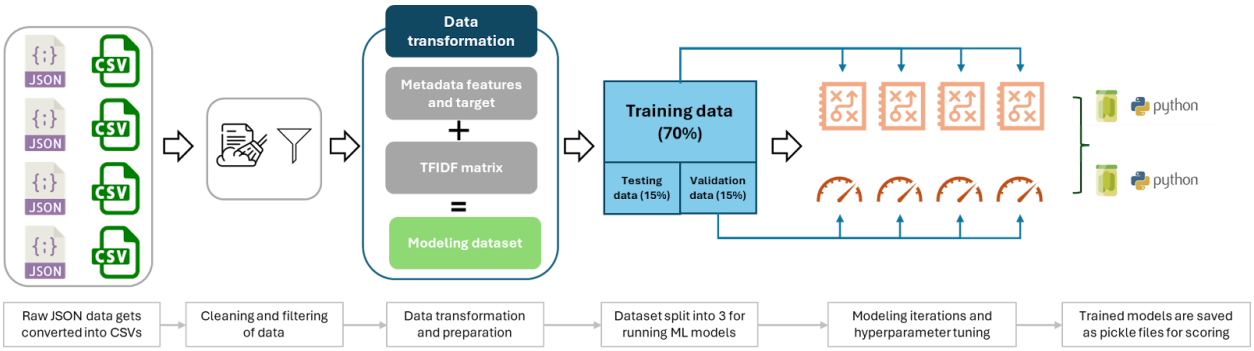
4.2.2. Model Architecture and Dataflow

The first step is to convert the raw JSON to CSV which simplifies the data handling and processing. In the next step we perform data cleaning and filtering to remove duplicates and irrelevant data. During transformation we create metadata features and target variables (review helpfulness) for stratified samples. TF IDF method is used to get the importance of words. Finally, metadata features and TFIDF matrices are used for modeling. Next the dataset is split into three parts: train, test and validate which is 70%, 15% and 15% respectively. We have used Logistic Regression, SVM, Random Forest and XG Boost machine learning algorithms. Multiple iterations of model training will occur on training data and adjustments are made to model's hyperparameters using validation dataset which aims to find an optimal result with best model

performance. We test these models on test data and Once models are trained and tuned, they are saved as pickle files.

Figure 58

Architecture of amazon product review helpfulness



4.3. Model Comparison and Justification

Our project aimed to train machine learning models to classify and predict the helpfulness of online product reviews on Amazon across various product categories. The models considered text processing techniques and features extracted from review text, vocabulary richness, and metadata to drive the review helpfulness. As part of data transformation, we have implemented TF-IDF techniques to extract features from our cleaned and pre-processed review datasets. The most informative terms are selected based on the TF-IDF scores with this technique. This helps to reduce the dimensionality of the feature space, that can improve the computational efficiency and model performances. These computed TF-IDF feature vectors were used as input to all the machine learning algorithms outlined. For our use case, we explored these following Machine Learning models Logistic Regression, SVM, Random Forest and XGBoost.

We chose Logistic Regression primarily due to its simple implementation and interpretability when it comes to text classification tasks. It is easy to implement and is extremely efficient from a computational standpoint. It is less prone to overfitting and the model

coefficients can help to provide valuable insights into importance of different features of the review text. It is well suited for classification tasks having binary targets. As our target variable was binary, it was a logical choice for us to start our modeling process with this model to quickly identify the patterns in the data. We used Logistic Regression to establish a quick baseline before diving deeper into more complex algorithms.

Support Vector Machine (SVM) was the next model we trained and evaluated. It was chosen primarily due to its ability to establish an optimal boundary especially when dealing with high dimensional spaces. With text classification tasks, high dimensionality is a major challenge that makes modeling a complex and resource intensive task. Using hyperparameter tuning the aim was to establish an optimal set of parameters and kernels for the SVM model. After feature extraction using TF-IDF, our data dimensionality was very high, so SVM was an ideal choice.

Next, we explored ensemble learning methods due to their robust performance in capturing complex patterns and relationships in textual data. They can handle class imbalance effectively and that is usually the case with text review datasets as majority of the review are not helpful. Random Forest and XGBoost are both ensemble learning method that combine weak learners to create a strong prediction model. Random Forest combines multiple decision trees to improve the overall performance. Its ability to efficiently handle noisy features and capture complex patterns made it a suitable choice to include in our project. XGBoost (Extreme Gradient Boosting) is a gradient boosting framework that is very powerful and can effectively be applied on textual data. Regularization techniques like L1 and L2 are an integral part of XGBoost, that can help to prevent overfitting. It also improves the generalization performance which is helpful when dealing with high dimensionality data. Both Random Forest and XGBoost provide a

measure of feature importance, which is an extremely critical for identifying the most helpful words or terms relevant to our categories.

Table 19 provides the justification for choosing these models in our analysis. We have mentioned the strengths and weaknesses of each model based on factors like interpretability, scalability, training time, prediction time, handling of missing data, robustness to outliers etc, relevant to our use case.

Table 19

Model Comparison

Model	Strengths	Weakness
Logistic Regression	• Simple and relatively interpretable • Beneficial for small datasets • It provides the probabilistic output, can be helpful for scoring the review helpfulness	• Assumes the linear relationship which may not hold true for textual data • Extensive feature engineering requirement • Limited performance to complex problems
Support Vector Machine	• Can model non-linear decision boundaries using Kernel trick • Effective for Binary classification • Ability to handle imbalanced dataset • Robustness to overfitting, even with high dimensionality data	• Increase the computational complexity and training time • Less interpretable than logistic regression • Constraints on memory requirements as data size grows
Random Forest	• Ensemble Learning can lead to better predictive performance • Reduced overfitting • Resistance to noise and outliers • Parallelization and scalability	• Computational complexity • Bias towards corelated features • Sensitive to noisy features
XGBoost (Extreme Gradient Boosting)	• Includes built in regularization techniques, which help prevent overfitting • Supports Parallel Processing • Memory efficient • Provides a measure of feature importance	• Sensitive to hyperparameters • Computational complexity • Potential overfitting if not properly tuned

4.4. Model Evaluation Methods

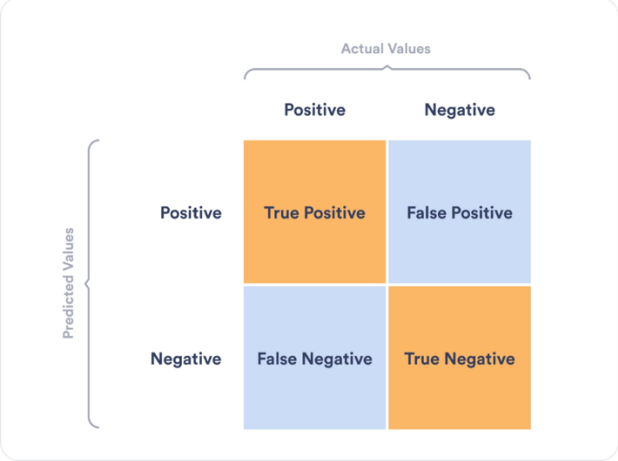
We have four classification models for predicting review helpfulness for each of the chosen product categories. To evaluate the metrics for the models, we have used several metrics such as Accuracy, Precision, Recall, and F1-score. These metrics can be calculated using the confusion matrix, which represents the actual versus the predicted values. Additionally, we're also plotting the ROC (Receiver Operating Characteristic) curve, which is a graph of sensitivity versus (1-specificity). The below subsections will explain the metrics used for predicting the helpfulness of the review as explained in research paper by Hudgins et al. (2023)

4.4.1. Confusion Matrix

A confusion matrix is a type of representation that can be used to evaluate the quality of the classification that is between the predicted class and the actual class. It is also referred to as the error matrix. The confusion matrix can represent whether the review is classified correctly or incorrectly. A typical confusion matrix is a 2x2 grid, as shown in the below Figure 60.

Figure 60

Exemplary *Confusion Matrix*



Note. Exemplary Confusion Matrix from "What Is a Confusion Matrix And How to Read It? " by Lory Seraydarian (2022), *Plat ai* (https://plat.ai/blog/confusion-matrix-in-machine-learning/)

In our project, the True Positives (TP) are the reviews which are classified correctly. The True Negatives (TN) are the reviews that are classified as not helpful and in reality, they are not helpful. False Positives (FP) are the reviews that are actually not helpful but are incorrectly classified as being helpful, whereas False Negatives (FN) are the reviews that are actually helpful, but the model classifies as not helpful. An ideal classification model will not have any FP or FN or have very few misclassification instances.

4.4.2. Accuracy

Accuracy is the overall percentage of reviews that are classified correctly. It is calculated using (22).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{22}$$

In our project, the accuracy of predicting the overall helpfulness gives us a sense of the overall correctness of a model's predictions. While it doesn't give us the complete picture of predictions and may be misleading, especially since our data is imbalanced, it is a simple metric that can help us judge the overall performance of a model and can help in comparing against other models.

4.4.3. Precision

Precision gives us the proportion of correctly predicted helpful reviews (true positives or TP) among all reviews predicted as helpful. It is calculated using (23).

$$Precision = \frac{TP}{TP + FP} \tag{23}$$

In our project, precision is a measure of predicting helpfulness correctly. It gives us an understanding of the proportion of predicted helpful reviews that are actually helpful. A model that is highly precise for helpfulness prediction minimizes the number of false positive

predictions. To simplify, a highly precise model accurately identifies helpful reviews and avoids misclassifying not helpful as being helpful.

4.4.4. Recall

Recall measures the proportion of correctly predicted positive instances (true positives) among all actual positive instances. Recall, which is also called Sensitivity, gives us the proportion of correctly predicted helpful reviews (true positives or TP) among all reviews that are actually helpful. Recall is also referred to as the True Positive Rate or TPR. It is calculated using (24).

$$Recall = \frac{TP}{TP + FN} \tag{24}$$

In our project, recall is a measure of how many reviews that are actually helpful were identified correctly by the model. It gives us an understanding of whether the model is capturing a large proportion of reviews that are actually helpful.

4.4.5. F1 Score

F1-score is calculated as the harmonic mean of precision and recall and helps in striking a balance between precision and recall, effectively addressing the trade-off between the 2 metrics. It is calculated using (25).

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{25}$$

In our project, the F1-score provides a measure of the model's performance by giving us a holistic measure of a model's ability to classify helpful reviews correctly. Since it is a harmonic mean, it gives equal weightage to precision and recall. The F1 score is useful in our project since our data is imbalanced, where the share of helpful reviews is significantly smaller than the non-

helpful reviews. The F1 score is also an objective metric that helps in comparing the performance of different classification models built for predicting helpfulness.

4.4.6. ROC curve

The Receiver Operating Characteristic (ROC) curve plots the True Positive Rate (TPR, also referred to as sensitivity) against the False Positive Rate (FPR, also calculated as 1 - specificity). There are two types of ROC curves in the context of predicting helpfulness. ROC curve with probabilities plots the decision threshold across a range of predicted helpfulness probabilities.

ROC curve with binary outcomes plots a fixed decision threshold (50%) to convert predicted helpfulness into binary predictions. In our project, we are using the ROC curve with binary outcomes for its simplicity. ROC curves can be plotted for different models for comparing the discriminative power of predicting helpfulness. Additionally, from the ROC curve, we can calculate the Area Under the Curve (AUC), which measures the overall performance of predicting helpfulness of reviews. A higher AUC value indicates better differentiation between helpful and not helpful reviews, with an AUC being one is a perfect classifier that can fully differentiate between helpful and not helpful reviews. An AUC of 0.5 means that our model is as good as randomly guessing whether a review is helpful or not. A typical ROC curve is shown in Figure 61.

Figure 61

Classic ROC Curve

4.5. Model Validation and Evaluation

Logistic Regression

Base Model. Multiple logistic regression baseline models for four different categories were trained and tested with 70,15,15 split for train, test and validation. The output includes accuracy, precision, recall, f1-score and classification outcomes labeled as not helpful (0) and helpful (1). For appliances the accuracy is 91.69%, automotive 89.95%, cellphones and accessories is 93.07%. For the category 'Tools & Home Improvement" we considered class weights to manage imbalance in data. Hence, for tools and home improvement, accuracy is 74.5%, however, the recall and AUC improved significantly. Here, all models without class weights show high precision and recall for not helpful but perform poorly for helpful, however, with class weights there is high recall but lower precision and accuracy.

Hyperparameter-tuned. There was a low performance in baseline models due to which we decided to perform hyperparameter tuning. Hyperparameter tuning is done using grid search, which performs a search over all possible combinations of the hyperparameter ranges to find the optimal combination of hyperparameters. For logistic regression models, we have tuned across

two hyperparameters, C and penalty. C is the inverse of regularization strength which means lower values indicate stronger regularization and penalty is penalization. For review helpfulness prediction, we have used C values 0.01, 0.1, 1, 10, 20, and for Penalty, we have used L1(Lasso Regression) and L2 (Ridge Regression) regularization.

A lower C value indicates that stronger regularization helped a little in managing the overfitting but not enough to improve helpful prediction metrics. The best parameter across four categories is shown in Table 20.

Table 20

Optimal parameters for logistic regression across categories

Category	Inverse Regularization C	Penalty	Best Parameter
Appliances	[0.01, 0.1, 1, 10, 20]	[l1 and l2]	[C': 0.01, 'penalty': 'l1']
Automotive	[0.01, 0.1, 1, 10, 20]	[l1 and l2]	['C': 0.1, 'penalty': 'l2']
Cellphones and Accessories	[0.01, 0.1, 1, 10, 20]	[l1 and l2]	['C': 0.1, 'penalty': 'l2']
Tools and Home Improvement	[0.001, 0.01, 0.1, 1, 10]	[l1 and l2]	['C': 0.01, 'penalty': 'l1']

Test data. The tuned models were tested on the testing dataset, which is 15% of stratified sample data. To understand the performance of classification model we are using confusion matrices. In all categories the class is imbalanced due to which there is high number of true negatives when compared to true positives. Appliances, cellphones and Automotive categories are unable to identify helpfulness which indicates the high bias towards majority class which is not helpfulness which is why there is underfitting for minority class. However, for Tools category, the AUC and Recall improved drastically similar to the base models as the model was trained using class weights. This improvement was with a trade-off with accuracy and precision. So with Logistic regression, we were not able to get a perfectly balanced model. Figure 62 shows confusion matrix for logistic regression across all categories.

Figure 62

Confusion matrix across categories for baseline and tuned models for logistic regression

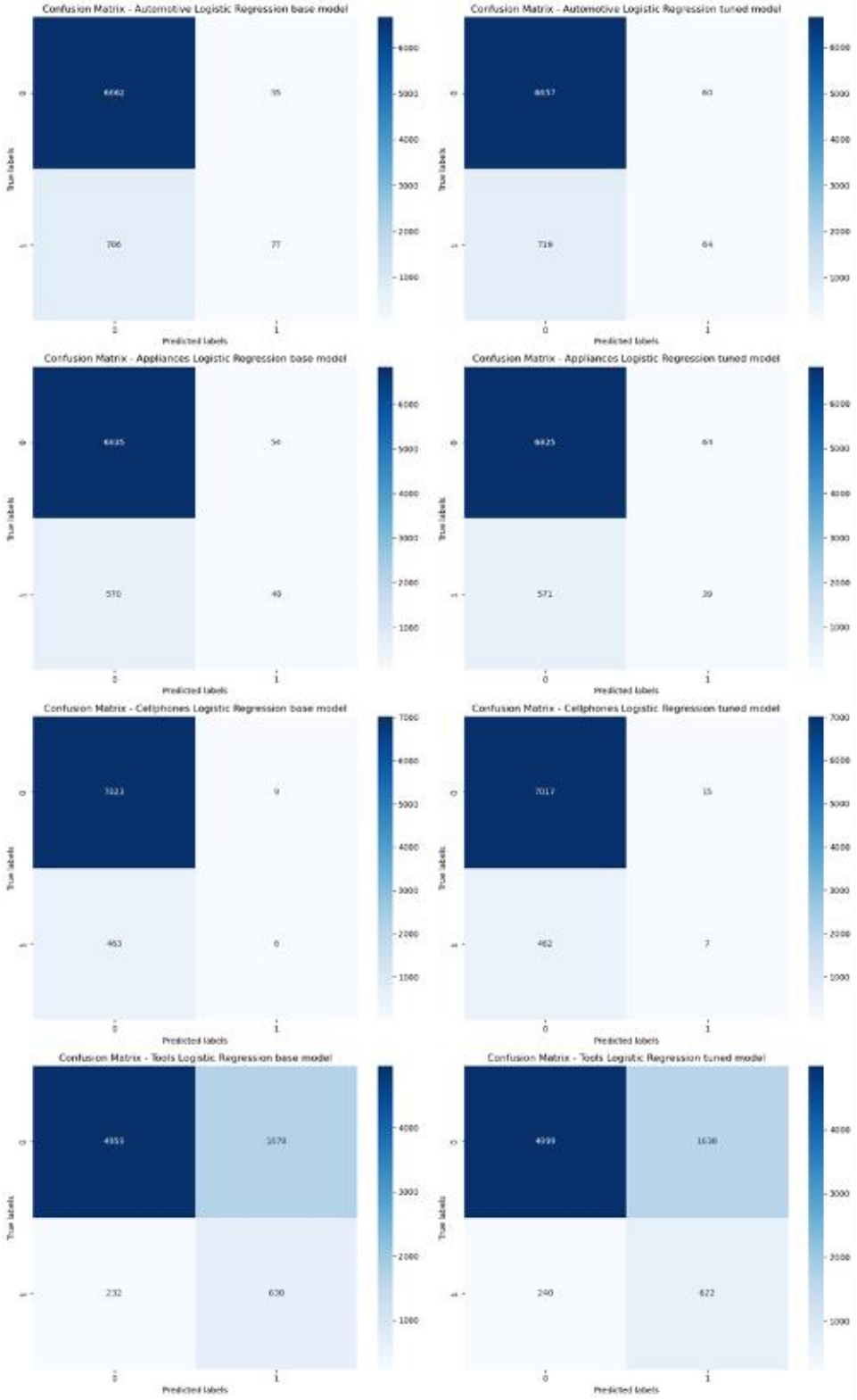
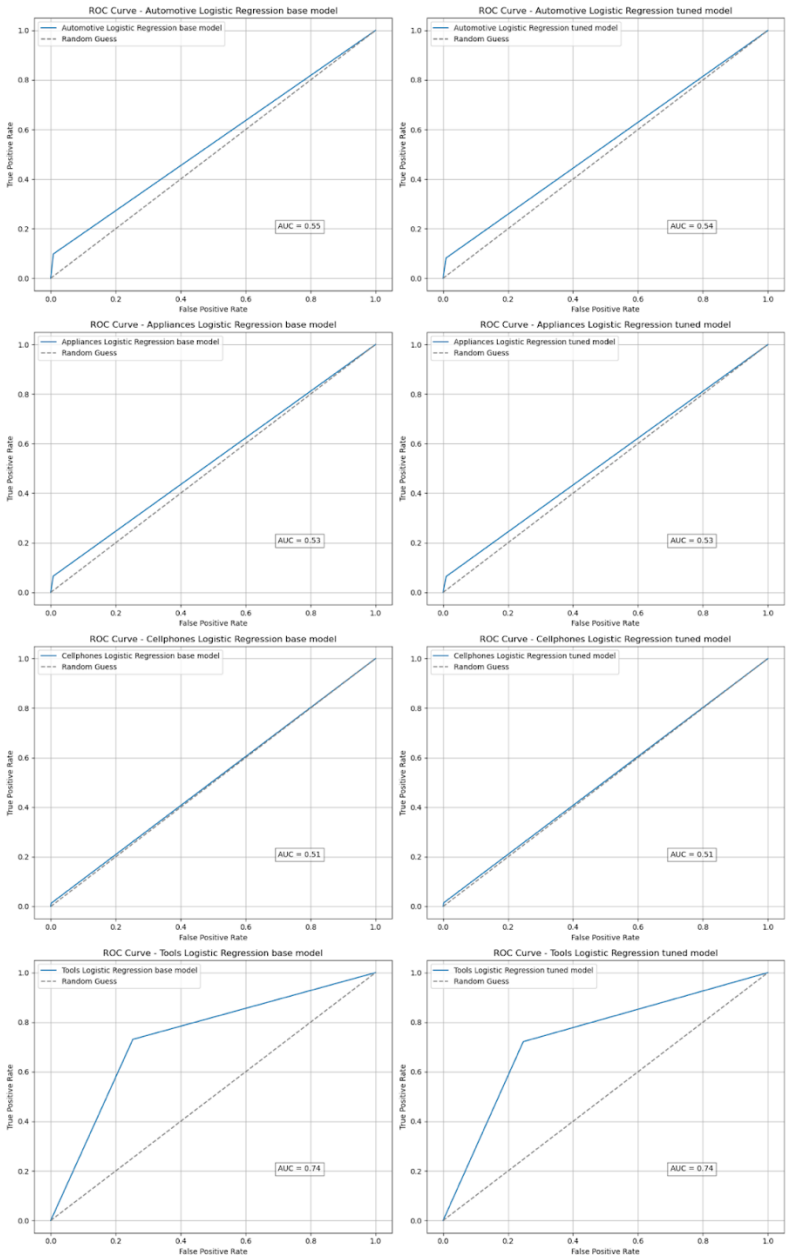


Figure 63, is the ROC curves for true positive rate against false positive rate for logistic regression model for all four categories. For appliances, cellphone and automotive categories the model performance is low where it is not able to discriminate between classes. However, for tools the model shows better performance as compared to others in terms of AUC.

Figure 63

AUC-ROC across categories for baseline and tuned models for logistic regression



A summary of all the classification metrics for logistic regression across four categories is shown in Figure 64.

Figure 64

Comparison of Evaluation Metrics across categories for baseline and tuned models

Model	Accuracy	Precision	Recall	F1-Score	AUC
Automotive Logistic Regression base model	0.898533	0.583333	0.0983397	0.168306	0.545076
Automotive Logistic Regression tuned model	0.896133	0.516129	0.0817369	0.141125	0.536402
Appliances Logistic Regression base model	0.916789	0.425532	0.0655738	0.113636	0.528868
Appliances Logistic Regression tuned model	0.915322	0.378641	0.0639344	0.109397	0.527322
Cellphones Logistic Regression base model	0.937075	0.4	0.0127932	0.0247934	0.505757
Cellphones Logistic Regression tuned model	0.936408	0.318182	0.0149254	0.0285132	0.506396
Tools Logistic Regression base model	0.745299	0.272964	0.730858	0.397476	0.739017
Tools Logistic Regression tuned model	0.749567	0.275221	0.721578	0.398463	0.73739

Support Vector Machine (SVM)

Base Model. We trained our models with default parameters to establish a baseline across all models. The baseline models were trained using SVC class for SVM classification with parameters “kernel = rbf” across three review categories (Appliances, Automotive and Cellphone & Accessories). Linear SVC class was used for SVM classification for the category ‘Tools & Home Improvement” by considering class weights to manage imbalance in data. All the models were trained on training set and Table 21 shows the evaluation metrics.

The baseline models trained with SVC class give high overall accuracies of 91%, 89% and 93% for across the three categories tested, suggesting it correctly classifies the majority class. However, the remaining metrics reveals the limitations in its ability to distinguish between positive and negative instances. With an AUC of 0.5 across the board, the performance of these models is no better than random guessing. Additionally, these models are unable to classify the positive instances, as shown by the recall score as 0.0. These models achieve a perfect precision score of 1.0, indicating no false positives for any one class, however, when considering recall of 0.0 for that class, it suggests that these models are not making any actual positive predictions. Further F-1 score being 0.0 tells us that model is not able to strike a balance between precision

and recalls, indicates the poor performance in terms of accurately identifying instances. Therefore, even though the accuracy is high, it is heavily biased on the majority class and struggles to identify any instances of the minority class.

The same pattern was originally observed in category “Tools & Home Improvement” as well with original SVC class. However, with these observations we tried to make a slight change to our approach and see if the results could vary. The dataset for Amazon Reviews is imbalanced, where one class significantly dominates (Not helpful reviews) as compared to the other (helpful reviews). It is extremely important to handle this imbalance as it can lead to bias towards the majority class which is what we observed with the previous three categories. To address this, we have employed the concept of class weights in all the ML algorithms specifically for category “Tools and Home Improvement”. Class weights allow us to assign higher weights to the minority class during model training, this allows the model to focus more on the minority class and achieve better results. Along with that we changed the class from SVC with rbf kerner to Linear SVC which uses linear kernel. This was done as SVC is extremely resource and time intensive on high dimensional data and it wasn’t yielding useful results as evident from Table 21. So, we just focused on linear SVC with class weights to avoid extensive processing times. We trained the baseline model for this category taking the default parameters “max_iters =5000, dual = False”. Dual was taken as “False” to use the primal formulation of SVM which is more efficient large number of features. “Max_iter = 5000”, is taken to consider the maximum number of iterations for the solver to converge. Setting a higher value of max_iter, allows the solver (algorithm that finds the optimal hyperplane) to run for more iterations, leading to better convergence. This model returned accuracy of 0.738 with acceptable recall of 0.722 but low Precision 0.265 and with AUC 0.731. Table 21 shows the metrics of the model on “Tools &

Home Improvement" category. This model performed better than SVC as it was able to detect actual positive instances as compared to models for previous categories. However, it came at a trade-off with accuracy and precision meaning the model was often predicting negative instances as positive. The AUC-ROC of 0.731 suggests a fair ability to distinguish the positive and negative instances. We see that using class weights helps to manage the imbalance in data and helps the model perform slightly better.

Table 21

Comparison of evaluation metrics for baseline models

Categories	Accuracy	AUC	Recall	Precision	F-1 score
Appliances	0.918656	0.5	0.00	1.0	0.0
Automotive	0.895600	0.5	0.00	1.0	0.0
Cell Phones and Accessories	0.937475	0.5	0.00	1.0	0.0
Tools and Home Improvement	0.738765	0.731792	0.722738	0.265898	0.388768

Hyperparameter-tuned. As the baseline models were not performing, we decided to use hyper parameter tuning to achieve the best parameters for modelling. A grid search is performed on the validation set for the first three categories from Table <> to find the best parameters for the model training. The parameters considered for search with the SVC class were "kernel = linear, rbf , poly", "cv = 3" as the number of folds for the cross validation and the scoring was set to "accuracy". The best parameter returned for all the categories was "kernel = rbf". We used this parameter to train the final models.

However, for category "Tools & Home Improvement" we handled the hyper parameter tuning differently. A grid search was performed to find the best parameters for the model. We defined a dictionary called svm_param_grid that specifies the parameters and their values to be

explored during Grid Search. The following parameters were considered for tuning “C (Regularisation parameter) = 10, 100, 500, 1000, 10000”, “max_iter = 5000, 10000”, “loss = squared_hinge”, “penalty = l2 (Ridge), l1 (Lasso)”, and “dual = False”. We also considered “cv = 3” as the number of folds for the cross validation. We defined the custom scoring function called roc_auc_scorer that calculates ROC_AUC (Receiver Operating Characteristic Area Under the curve) score, as we want the model to have the ability to distinguish between positive and negative classes. We used “estimator.predict_proba(X)” to predict the probability of each sample in X that belongs to the positive class. We used this custom scorer to focus more on the ROC AUC score for our model as we noticed that “Accuracy” was not a true measure of performance for our use case as is evident from Table 22. ROC AUC is a good metric to focus on when classes are imbalanced for binary classification problems. We performed the actual hyper parameter tuning process by fitting the GridSearchCV object to the validation data using sample weights. The best parameters returned were at “C= 10, penalty = l2, max_iter = 5000”.

Table 22

List of Parameters used for Hyperparameter Tuning

Category	**SVM Class Used**	**Parameters for Tuning**	**Best parameters**
Appliances	SVC	kernel = linear, rbf, poly	kernel = rbf
Automotive	SVC	kernel = linear, rbf, poly	kernel = rbf
Cell Phones and Accessories	SVC	kernel = linear, rbf, poly	kernel = rbf
Tools and Home Improvement	Linear SVC	C = 10, 100, 500, 1000, 10000 max_iter = 5000, 10000 loss = squared_hinge penalty = l2 (Ridge), l1 (Lasso) dual = False	C= 10 penalty = l2 max_iter = 5000

Test Data. We used the best parameters outlined in Table 22 to train our final model on the test dataset. The metrics from the final tuned model are available in the Table 23

.

Table 23

Comparison of evaluation metrics for tuned models

Categories	Accuracy	AUC	Recall	Precision	F-1 score
Appliances	0.918656	0.5	0.00	1.0	0.0
Automotive	0.895600	0.5	0.00	1.0	0.0
Cell Phones and Accessories	0.937475	0.5	0.00	1.0	0.0
Tools and Home Improvement	0.738899	0.730353	0.719258	0.265411	0.387742

We noticed that with SVC classifier for the categories “Appliances, Automotive and Cellphones & Accessories” the tuned models returned the exact same metrics as the baseline models. This was primarily since the tuned parameter was same as the baseline model which was “kernel=rbf”. For “Tools & Home Improvement” category we saw a minor reduction in performance as compared to the baseline model in terms of AUC (0.730 – tuned vs 0.731 - baseline) which indicates that the tuning of parameters did not make a significant impact the overall outcome. SVM being highly sensitive to hyperparameters did not perform well with either of the approaches. However, using class weights we were able to achieve slightly better results. In conclusion SVM cannot be considered an ideal algorithm for this problem. The comparison metrics between baseline and tuned models are presented in the Figure 65, the metric comparisons are available in the Figure 66 and the confusion matrix comparisons are presented in Figure 67.

Figure 65

Comparison of Evaluation Metrics across categories for baseline and tuned models

Model	Accuracy	Precision	Recall	F1-Score	AUC	TPR	FPR
Automotive SVM base model	0.8956	1	0	0	0.5	1	1
Automotive SVM tuned model	0.8956	1	0	0	0.5	1	1
Appliances SVM base model	0.918656	1	0	0	0.5	1	1
Appliances SVM tuned model	0.918656	1	0	0	0.5	1	1
Cellphones SVM base model	0.937475	1	0	0	0.5	1	1
Cellphones SVM tuned model	0.937475	1	0	0	0.5	1	1
Tools SVM base model	0.738765	0.265898	0.722738	0.388768	0.731792	0.722738	0.259153
Tools SVM tuned model	0.738899	0.265411	0.719258	0.387742	0.730353	0.719258	0.258551

Figure 66

Comparison of AUC-ROC across categories for baseline and tuned models

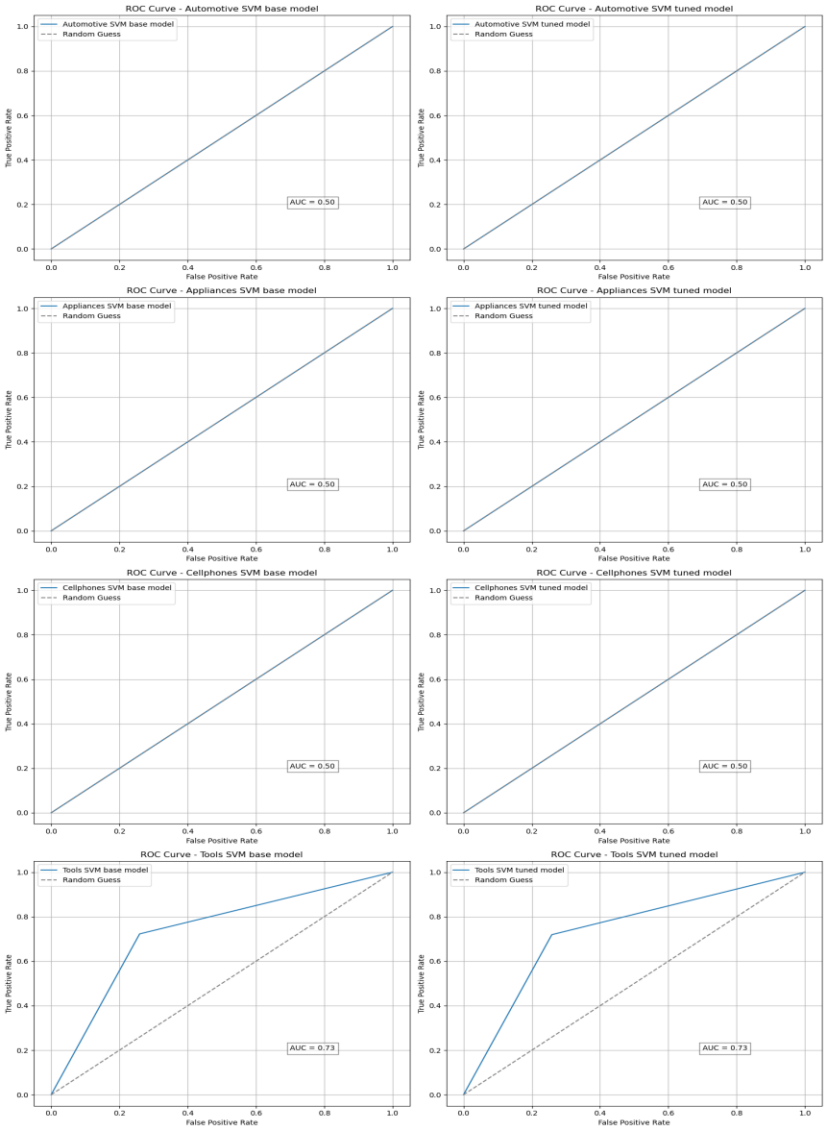
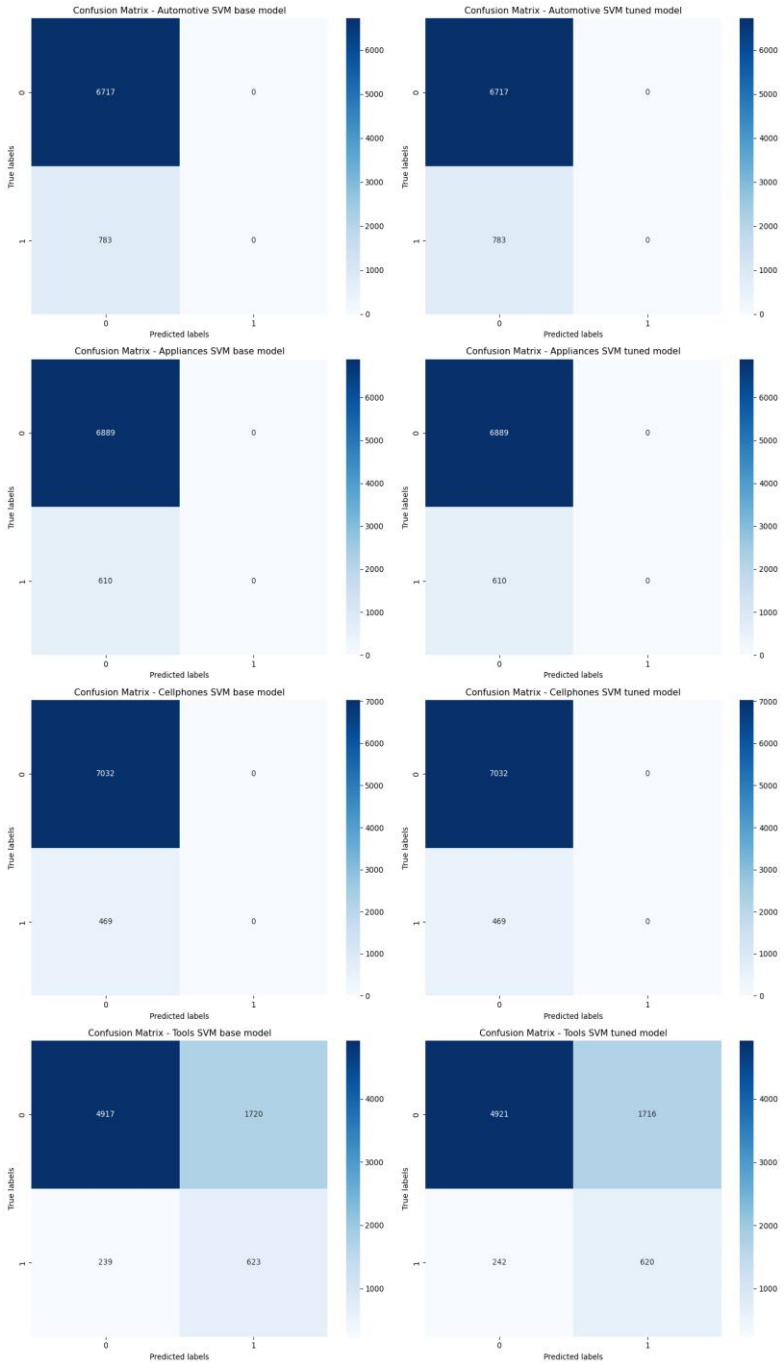


Figure 67

Comparison of confusion matrix across categories for baseline and tuned models



Random Forest

Base Model. Each category's baseline, or base, Random Forest model is trained using default parameters on the training dataset for various categories. There was a slight variation for the “Tools & Home Improvement” category as we used the concept of class weights to handle imbalance in the dataset. This helps the model focus on the minority class instead of being biased towards the majority class.

Hyperparameter-tuned. The validation dataset is used to fine-tune the models using the best possible parameters. The validation dataset was used for tuning since this method lessens overfitting and enhances the model's capacity for generalization. Additionally, since hyperparameter tuning occurs on a smaller dataset than training, which comprises 70% of the dataset, employing the validation dataset, which comprises 15% of the modeling dataset, speeds up the process.

We tested with "n_estimators" and "max_depth" as hyperparameters for the Random Forest models in all categories. In order to estimate how beneficial reviews will be, the "max depth" option sets the maximum depth for each tree in the ensemble. It is a measure of a tree's potential complexity and affects the model's ability to identify patterns in the input data. While a smaller value prevents the model from learning, a higher value or a deeper tree might capture more subtle aspects of the data but also run the risk of overfitting. As a result, determining the ideal value for the "max_depth" parameter is essential. We tried three different values: 3, 5, and 7.

The number of boosting rounds or trees to be constructed in the ensemble is indicated by the "n_estimators" parameter. Since the model would be better able to learn complicated feature interactions if there were more trees in the ensemble, a higher number would ideally improve

model performance. However, there is a trade-off, as we have limited computational resources, a higher number would also significantly lengthen the model's training period and computational requirements.

After the value ranges are established, grid search is used to tune the hyperparameters. It searches through all possible combinations of the hyperparameter ranges to identify the best possible combination. We have optimized the training time using 3-fold cross-validation because of our restricted computational resources. Finding the best possible parameter combination to get the highest accuracy is our aim when it comes to hyperparameter tweaking. The Table <> displays the ideal hyperparameter values for every category run.

Table 24

Optimal hyperparameters for Random Forest across all the categories

Category	max_depth	n_estimators
Appliances	None	50
Automotive	20	200
Cellphones and Accessories	10	50
Tools and Home Improvement	30	200

Note. All the hyperparameters are calculated with max features as 1000, along with metadata features.

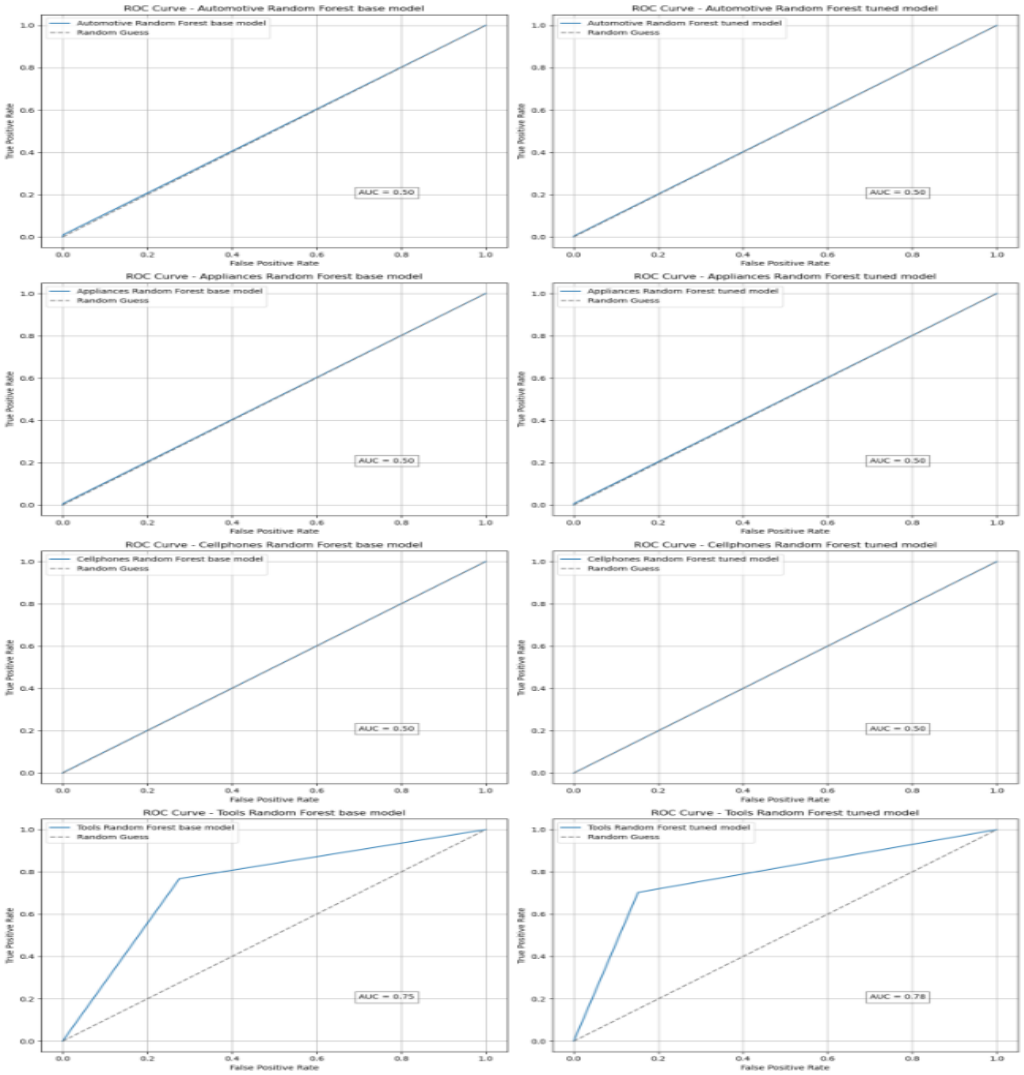
Test data. The testing dataset, comprising 15% of stratified sample data, was used to evaluate the tuned models. By testing the model with data that hasn't been seen before, we can determine how effectively it generalizes and whether it is overfitting the training set. To get baseline model predictions and tuned model predictions, respectively, we employed the baseline and tuned models on the test dataset. This made it possible for us to assess how much the

tweaked models' performance improved over the baseline. We noticed that with use of class weights with the “Tools and Home Improvement” category, the AUC scores improved as well as the recall improved indicating better performance when it comes to identifying true positives. However, this result came as a trade-off with accuracy and precision which means that the model identifies some negative results as positive. This indicates that handling class imbalance helps to tune the model, however, there is no significant improvement in the overall performance in review helpfulness classification.

A comparison of Random Forest models ROC (Receiver Operating Characteristic) curves before and after hyperparameter adjustment is shown in Figure 68. The performance of the model is depicted in each set of plots; Notably, tuning did not have any significant impact on models’ performance. Though precision and accuracy improved, however, the recall was down to 0 meaning the model is doing random guess and is not able to correctly distinguish between positive and negative instances. Tuning had no major impact on tools category as well, however, with a recall of 0.7, the mode is performing better in distinguishing between positive and negative instances. However, this is with a trade-off on a accuracy and precision.

Figure 68

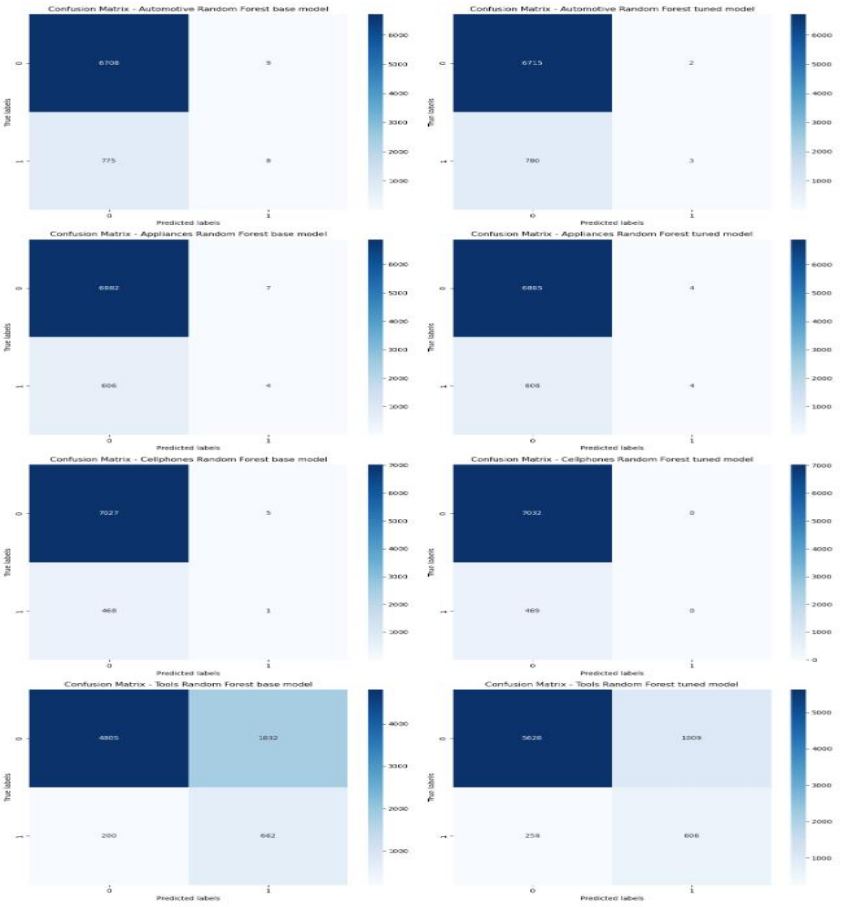
ROC curve for baseline and tuned Random Forest model for all categories



A set of confusion matrices for Random Forest models applied to different categories, both before and after hyperparameter adjustment, are shown in Figure 69. For every prediction produced by the models, the number of true positives, true negatives, false positives, and false negatives is shown in each matrix.

Figure 69

Confusion matrix for baseline and tuned Random Forest model for all categories



Comparing the base models and their customized models, the Table 25 shows classification metrics for Random Forest models across four categories. Accuracy, precision, recall, F1-score, and AUC-ROC are among the measurements. Precision is marginally increased for appliances and automobiles, but tuning has little effect on other metrics, such as the extremely low recall and F1-scores, which show poor performance in accurately predicting the positive class. Significantly, the tuned model for accessories and cellphones exhibits a zero precision, recall, and F1-score, indicating an inability to anticipate positive events upon tuning. The Tools and Home Improvement category, on the other hand, exhibits a notable improvement

after tweaking in all measures, especially in recall and F1-score, indicating a minor improvement in detecting positive instances while retaining appropriate accuracy and AUC-ROC values.

Table 25

Classification metrics for Random Forest model for all categories

Category	Accuracy	Precision	Recall	F1-score	AUC-ROC
Appliances (base model)	91.82	0.3636	0.0065	0.0128	0.5027
Appliances (tuned model)	91.86	0.5000	0.0065	0.0129	0.5029
Automotive (base model)	89.54	0.4705	0.0102	0.0200	0.5044
Automotive (tuned model)	89.57	0.6000	0.0038	0.0076	0.5017
Cellphones and Accessories (base model)	93.69	0.1666	0.0021	0.0042	0.5007
Cellphones and Accessories (tuned model)	93.74	0.0000	0.0000	0.0000	0.5000
Tools and Home Improvement (base model)	71.88	0.2559	0.7394	0.3802	0.7277
Tools and Home Improvement (tuned model)	79.49	0.3053	0.5942	0.4034	0.7078

XGBoost

Base Model. The baseline or base XGBoost model for each category is trained with default parameters on the respective category's training dataset. The output includes accuracy, precision, recall, f1-score, and classification outcomes labeled as not helpful and helpful reviews. For appliances, the accuracy is 91.33%, automotive is 89.13%, and cellphones and accessories are 93.60%. For the category 'Tools & Home Improvement" we considered class weights to manage imbalance in data. This helps the model focus on the minority class instead of being biased towards the majority class. Hence, for tools and home improvement, accuracy is 76.62%,

however, AUC and Recall were improved. All the models were trained on the training set, tested on the test set and Table 27 shows the evaluation metrics.

Hyperparameter-tuned. The models are tuned with optimal parameters using the validation dataset. Tuning was done on the validation dataset since this approach helps in reducing the overfitting of the model and improves the model's ability to generalize. Also, using the validation dataset, which is 15% of the modeling dataset improves the speed of hyperparameter tuning since it happens on a smaller dataset compared to training, which is 70%.

The hyperparameters that we experimented with for the XGBoost models across all categories were "learning_rate", "max_depth," and "n_estimators". Since XGBoost is an ensemble of multiple trees, the "learning_rate" parameter controls the contribution of each tree in the ensemble to the final helpfulness prediction. A lower value would mean more boosting rounds are required to train the model, making the learning process slow. A higher value increases the learning speed but may result in overfitting. Hence, choosing an optimal value for the "learning_rate" parameter is crucial, and we attempted three values, i.e., 0.01, 0.1, and 0.2.

The "max_depth" parameter controls the maximum depth of each tree in the ensemble to predict the helpfulness of reviews. It is a measure of how complex individual trees can be and influences the model's capacity to learn patterns from the input data. A higher value or a deeper tree can capture more nuanced features of the data but can also overfit, while a lower value inhibits the model's ability to learn. Hence, choosing an optimal value for the "max_depth" parameter is crucial, and we attempted three values, i.e., 3, 5, and 7.

The "n_estimators" parameter pertains to the number of boosting rounds or trees to be built in the ensemble. A higher value would ideally improve model performance since more trees in the ensemble would increase the ability of the model to learn complex feature interactions. But

the trade-off is that a higher value would also considerably increase the training time and computation requirement of the model which presents a challenge since we are working with limited computation resources.

Once the ranges of values are defined, hyperparameter tuning is done using grid search (we used GridSearchCV from sklearn library), which performs a search over all possible combinations of the hyperparameter ranges to find the optimal combination of hyperparameters. Given our limited computational resources, we have used 3-fold cross validation for optimizing the training time. Our purpose in hyperparameter tuning is to determine the optimal combination of parameters that will result in the highest possible accuracy. For the appliances category, the learning rate is 0.1 with max depth as 5 and n_estimator as 50. In the automotive category, the learning rate is 0.01 with max depth as 3 and n_estimator as 200. In the cellphones and accessories, the learning rate is 0.2 with max depth as 3 and n_estimator as 50. For the category ‘Tools & Home Improvement,” we considered class weights to manage imbalance in data. Hence for tools and home improvement, the learning rate is 0.01 with max depth as 3 and n_estimator as 200.The optimal hyperparameter values for each of the category runs are shown in the Table 26

Table 26

Optimal hyperparameters for XGBoost across all the categories

Category	learning_rate	max_depth	n_estimators
Appliances	0.1	5	50
Automotive	0.01	3	200
Cellphones and Accessories	0.2	3	50
Tools and Home Improvement	0.01	3	200

Note: All the hyperparameters are calculated for XGboost model

Test data. The tuned models were tested on the testing dataset, which is 15% of stratified sample data. Testing on unseen data will help in understanding how well the model generalizes and we can also understand if the model is overfitting the training data. We used the baseline and tuned models on the test dataset to get baseline model predictions and tuned model predictions, respectively. This allowed us to compare the improvement in performance between the baseline and the tuned models. We noticed that with the use of class weights with the "Tools and Home Improvement" category, the AUC scores improved as well as the recall improved indicating better performance when it comes to identifying true positives. However, this result came as a trade-off with accuracy and precision which means that the model identifies some negative results as positive. This indicates that handling class imbalance helps to tune the model, however, there is no significant improvement in the overall performance in review helpfulness classification.

Figure 69

ROC curve for baseline and tuned XGBoost model for all categories

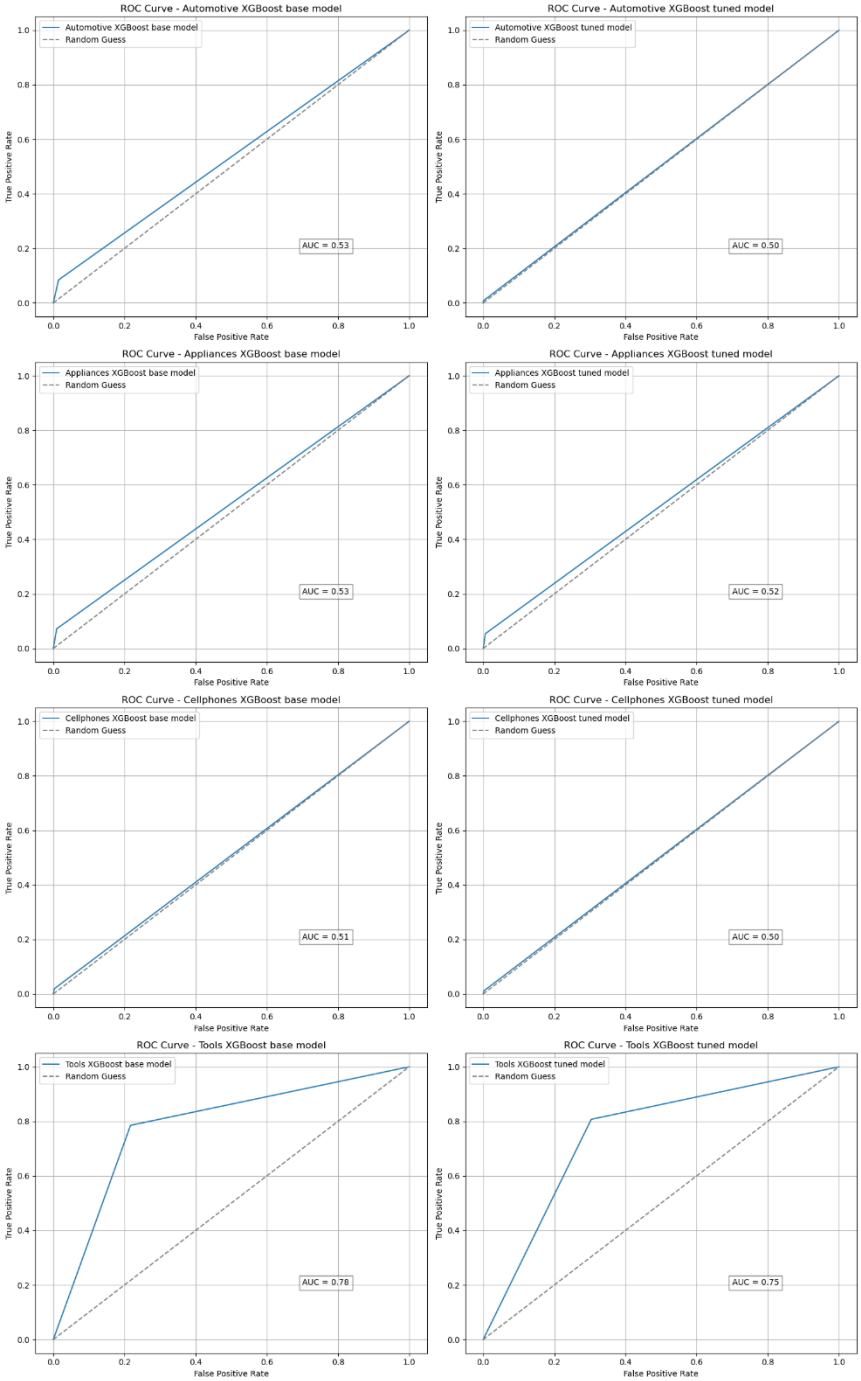


Figure 70

Confusion matrix for baseline and tuned XGBoost model for all categories

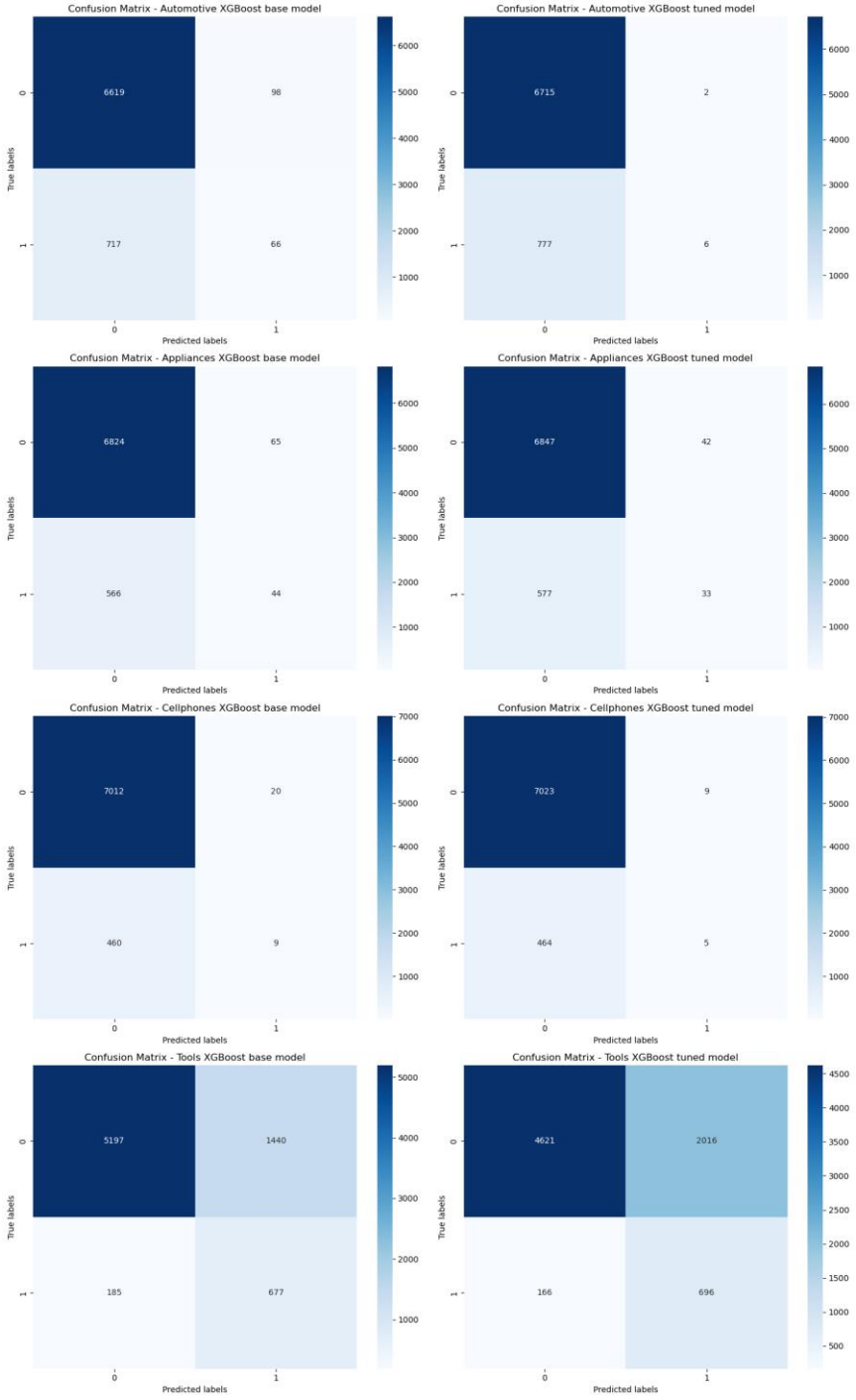


Table 27

Classification metrics for XGBoost for all categories

Category	Accuracy	Precision	Recall	F1-Score	ROC AUC
Appliances (XGBoost Baseline)	91.33	0.3484	0.0754	0.1239	0.5314
Appliances (XGBoost Tuned)	91.47	0.3209	0.0426	0.0752	0.5173
Automotive (XGBoost Baseline)	89.13	0.4024	0.0842	0.1393	0.5348
Automotive (XGBoost Tuned)	89.61	0.7500	0.0076	0.0151	0.5036
Cellphones and Accessories (XGBoost Baseline)	93.60	0.3103	0.0191	0.0361	0.5081
Cellphones and Accessories (XGBoost Tuned)	93.69	0.3571	0.0106	0.0207	0.5046
Tools and Home Improvement (XGBoost Baseline)	76.62	0.2856	0.6685	0.4002	0.7238
Tools and Home Improvement (XGBoost Tuned)	69.66	0.2514	0.8091	0.3836	0.7454

Note. These values are specific to our project for XGBoost model.

4.5.2. Comparison of Model Evaluation Results

The model performance for each category has been summarized in the Table 28. We have chosen the best-performing models across categories based on accuracy. Even though the accuracies is low for tools and home improvement category, but the AUC and Recall improved using class weights.

Table 28

Summarized accuracy comparison of model evaluation

Algorithm	Appliances	Automotive	Cellphones	Tools
Logistic Regression	91.68% (base)	89.85% (base)	93.70% (base)	74.95% (tuned)
Random Forest	91.86% (tuned)	89.57% (tuned)	93.74% (tuned)	83.13% (tuned)
SVM	91.86% (tuned)	89.56% (base)	93.75% (base)	73.88% (tuned)
XGBoost	91.75% (tuned)	89.61% (tuned)	93.69% (tuned)	78.33% (base)

For Appliances, the tuned version of Random Forest and SVM showed the highest accuracy. For Automotive, Logistic Regression base model showed the highest accuracy. For Cellphones, SVM showed the highest accuracy. For Tools, even though the accuracy is low for tools and home improvement category, we saw improvement with Recall and AUC scores. This helped the model perform better than without class weights. We can observe that different categories had different best models indicating that the model performance is a function of the underlying data and there is no clear winner across the board. It is to be noted that accuracy is not the only measure of performance. Some models show higher recall, higher F1-score, or higher Area Under Curve.

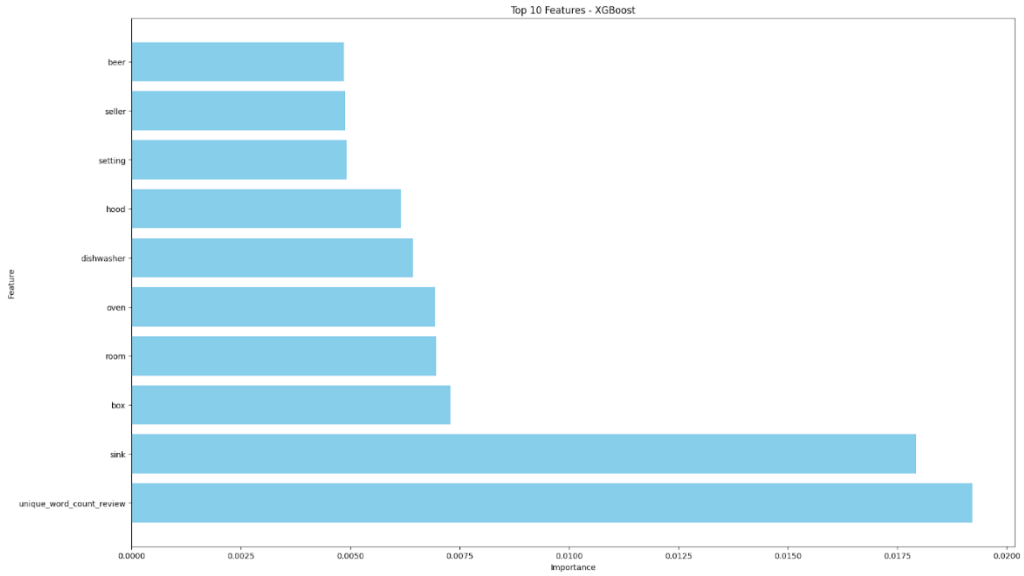
4.5.3. Model Explainability

It is crucial for us to interpret the model correctly and explain why it is working the way it is. Model explainability is important to provide transparency about what features the model considers important in predicting helpfulness, what is the direction of influence of the feature (positive and negative) in predicting the outcome and do the features contribute to the model in a way that makes intuitive sense. There are multiple ways of quantifying and representing model explainability and these vary by algorithms. Feature importance in tree-based methods like

Random Forest and XGBoost are directly available from the model object using the "feature_importances_" attribute which calculates the "gain" of each feature. The importance score of a feature is calculated based on how much each feature contributes to reducing the loss function during the construction of decision trees in the ensemble to predict helpfulness. This contribution is measured by the improvement in the model's performance (reduction in the loss) attributed to each split that involves the feature. These values can be sorted and plotted to visualize the relative importance of each feature. Figure 71 shows the relative feature importance from the XGBoost model for Appliances category.

Figure 71

Feature Importance for XGBoost

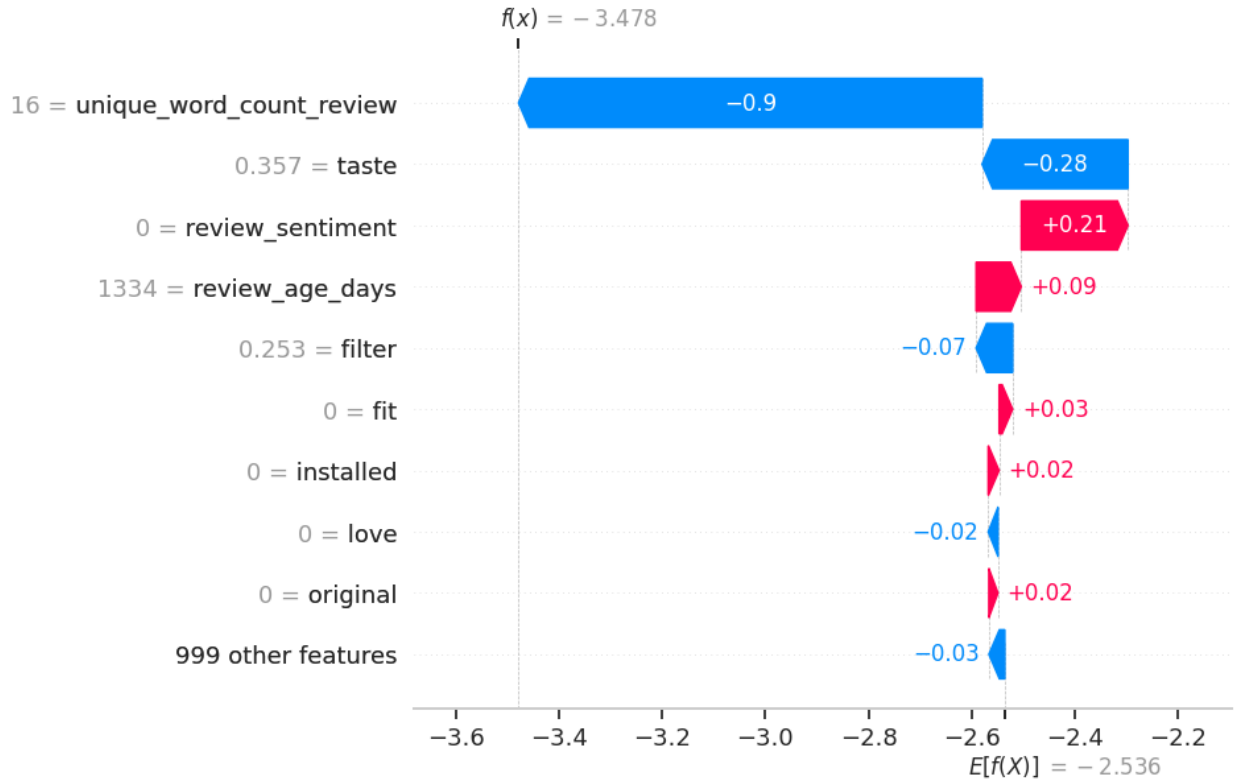


We can observe that "unique_word_count_review" is the most important feature in predicting helpfulness. However, the shortcoming of this method of feature importance is that we know the magnitude of the influence but not the direction i.e. we can't determine whether unique word count in the review is positively or negatively affecting helpfulness, and this is something that we need to analyze separately.

Another method to determine feature explainability is through Shapley values. SHAP (SHapley Additive exPlanations) values are a way for explaining the outcome of machine learning models. They provide a way to understand the contribution of each feature to the model's prediction for helpfulness. SHAP values are based on cooperative game theory and the concept of Shapley values originate from economics. They aim to fairly distribute the "credit" or "contribution" of each feature across all possible combinations of features. Positive SHAP values indicate that the feature contributes to increasing the prediction of helpfulness, while negative values indicate the opposite. The magnitude of the values represents the impact of the feature on the prediction, with larger magnitudes indicating greater influence. Figure 72 shows a waterfall chart visualizing the SHAP values for the XGBoost tuned model for the Appliances category.

Figure 72

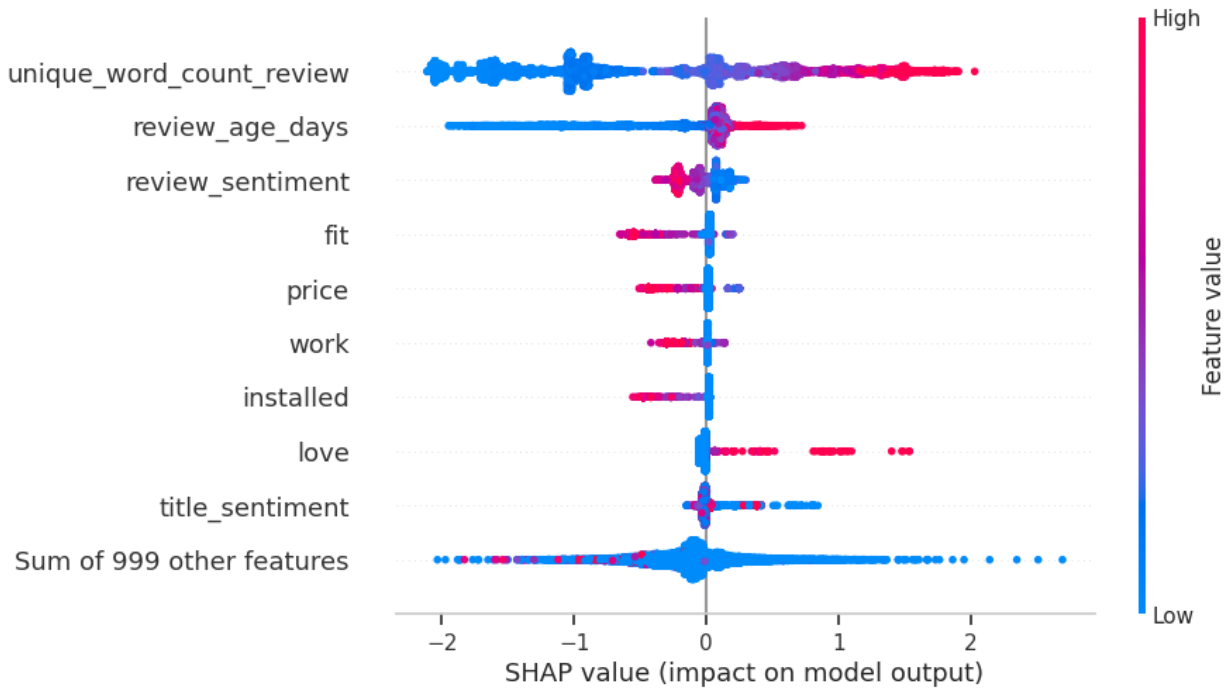
Waterfall chart visualizing the SHAP values



Features pushing the helpfulness prediction higher are shown in red and those pushing the prediction lower are in blue. We can now observe that unique word count negatively impacts the prediction i.e., more unique words in a review is associated with non-helpful reviews. Similarly, we can notice that review sentiment is positively influencing the model i.e. higher the sentiment score of the review, better is the helpfulness. The Figure 73 shows a beeswam chart, an alternative way of visualizing the SHAP values for the XGBoost tuned model for the Appliances category with a distribution of the data points based on helpfulness.

Figure 73

Beeswam chart for SHAP values



4.5.4. Conclusion

Models for three categories “Appliances, Automotive and Cellphones & Accessories” have high accuracy but low precision and recall, it can be attributed to the fact that our data is imbalanced and has high dimensionality. Our data has the majority class as “Not Helpful” with

“Helpful” being the minority class. It's representative of real-world data since helpful reviews are limited and exclusive. Since we are using metadata and TF-IDF features, we have a high number of dimensions. We can use dimensionality reduction techniques like PCA, but the trade-off is that our model explainability will be limited. Additional features that we have not considered are likely features around human behavior that can’t be quantified. We attempted iterations of the model with limited TF-IDF features (100), a larger TF-IDF feature matrix (2000), and another iteration with only metadata features however, the results did not change much across the iterations, indicating that our current set of features are not adequate to accurately predict helpfulness.

Another approach we explored was to use sample weights to offset the imbalance in the “Tools & Home Improvement” dataset. We were able to make the models focus more on the minority class that is “helpful” reviews. We were able to get decent recall values and an ROC score of more than 0.75, however, that was achieved at the cost of accuracy and precision. With this, we could interpret that Accuracy is not the only metric that should be considered for our use case, precision and recall are extremely important to determine whether the models are able to capture true positives. We noticed that with sample weights, we had to trade-off accuracy to allow models to capture more true positives while introducing some noise. Given these observations, we conclude that even though we were able to arrive at the helpfulness with a certain degree of accuracy, these models do not perform optimally. The takeaway is that it is difficult to clearly separate helpful vs. non-helpful reviews using multiple classification techniques that we attempted as the performance for our models was sub-optimal and did not strike a balance.

Our hypothesis is that the helpfulness of reviews is subjective and a function of human behavior. Further exploration of different NLP techniques and may be required to achieve highly optimal models for our use case.

4.5.5. Limitations and Future Scope

With the traditional machine learning models, we have not been able to strike a balance with our trained models. Although we were able to achieve fairly performing models, however, they cannot be considered good from a real-world perspective. We primarily explored features generated with TF-IDF along with certain metadata features like sentiment, review length and age. However, there are other techniques like Part-of-Speech (POS), Named Entity Recognition (NER), Word Embeddings etc., that were not considered in scope for this analysis. We had constraints with computational resources, and the high dimensionality of our data made it difficult to train the models on conventional machines. The next step for this research calls for exploration of advanced feature extraction techniques along with more powerful algorithms like transformer-based models (e.g. BERT, GPT) or deep learning techniques like CNN (Convolutional Neural Networks), Bi-LTSM, XLNet.

References

Abhilasha Singh Rathor, Amit Agarwal, Preeti Dimri, *Comparative Study of Machine Learning Approaches for Amazon Reviews, Procedia Computer Science*, Volume 132, 2018, Pages 1552-1561, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.05.119

Chatterjee, I.; Zhou, M., Abusorrah, A., Sedraoui, K., Alabdulwahab (2021). *A Statistics-Based Outlier Detection and Correction Method for Amazon Customer Reviews. Entropy 2021, 23, 1645*. https://doi.org/10.3390/e23121645

Hudgins, Triston, Joseph, Shijo, Yip, Dougla, and Besanson, Gaston (2023). Identifying Features and Predicting Consumer Helpfulness of Product Reviews. SMU Data Science Review: Vol. 7: No. 1, Article 11. https://scholar.smu.edu/datasciencereview/vol7/iss1/11

Kapil Kaushik, Rajhans Mishra, Nripendra P. Rana, Yogesh K. Dwivedi (2018). *Exploring reviews and review sequences on e-commerce platform: A study of helpful reviews on Amazon.in. Journal of Retailing and Consumer Services, Volume 45, 2018, Pages 21-32, ISSN 0969-6989.* https://doi.org/10.1016/j.jretconser.2018.08.002

Kim, S. M., Pantel, P., Chklovski, T., & Pennacchiotti, M. (2006, July). Automatically assessing review helpfulness. *In Proceedings of the 2006 Conference on empirical methods in natural language processing (pp. 423-430).* https://aclanthology.org/W06-1650.pdf

Li, H. (2017). *Amazon Product Reviews Helpfulness Prediction.* https://doi.org/10.17615/jxj5-jt77

Mahdikhani, M. (2023). Exploring commonly used terms from online reviews in the fashion field to predict review helpfulness. International Journal of Information Management Data Insights, 3(1), 100172. https://doi.org/10.1016/j.jjimei.2023.100172

Miao Fan et al. (2019). Product-Aware Helpfulness Prediction of Online Reviews. WWW '19: The World Wide Web Conference, May 2019, Pages 2715–2721. https://doi.org/10.1145/3308558.3313523

MSI Malik, & Ayyaz Hussain (2018). *An analysis of review content and reviewer variables that contribute to review helpfulness*. *Information Processing & Management, Volume 54, Issue 1, 2018, Pages 88-104, ISSN 0306-4573.* https://doi.org/10.1016/j.ipm.2017.09.004

Nguyen, L. T. K., Chung, H.-H., Tuliao, K. V., & Lin, T. M. Y. (2020). Using XGBoost and Skip-Gram Model to Predict Online Review Popularity. *Sage Open, 10(4)*. https://doi.org/10.1177/2158244020983316

Park, Y. J. (2018). Predicting the helpfulness of online customer reviews across different product types. Sustainability, 10(6), 1735. https://www.mdpi.com/2071-1050/10/6/1735

Pumrapee Poomka, Nittaya Kerdprasop, & Kittisak Kerdprasop (2021). Machine Learning Versus Deep Learning Performances on the Sentiment Analysis of Product Reviews. *International Journal of Machine Learning and Computing, Vol. 11, No. 2, March 2021.* https://www.ijmlc.org/vol11/1021-DY026.pdf

Robert Ireland, & Ang Liu (2018). *Application of data analytics for product design: Sentiment analysis of online product reviews*. *CIRP Journal of Manufacturing Science and Technology, Volume 23, 2018, Pages 128-144, ISSN 1755-5817.* https://doi.org/10.1016/j.cirpj.2018.06.003

Sara Ashour Aljuhani and Norah Saleh Alghamdi. *A Comparison of Sentiment Analysis Methods on Amazon Reviews of Mobile Phones*. International Journal of Advanced Computer Science and Applications (IJACSA), 10(6), 2019. http://dx.doi.org/10.14569/IJACSA.2019.0100678

Sobia Wassana, Xi Chenb , Tian Shenc, Muhammad Waqard , NZ Jhanjhie (2021). Amazon Product Sentiment Analysis using Machine Learning Techniques. *Revista Argentina de Clínica Psicológica 2021, Vol. XXX, N°1, 695-703.* https://doi.org/10.24205/03276716.2020.2065

Subhashini, L.D.C.S., Li, Y., Zhang, J. *et al.* (2021). Mining and classifying customer reviews: a survey. *Artif Intell Rev* 54, 6343–6389. https://doi.org/10.1007/s10462-021-09955-5

Thomas L. Ngo-Ye, & Atish P. Sinha (2012). Analyzing Online Review Helpfulness Using a Regressional ReliefF-Enhanced Text Mining Method. *ACM Transactions on Management Information Systems, Volume 3, Issue 2, Article No.: 10pp 1–20.* https://dl.acm.org/doi/10.1145/2229156.2229158

Tomas Pranckevicius, & Virginijus Marcinkevicius (2017). Exploring Machine Learning Classifiers for Sentiment Analysis of Online Product Reviews: A Comparative Study. *Baltic J. Modern Computing, Vol. 5 (2017), No. 2, 221-232.* http://dx.doi.org/10.22364/bjmc.2017.5.2.05

Yen-Liang Chen 1, Chia-Ling Chang 2, ORCI and An-Qiao Sung (2021). Predicting eWOM's Influence on Purchase Intention Based on Helpfulness, Credibility, Information Quality, and Professionalism. Sustainability 2021, 13(13), 7486. https://doi.org/10.3390/su13137486

Appendix

GitHub is used for handling source code developed as part of this project implementation and to maintain the code. GitHub link: https://github.com/VeenaBeknal/Predicting-Amazon-Product-Review-Helpfulness

Python Jupyter notebooks for each category along with the respective category data files and model pickle files are uploaded in the Google drive here: https://drive.google.com/drive/u/1/folders/1SiHD1419_eWjSmahj36uzBwKtVmAnF_i .

The original dataset is available here: https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/