Sentiment Analysis on Amazon Fine Food Reviews

[1]Goutham Thota, [2]Jaya Surya Thota, [3]Tejaswi Chintapalli

Abstract

This research provides a comprehensive analysis of the significance of customer reviews, which has grown substantially with the increasing number of online shoppers. The study employs machine learning algorithms to examine sentiment in Amazon Fine Food Reviews, utilizing a vast dataset of over 568,454 reviews to investigate the various aspects of sentiment analysis. The study assesses the effectiveness of different machine learning algorithms in categorizing the polarity of the reviews, including Bernoulli Naive Bayes, Multinomial Naive Bayes, Random Forest, Logistic Regression, and Decision Tree. The performance of each algorithm is evaluated, and Logistic Regression is found to demonstrate superior performance with an accuracy rate of 89%. The results of this investigation can offer valuable insights to entities in comprehending customer sentiment and enhancing their products and services by leveraging the collected information. It is a crucial tool for organizations to understand customer feedback and improve their offerings, leading to better customer satisfaction and loyalty.

Introduction

Sentiment analysis is a method for drawing conclusions from text data. It includes examining and classifying literature into positive, negative, or neutral sentiment categories. People are expressing their ideas more openly than ever before due to the development of social media and e-commerce platforms. These suggestions and comments might offer insightful information about the customer's experience and aid companies in enhancing their goods and services.

Amazon is one such online store with a large number of client reviews. Popular online retailer Amazon sells a variety of goods in a number of different categories. Customers can provide comments and reviews on the goods they buy, which others who are interested can use to make sensible choices. Manually assessing these reviews, however, can be a lengthy and difficult task. That's where sentiment analysis comes in handy.

Sentiment analysis's major goal is to categorize customer feedback into positive, negative, or neutral. Positive reviews generally indicate that customers are satisfied with the product, while negative reviews indicate that customers are dissatisfied with the product. Neutral reviews do not express any strong sentiment and may indicate a lack of opinion or ambivalence towards the product.

Sentiment analysis on Amazon Fine Food Reviews can be used to identify the overall sentiment of customers towards food products on Amazon. Over 568,454 reviews are included in the Amazon Fine Food Reviews dataset, which makes it a valuable tool for sentiment analysis. The dataset can be examined in order to provide significant findings using machine learning algorithms including Bernoulli Naive Bayes, Random Forest, Multinomial Naive Bayes, Decision Tree, and Logistic Regression. Businesses can gain important insights into client happiness and feedback from sentiment analysis. Businesses can use it to analyze client needs and developments, as well as locate opportunities for product or service improvement. Businesses can better understand their customers and enhance the overall customer experience by researching customer reviews.

Additionally, sentiment analysis can be used in many other fields. To study public opinion on various political parties or programs, for example, it might be utilized in politics. In order to examine patient input and enhance patient care, it can also be utilized in healthcare. To examine consumer reviews and brand reputation via social media, sentiment analysis is also a useful tool.

In conclusion, sentiment analysis is an efficient tool that may give businesses insightful information about client feedback and aid in the enhancement of their goods and services. Sentiment analysis can be used to evaluate massive datasets, such as Amazon Fine Food Reviews, and deliver precise conclusions with the aid of machine learning techniques. By analyzing customer preferences and trends and improving customer experiences, businesses that use sentiment analysis gain a competitive advantage.

Data

The dataset is Amazon Fine Food Reviews from Kaggle. The Dataset has 568,454 records with 10 attributes total and Score is our target variable. Reviews from October 1999 to October 2012 are included in the dataset.

Attributes	Description
Id(Integer)	A unique number assigned to each record in the dataset.
ProductId(string)	A unique identification number for each product.
UserId(string)	A unique Identification number for each user.
ProfileName(string)	A user's profile name on Amazon
HelpfulnessNumerator(integer)	The number of users who found the reviews helpful.
HelpfulnessDenominator(integer)	The number of users who indicated whether they found the review helpful or not.
Score(integer)	The rating given by the user on a scale of 1 to 5.
Time(integer)	The time when the review was written.
Summary(string)	A short summary of the review.
Text(string)	The complete text of the review.

Table: 1 Dataset Description

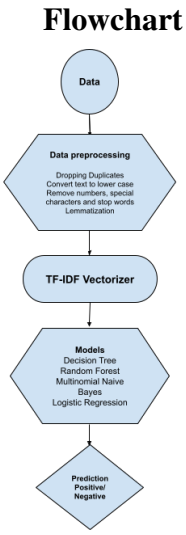

Figure: 1 Flowchart

Data Preprocessing

In contemporary society's data-centric landscape, the text corpus has become increasingly consequential due to the colossal amount of unstructured data that is generated daily. To glean cogent insights from this repository, Natural Language Processing (NLP) methods are deployed, which enable diverse applications such as sentiment analysis, topic modeling, and text classification.

However, before undertaking any text data analysis, it is crucial to perform a data cleansing process to eliminate all irrelevant elements, including special characters, numerical values, and stopwords. The code segment under consideration offers a robust approach to cleansing text data, rendering it more viable for NLP applications.

The first step of the code entails creating a novel data frame, "new_data," by selecting solely the "Text" and "Score" columns from an antecedent data frame, "data." The "Score" column can be invaluable for several NLP applications, such as sentiment analysis, where the text's sentiment is evaluated based on the provided score. The ensuing step necessitates eliminating any duplicate rows based on the "Text" column, exclusively retaining the primary instance of each unique text. This step guarantees that the resulting dataset only comprises exclusive text elements, mitigating the possibility of redundant analysis and conserving computational resources.

The "clean_text" function is subsequently defined to execute a sequence of text-cleansing measures. Primarily, the function alters the text to lowercase, rendering it more amenable to comparison and analysis. Next, numerical characters within the text are discarded, as they are often tangential to NLP applications, using regular expression substitution with the pattern '\d+'. Likewise, special characters are expunged through regular expression substitution with the pattern '[^{\w\s}]'. These characters, such as punctuation and symbols, are often extraneous and can impede analysis.

Following this, the function eradicates common stopwords from the text corpus. These stopwords, such as "a," "the," and "in," are frequently employed but provide minimal contributions to the text's import. The 'stopwords' package from the Natural Language Toolkit (NLTK) is enlisted for this task, along with list comprehension. The outcome of this step is text data that exclusively comprises the most germane terms, driving the meaning of the text.

The final measure of the function entails lemmatizing the remaining words. This process involves reducing words to their basic form, such as converting "running" to "run." This measure is crucial in downsizing the overall dimensionality of the text corpus, facilitating ease of analysis and insight extraction.

Subsequently, a new column 'Sentiment' is generated in the 'new_data' data frame using a lambda function, which classifies each score in the 'Score' column as either positive or negative based on a threshold value of 3. Thereafter, the 'Score' column is purged from the data frame employing the 'drop()' method.

Finally, a seaborn count plot is generated in Fig1.2 to visualize the distribution of sentiment categories in the 'new_data' data frame. Each bar in the plot represents the count of either a positive or negative sentiment category. The count of each category is labeled using the 'ax.text()' method for clarity. The resulting plot provides cogent insights into the overall sentiment distribution of the given dataset.

In conclusion, the code furnishes a highly efficacious and expedient means of cleansing text data, augmenting its relevance for diverse NLP applications. By weeding out extraneous elements and lemmatizing the remaining terms, the resultant text data exclusively comprises the most substantive information, engendering ease of analysis and insight derivation. This technique can be highly advantageous for commercial entities and organizations that grapple with voluminous quantities of unstructured text data, enabling them to extract meaningful insights and gain a competitive edge.

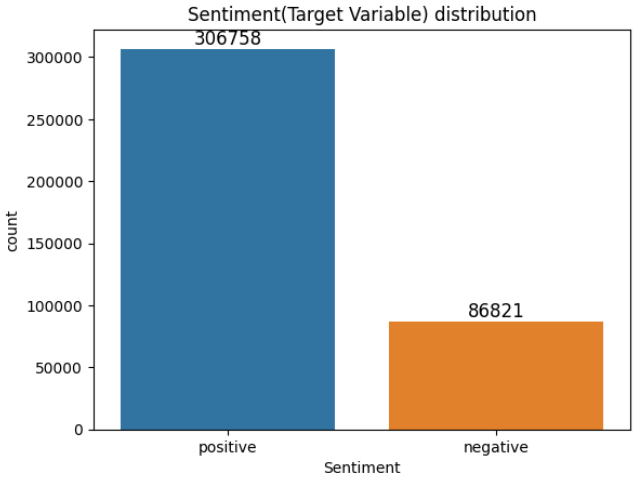

Figure 2: Distribution of Target Variable (Sentiment)

Methods

We have implemented the Pre-trained Model, Random Forest, Decision Trees, Logistic Regression, Bernoulli Naive Bayes, and Multinomial Naive Bayes.

Pre-Trained Model

Sentiment analysis extracts subjective information from text data like opinions, attitudes, and emotions. The SentimentIntensityAnalyzer from the NLTK library calculates a compound sentiment score for a given text input, which ranges from -1 to 1. The code applies this method to a preprocessed text dataset, adding the sentiment score as a new column to analyze the sentiment distribution.

The 'get_sentiment_category' function converts the numerical score into categorical labels for easier interpretation. The accuracy of the sentiment analysis model is 78% and is evaluated using the sci-kit-learn library's accuracy_score function.

Overall, the code provides a practical solution for sentiment analysis with insights applicable across domains like finance, marketing, and public opinion analysis.

Logistic Regression

A linear model called logistic regression calculates the likelihood of the positive sentiment class given the input features. The linear combination of input features is converted by the logistic function to a probability value between 0 and 1. The target variable and the input features are assumed to have a linear relationship by the logic regression method. Overfitting can be avoided by using regularization techniques like L1 or L2 regularization. When the input features are continuous or categorical, binary sentiment analysis activities can apply logistic regression.

Random Forests

Each decision tree in a Random Forest ensemble is trained using a different random subset of the input features. By combining the output of numerous decision trees, Random Forests lessen overfitting and enhance generalization performance. The input features for each tree in the ensemble are chosen at random, and each tree's forecast is then averaged to provide the final prediction. Random Forests can be used in binary sentiment analysis to categorize text data using bag-of-words or word embedding features.

Decision Trees

A decision tree is a tree-like architecture in which choices are determined depending on input features at each node. Recursively dividing the data into subsets depending on the most discriminating characteristic results in the construction of the tree. The feature with the greatest information gain at each node is selected as the splitting criterion. The tree's leaf nodes stand in for the final judgment, which can be either a positive or negative feeling. Overfitting is a possibility with decision trees, which can handle both categorical and continuous variables. To lessen overfitting, regularization procedures like pruning might be applied.

Bernoulli Naive Bayes

A probabilistic model called Bernoulli Naive Bayes makes the assumption that each feature's presence or absence is conditionally independent of the target variable. When the input features are binary (0 or 1), it is frequently utilized for binary classification tasks. With binary bag-of-words characteristics, text data can be classified using Bernoulli Naive Bayes in binary sentiment analysis.

Multinomial Naive Bayes

A probabilistic model called Multinomial Naive Bayes assumes a multinomial distribution over the counts of each feature in light of the target variable. The counts of each word in the text are the input features for text classification tasks, where it is frequently employed. When using word embedding or bag-of-words features as the input features for binary sentiment analysis tasks, Multinomial Naive Bayes can be used.

Data Splitting

This code segment involves a process of partitioning the input data into training and testing subsets, which is performed through the application of the 'train_test_split' method belonging to the 'scikit-learn' library. The input data is represented by two variables, namely 'X' and 'y', where 'X' is a set of preprocessed textual data, and 'y' is a set of associated sentiment labels.

The 'train_test_split' function is invoked with the following parameters: 'X' and 'y', which are the input data and their corresponding labels; 'test_size', which is the ratio of the testing data to the whole data and is set to 30% in this implementation; 'stratify', which is assigned 'y' to ensure that the class distribution of the input data is preserved in both the training and testing subsets; and 'random_state', which is set to a fixed value of 42 to ensure the reproducibility of the results.

After the partitioning of the input data, the TF-IDF vectorizer is utilized to transform the textual data into numerical

feature vectors, which can be utilized by machine learning algorithms. Specifically, the 'TfidfVectorizer' is instantiated with 'binary=True', which means that the resulting feature vectors are binary, i.e., the presence of each term in the document is represented by either 1 or 0.

Subsequently, the 'fit_transform' method is utilized on the training data to learn the vocabulary of the corpus and convert the input data into a sparse matrix of TF-IDF features. The 'transform' method is then applied to the testing data to transform it into a matrix of TF-IDF features, utilizing the previously learned vocabulary.

The final result of this implementation includes two variables, 'X_train_tr' and 'X_test_tr', which represent the transformed feature matrices for the training and testing subsets, respectively.

In summary, this code fragment implements the partitioning of input data into training and testing sets, followed by the application of a binary TF-IDF vectorizer for feature extraction, which is utilized for sentiment analysis.

Results

Model	Train Accuracy	Test Accuracy	Precision	Recall	F1-score
Logistic Regression	0.901559	0.887952	0.856004	0.801687	0.824142
Bernoulli Naive Bayes	0.839585	0.824661	0.748387	0.704447	0.721413
Multinomial Naïve Bayes	0.803710	0.793519	0.872346	0.533152	0.504269
Decision Tree	0.808486	0.802251	0.731538	0.595955	0.609903
Random Forest	0.779412	0.779410	0.389705	0.500000	0.438016

Table 2: Classification Report Table

- **Logistic Regression**: The logistic regression model achieved a training accuracy of 0.901 and a test accuracy of 0.888. The precision score was 0.856, indicating that 85.6% of the positive sentiment predictions were correct. The recall score was 0.802, indicating that 80.2% of the actual positive sentiment instances were correctly identified. The f1-score was 0.824, which is the harmonic mean of precision and recall.
- **Bernoulii Naive Bayes**: The Bernoulli Naive Bayes model achieved a training accuracy of 0.840 and a test accuracy of 0.825. The precision score was 0.748, indicating that 74.8% of the positive sentiment predictions were correct. The recall score was 0.704, indicating that 70.4% of the actual positive sentiment instances were correctly identified. The f1-score was 0.721.
- **Multinomial Naive Bayes**: The Multinomial Naive Bayes model achieved a training accuracy of 0.804 and a test accuracy of 0.794. The precision score was 0.872, indicating that 87.2% of the positive sentiment predictions were correct. The recall score was 0.533, indicating that 53.3% of the actual positive sentiment instances were correctly identified. The f1-score was 0.504.
- **Decision Tree**: The Decision Tree model achieved a training accuracy of 0.808 and a test accuracy of 0.802. The precision score was 0.732, indicating that 73.2% of the positive sentiment predictions were correct. The recall score was 0.596, indicating that 59.6% of the actual positive sentiment instances were correctly identified. The f1-score was 0.610.
- **Random Forest**: The Random Forest model achieved a training accuracy of 0.779 and a test accuracy of 0.779. The precision score was 0.390, indicating that 39.0% of the positive sentiment predictions were correct. The recall score was 0.500, indicating that 50.0% of the actual positive sentiment instances were correctly identified. The f1-score was 0.438.

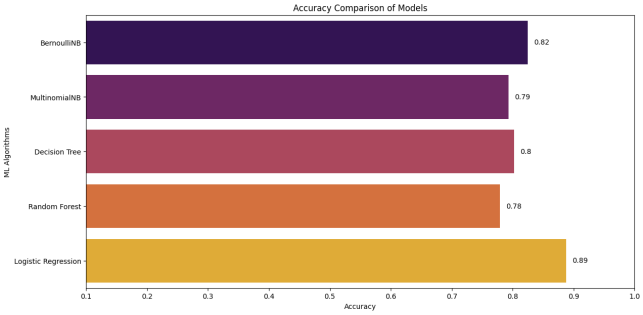

Figure 3: Accuracy Comparison

From the results, we can see that Logistic Regression had the best overall performance with the highest test accuracy, precision, and recall scores. Bernoulli Naive Bayes and Decision Tree performed similarly with moderate accuracy scores and precision/recall scores. Multinomial Naive Bayes had high precision but low recall, while Random Forest had the lowest overall performance. It's important to note that the choice of algorithm may depend on the specific requirements of the task and the nature of the data.

Conclusion

It may be inferred from the outcomes of the machine learning models examined for binary sentiment analysis that logistic regression is the best algorithm for the job. It successfully classified positive and negative sentiment in the provided dataset with the greatest test accuracy, precision, and recall scores, demonstrating this capability.

With moderate precision/recall scores and accuracy scores, the Bernoulli Naive Bayes and Decision Tree models also fared reasonably well. With lower precision, recall, and accuracy ratings, the Multinomial Naive Bayes and Random Forest models performed worse overall.

Future research could investigate more sophisticated feature engineering strategies to enhance the performance of the models. To increase the models' prediction ability even more, ensembling strategies like bagging or boosting could be used. Investigating how well deep learning models like convolutional neural networks (CNN) and recurrent neural networks (RNN) execute sentiment analysis tasks might also be intriguing. In order to assess the models' capacity for generalization, they could also be evaluated on larger and more varied datasets.

References

Bajaj, Aryan. “Can Python Understand Human Feelings Through Words? - a Brief Intro to NLP and VADER Sentiment Analysis.” Analytics Vidhya, 17 June 2021, www.analyticsvidhya.com/blog/2021/06/vader-for-sentiment-analysis.

“Sentiment Analysis Guide.” MonkeyLearn, monkeylearn.com/sentiment-analysis. https://monkeylearn.com/sentiment-analysis/

“Medium.” Medium, towardsdatascience.com/sentiment-analysis-on-trump-twitter-data-using-python-nltk-b95cd8a1a2e2. https://towardsdatascience.com/sentiment-analysis-on-trump-twitter-data-using-python-nltk-b95cd8a1a2e2