**HUMBER INSTITUTE OF TECHNOLOGY**

**AND ADVANCED LEARNING**

**(HUMBER COLLEGE)**

**Logo, company name

Description automatically generated**

**ASSIGNMENT 3 – CASE STUDY ANALYSIS**

**Team: 9**

|  |  |  |
| --- | --- | --- |
| **Last name** | **First name** | **Student number** |
| **Ganem** | **Farid** | **N01510374** |
| **Garg** | **Deepanshi** | **N01510660** |
| **Gupta** | **Sidharth** | **N01467753** |
| **Rocha** | **Milena** | **N01510465** |
| **Saxena** | **Lakshya** | **N01510799** |
| **Sharma** | **Kapil** | **N01501686** |

**Submitted to: Professor Dr. Haytham Qushtom**

**Course: Business Intelligence – BIA-5401-0LA**

**Submission Date: August 3rd, 2023**

Table of Contents

[**1.** **Introduction** 3](#_Toc141983311)

[**2.** **Data explanation** 4](#_Toc141983312)

[**3.** **Data preparation** 7](#_Toc141983313)

[**4.** **Algorithms** 11](#_Toc141983314)

[**5.** **Results** 16](#_Toc141983315)

[**6.** **Conclusion** 18](#_Toc141983316)

[**7.** **References** 18](#_Toc141983317)

# **Introduction**

In the ever-evolving landscape of the retail industry, understanding customer sentiments and opinions is paramount to success. As a response to this need, this report outlines a self-directed assignment aimed at assisting a chosen retail entity in harnessing the power of text mining to extract valuable insights from product reviews. The primary objective of this assignment is to design and implement a robust full-stack text-mining strategy that can effectively filter existing reviews into distinct positive and negative categories, thereby facilitating the classification of new reviews into these predefined segments.

The chosen approach involves working with a diverse dataset sourced from Datafiniti's Product Database, encompassing a collection of over 34,000 consumer reviews for a range of Amazon products, including popular items such as the Kindle, Fire TV Stick, and more. This dataset provides a rich pool of information, including essential product details, ratings, review text, and additional attributes, making it an ideal candidate for the development and evaluation of the proposed text-mining strategy.

The assignment unfolds across several key stages, each contributing to the creation of an effective text-mining pipeline:

**Dataset Selection and Collection**: A dataset consisting of at least 10 distinct products, each with a minimum of 20 reviews, is meticulously curated. This initial step ensures that the analysis is robust and representative of various product categories, allowing for a comprehensive evaluation of the text-mining strategy's performance.

**Data Preprocessing**: Prior to analysis, the collected dataset undergoes a rigorous data preprocessing phase. This stage involves tasks such as text cleaning, removal of irrelevant information, and handling missing values. By ensuring the dataset's quality and consistency, the subsequent analysis can generate more accurate and reliable results.

**Feature Extraction - Bags-of-Words**: The text-mining strategy employs the concept of bags-of-words for feature extraction. This approach involves converting the raw text data into numerical representations, enabling the utilization of machine learning algorithms. The chosen feature extraction technique is a crucial determinant of the model's effectiveness in capturing the inherent characteristics of product reviews.

**Machine Learning Algorithm Selection**: In this assignment, the focus shifts to selecting suitable machine learning algorithms for sentiment classification. The choice of algorithms is guided by their compatibility with the dataset's characteristics and the desired level of accuracy. Rigorous experimentation and evaluation are conducted to identify the algorithms that yield the most optimal results.

This report lays the foundation for a comprehensive text-mining strategy designed to empower retailers with the ability to effectively analyze and categorize product reviews. By employing a multi-stage approach encompassing dataset selection, data preprocessing, feature extraction, and machine learning algorithm selection, the assignment aims to provide retailers with a powerful tool for extracting valuable insights from customer sentiments. The subsequent sections delve into each stage of the process, offering detailed explanations, methodologies, and findings, all of which contribute to the overarching goal of enhancing customer experience and informed decision-making within the retail landscape.

# **Data explanation**

The dataset comprises a substantial collection of more than 34,000 consumer reviews for a range of Amazon products, including popular items like the Kindle and Fire TV Stick. Sourced from Datafiniti's Product Database, the dataset offers a comprehensive glimpse into customer experiences. Each entry is enriched with essential attributes, such as basic product details, user ratings, and detailed review text. In total, the dataset is structured across 21 columns, providing a wealth of information that serves as a valuable resource for analysis and insight generation.

* 1. **Exploratory Data Analysis:**

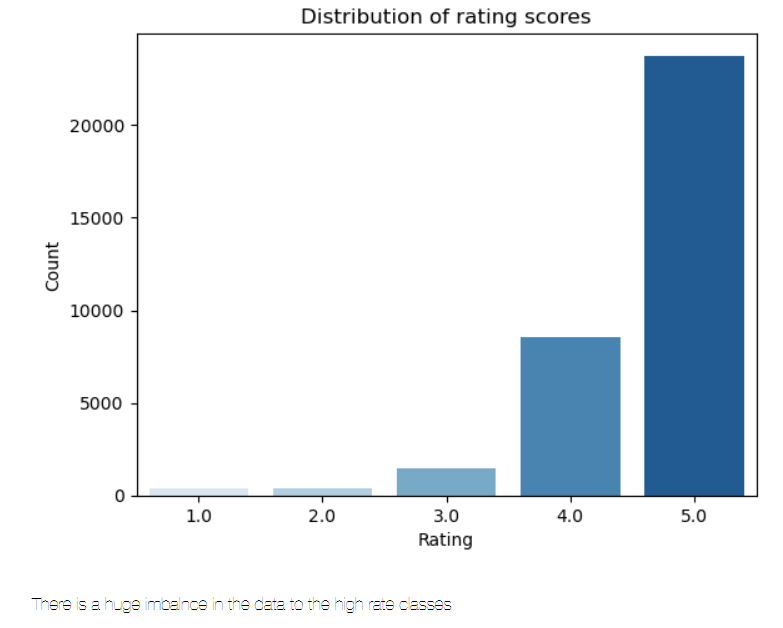
We conduct an exploratory data analysis (EDA) on the dataset. Our aim is to uncover insights and patterns within the data before diving into more advanced analyses or modelling. We start by selecting the pertinent columns, namely 'reviewsText' (textual reviews) and 'reviewsRating' (rating scores), and then provide an initial glimpse of the dataset by displaying its first few entries.

Moving forward, we delve into the descriptive statistics of the 'reviewsRating' column, which furnishes us with valuable summary measures such as mean, median, and standard deviation. To visually grasp the distribution of rating scores, we construct a count plot utilizing the Seaborn library. By observing the frequency of each rating score, we gain an understanding of how users have rated the product over time.

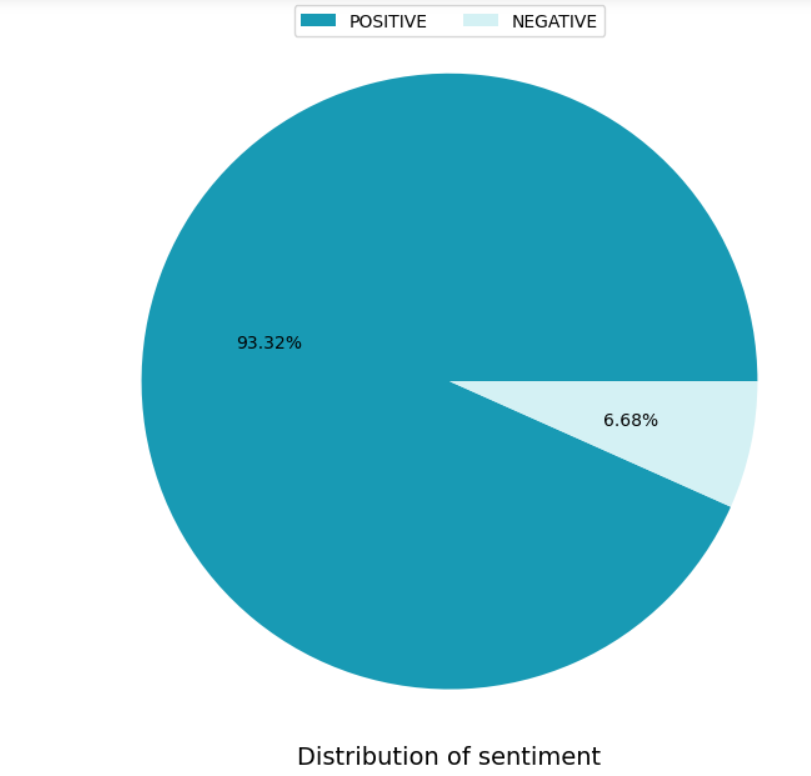
As sentiment analysis is crucial in understanding user opinions, we assign sentiment scores based on the rating scores. Ratings 1, 2, and 3 are categorized as sentiment score 0 (NEGATIVE), while ratings 4 and 5 are assigned sentiment score 1 (POSITIVE). These sentiment scores are then associated with corresponding sentiment labels. A pie chart is utilized to provide a clear visualization of the sentiment distribution, illustrating the proportion of positive and negative sentiments within the dataset.

The subsequent steps of our analysis focus on text data preprocessing. We introduce functions to clean the text, remove stopwords (commonly used words with little semantic significance), apply stemming to reduce words to their base form, and finally, perform lemmatization to transform words into their root form considering their part of speech. These processes ensure that the text data is refined and ready for subsequent analysis or modeling.

By following this systematic approach to exploratory data analysis, we have gained insights into the dataset's structure and content. The combination of statistical summaries, sentiment analysis, and text preprocessing equips us with a comprehensive understanding of the data's characteristics. This EDA process serves as a foundation for more advanced analyses, modeling, or natural language processing tasks that may be applied to uncover deeper insights or achieve specific objectives related to the product reviews.



*Figure 1: Distribution of rating scores*



*Figure 2: Distribution of sentiment*

# **Data preparation**

Data preparation is completed in multiple steps:

**Step 1** - **Importing necessary libraries:** The code starts by importing various libraries such as numpy, pandas, matplotlib, seaborn, nltk, spacy, and others, which are required for data processing, visualization, and natural language processing tasks.

**Step 2 - Data Cleaning:** The code uses NLTK (Natural Language Toolkit) to perform text data cleaning tasks such as tokenization, stopword removal, stemming, and lemmatization.

**Step 3 - Tokenization:** The word\_tokenize function from NLTK is used to break down the text into individual words or tokens.

**Step 4 - Stopword Removal:** Stopwords are common words that do not carry significant meaning in the context of natural language processing tasks. The code uses the NLTK's stopwords corpus to remove these stopwords from the text.

**Step 5 - Lemmatization:** Lemmatization is a more sophisticated process of reducing words to their base or dictionary form (lemma). The code uses the WordNetLemmatizer from NLTK to perform lemmatization on the text.

**Step 6 - Text Vectorization:** The code uses the TfidfVectorizer from scikit-learn to convert the text data into numerical vectors. TF-IDF (Term Frequency-Inverse Document Frequency) is used to represent the importance of each word in the text relative to the entire corpus.

**Step 7 - Data Splitting:** The data is split into training and testing sets using the train\_test\_split function from scikit-learn. This step is essential for training and evaluating machine learning models.

**Step 8 - Model Training:** The code utilizes the MultinomialNB (Naive Bayes) and XGBClassifier (XGBoost) classifiers to build classification models based on the preprocessed data.

**Step 9 -Model Evaluation:** The code uses various evaluation metrics such as classification\_report, confusion\_matrix, and roc\_auc\_score to assess the performance of the trained models.

**Step 10 - Ignoring Warnings:** The warnings.filterwarnings('ignore') statement is used to suppress any warning messages that might arise during the data preparation and model training process.

The purpose of cleaning the data set in this way is to sentiment analysis on a dataset.

The original dataset that we extracted had 20 columns but not all the columns were of use for us. Therefore, we created a subset dataset named ‘updated\_df’ and it has 7 columns which will help us recognizing the good and bad reviews posted by customers for multiple products.

Once this was done, we checked how many null values are present in our data, so that we don’t have any inconsistencies once we start processing the data. For this reason, we dropped all the rows that had null values. As, the sum of these rows was very small, and it won’t make any significant change in our dataset. We also checked if we had any duplicate rows, but our all the rows were unique.

After this, we checked how many unique products and reviews we have and found that we have 37 unique items and 34592 unique reviews. The next step is to clean the text of the reviews posted by the customers as

* 1. **Cleaning text**

For cleaning text as part of the sentiment analysis, we performed the following steps:

1. We are defining a function named ‘clean\_text’. This function performs the following transformations on the text to standardize it:

* Makes the text lowercase;
* Removes whitespaces;
* Removes HTML tags;
* Replaces digit with spaces;
* Replace punctuations with spaces;
* Remove extra spaces and tabs.

1. The next step for cleaning the text involves removing stop words. Stop words include the following list of words – ‘a’, ‘an’, ‘the’, ‘this’, ‘that’, ‘is’, ‘it’, ‘to’, and ‘and’. For this purpose, we are creating another function named ‘remove\_stopwords’ which will remove the stop words along with tokenizing the sentence, which is made possible using the ‘word\_tokenize’ method from the NLTK.tokenize library in python.
2. Next, we are defining the function ‘get\_wordnet\_pos’ which takes only a single argument representing the POS tag obtained from NLTK’s POS tagging. It then maps this POS tag to the corresponding WordNet POS tag using a series of if-elif statements. The mapping is as follows:

* If the POS tag starts with 'J', it is considered an adjective, and the function returns wordnet.ADJ.
* If the POS tag starts with 'V', it is considered a verb, and the function returns wordnet.VERB.
* If the POS tag starts with 'N', it is considered a noun, and the function returns wordnet.NOUN.
* If the POS tag starts with 'R', it is considered an adverb, and the function returns wordnet.ADV.
* For any other POS tag, the function defaults to considering it as a noun (wordnet.NOUN).

The purpose of this helper function is to assist in the lemmatization process by providing the lemmatizer with the correct POS tag, which allows the lemmatizer to accurately determine the base form of words based on their grammatical role in the text. By using the appropriate POS tag, the lemmatizer can disambiguate between different word meanings and provide more accurate lemmatization results, ultimately enhancing the quality of the sentiment analysis by capturing the underlying sentiment expressions in the text more effectively.

1. After getting the POS tags, we are lemmatizing the words to get the base grammatical word of the different forms of words. This is performed by defining the function ‘lemmatize’.

These defined functions then help us get the base dictionary words out of the reviews, that can be used as the keywords used for the sentiment analysis. We are then checking the functions to identify the keywords out of the random sentences which gives us a list of keywords that can then be used to bifurcate the words as positive or negative to perform the analysis.

* 1. **Feature selection**

Feature selection is an important step in sentiment analysis, as it can improve the performance of the sentiment classifier and reduce computational complexity. Here are some key reasons why feature selection is commonly used in sentiment analysis:

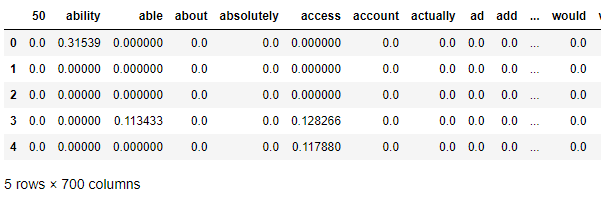
* Noise Reduction: Not all words or n-grams in the text data contribute equally to sentiment classification. Some words may be noise or irrelevant for sentiment analysis. Feature selection helps to eliminate or down-weight these irrelevant features, leading to a cleaner representation of the text data.
* Dimensionality Reduction: Text data in sentiment analysis can be high-dimensional due to the large number of unique words or n-grams in the corpus. By selecting a subset of relevant features, feature selection techniques reduce the dimensionality of the data, making the analysis more manageable and efficient.
* Overfitting Prevention: Including too many features can lead to overfitting, where the sentiment classifier becomes too specialized in the training data and performs poorly on unseen data. Feature selection helps to avoid overfitting by selecting the most informative and discriminative features that generalize well to new data.
* Improved Model Interpretability: A smaller set of selected features makes the sentiment analysis model more interpretable. It becomes easier to understand which words or phrases are contributing the most to the sentiment classification decision.
  + 1. **Steps Involved**

We chose the “TfidfVectorizer” library in python to do the feature selection for the sentiment analysis model we are making. The TfidfVectorizer is a feature extraction technique commonly used in natural language processing (NLP) and text mining tasks. It stands for "Term Frequency-Inverse Document Frequency" vectorizer. It converts a collection of raw text documents into a numerical matrix representation, which can be used as input for machine learning algorithms.

By using in the TfidfVectorizer(max\_features=700) we set the maximum feature limit to 700 i.e. the maximum number of unique words to consider as features. Post this we fit the instance of the TfidfVectorizer on our Amazon reviews data, which analyzes the text data and builds the vocabulary based on the words and their frequencies in the corpus.

We then create a numerical matrix representing the words and their TF-IDF values in the document. TF-IDF score quantifies the relative importance of a term in a specific document compared to its importance in the entire corpus.

We can use the table thus formed as the input for the models we are making.



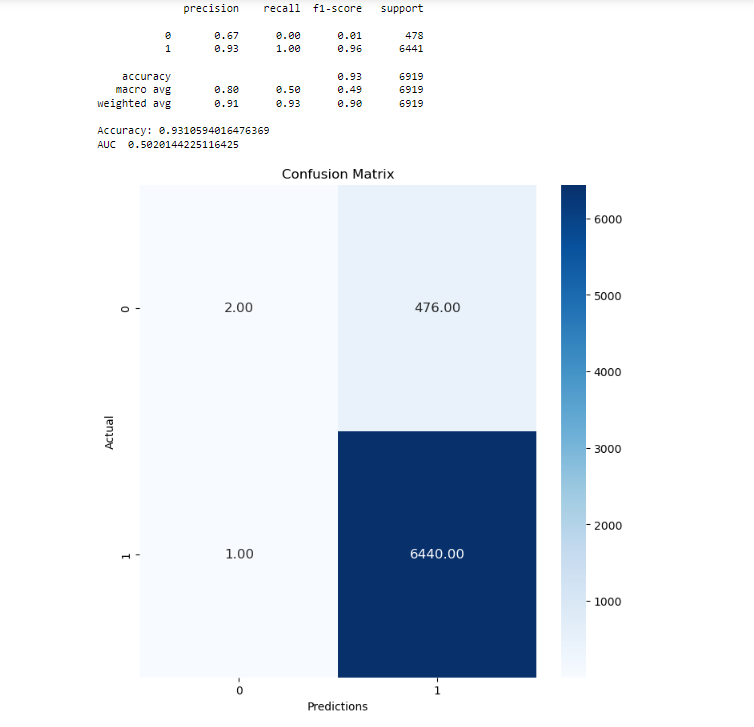
*Figure 3: Shows the first 5 rows and some of the column*

# **Algorithms**

* 1. **Multinominal Naive Bayes:**

Multinomial Naive Bayes is a commonly used algorithm for sentiment analysis on text data. It works well for sentiment analysis tasks because it can effectively handle the discrete nature of text features (word counts or TF-IDF values) and can quickly process large volumes of data.

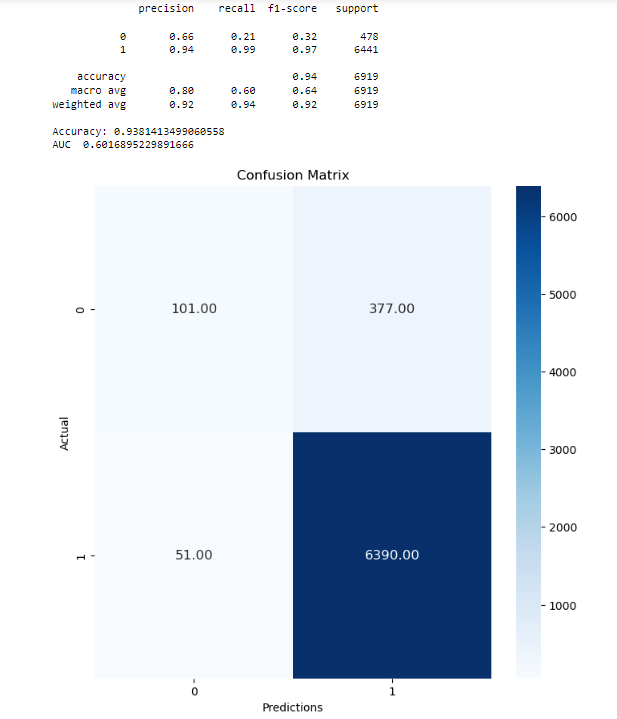
The simplicity and efficiency of Multinomial Naive Bayes make it a good choice for sentiment analysis tasks, especially when dealing with large-scale text data. However, it's important to note that Multinomial Naive Bayes assumes feature independence (the "naive" assumption), which may not hold true for all text data. Despite this simplifying assumption, Naive Bayes often performs surprisingly well in sentiment analysis tasks and serves as a strong baseline model. (*1.9. Naive Bayes — Scikit-Learn 1.3.0 Documentation*, n.d.)



*Figure 4: Confusion Matrix of MNB Model*

* 1. **XGBoost**

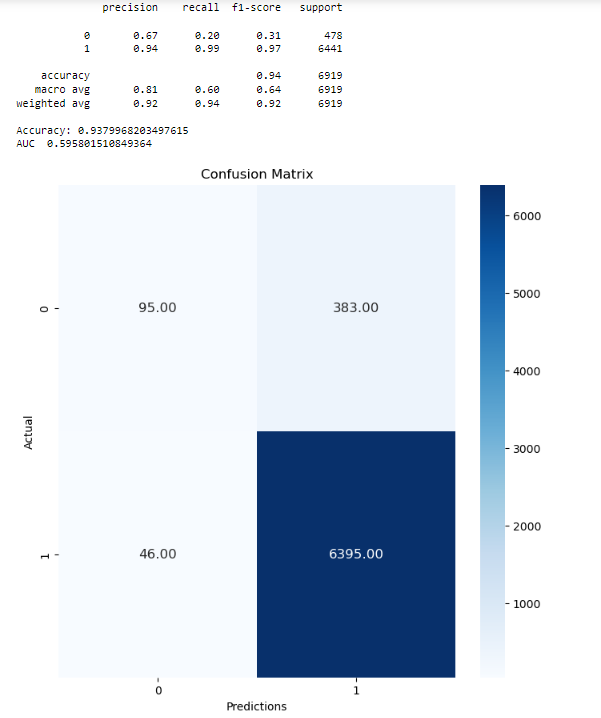
XGBoost can be a powerful and effective algorithm for sentiment analysis on texts, especially when combined with appropriate preprocessing techniques and feature engineering. (*XGBoost Documentation — Xgboost 2.0.0-Dev*, n.d.)



*Figure 5: Confusion Matrix of XGBoost*

* 1. **Logistic Regression**

Logistic regression is a binary classification algorithm, meaning it can classify instances into two classes (positive and negative sentiment in this case). It works well when there is a linear relationship between the features and the log-odds of the target class. While it's a simple and interpretable model, it may not capture complex interactions or non-linear relationships in the data. (Neal, n.d.)

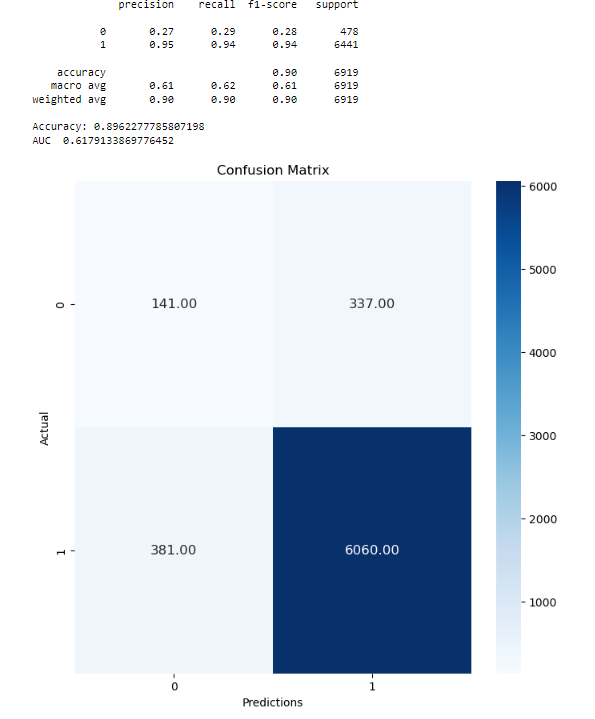


*Figure 6: Confusion Matrix of Logistic Regression*

* 1. **Decision Tree (CART)**

Decision trees can also be used for sentiment analysis on Amazon reviews. Decision trees are a popular machine learning algorithm for classification tasks, including sentiment analysis, as they can handle both numerical and categorical data.

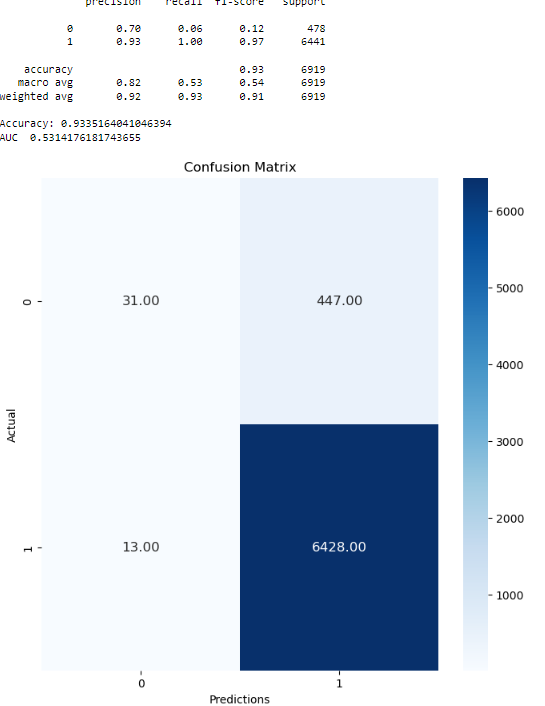
Decision trees are attractive for sentiment analysis as they are easy to interpret and visualize. They can capture nonlinear relationships between features and the target sentiment, and they can handle text data directly without the need for complex feature engineering. However, decision trees can be prone to overfitting, especially on high-dimensional text data. (*1.10. Decision Trees — Scikit-Learn 1.3.0 Documentation*, n.d.)



*Figure 7: Confusion Matrix of Decision Tree*

* 1. **Random Forest Classifier**

Random Forest is a powerful ensemble learning algorithm that can be effectively used for sentiment analysis on Amazon reviews. Ensemble methods like Random Forest can significantly improve the performance of classification tasks like sentiment analysis by combining the predictions of multiple decision trees. (*1.11. Ensemble Methods — Scikit-Learn 1.3.0 Documentation*, n.d.)

**

*Figure 8: Confusion Matrix of Random Forest Classifier*

# **Results**

In this project, we performed sentiment analysis on Amazon reviews to determine the sentiment (positive or negative) expressed in the reviews. We compared multiple machine learning models to find the best-performing one for this task. The dataset consisted of a collection of Amazon reviews, and the goal was to accurately classify each review as either positive or negative sentiment.

Data Preprocessing:

1. Data Cleaning: To prepare the text data for analysis, we performed data cleaning to remove any noise and irrelevant information from the reviews. The cleaning process included converting text to lowercase, removing special characters, and eliminating any punctuation.
2. Tokenization: Tokenizing the text data, which involved breaking down each review into individual words or tokens. This step was essential for further processing the text data.

Feature Selection:

TF-IDF: After data preprocessing, we used the TF-IDF (Term Frequency-Inverse Document Frequency) technique to convert the text data into numerical features. TF-IDF represents the importance of words in the reviews and is commonly used for text analysis tasks.

Model Comparison:

|  |  |
| --- | --- |
| **Model Name** | **Accuracy in %** |
| Multinominal Naive Bayes | 93.11 |
| XGBoost | 93.81 |
| Logistic Regression Classifier | 93.8 |
| Decision Tree Classifier | 89.62 |
| Random Forest Classifier | 93.35 |

Model Selection:

Among the models tested, XGBoost, Logistic Regression, and Random Forest performed exceptionally well, all achieving an accuracy close to 94%. These models outperformed Multinomial Naive Bayes and Decision Tree Classifier in accuracy.

The results demonstrate that ensemble models like Random Forest and gradient boosting models like XGBoost are highly effective for sentiment analysis on Amazon reviews, providing accurate and robust sentiment classification.

# **Conclusion**

In the ever-evolving retail landscape, deciphering customer sentiments is pivotal for success. This report has delved into the realm of text mining to facilitate insightful extraction from product reviews. The overarching goal was to create a potent text-mining strategy, which involves a full-stack approach from dataset curation to machine learning algorithm selection.

The dataset, sourced from Datafiniti's Product Database, encapsulated over 34,000 consumer reviews for various Amazon products. Enriched with attributes like product details, ratings, and review text across 21 columns, this dataset formed the foundation for our analysis.

Data preparation was meticulous, encompassing data cleaning, tokenization, and feature selection using TF-IDF. This step ensured the text data was transformed into a suitable format for machine learning algorithms.

Our journey through algorithm exploration unveiled intriguing insights. Models like XGBoost, Logistic Regression, and Random Forest demonstrated exceptional accuracy, hovering around 94%. These results underscored the effectiveness of ensemble models and gradient-boosting techniques for sentiment analysis in the context of Amazon reviews.

In conclusion, this report has not only introduced a comprehensive text-mining strategy but also provided empirical evidence of its viability through accurate sentiment classification. By integrating data-driven insights, retailers can navigate the complex landscape armed with a robust toolset for customer sentiment analysis, ultimately steering decisions towards enhanced customer experiences and informed retail strategies.

# **References**

Datafiniti. (2019, May 20). Consumer reviews of Amazon Products. Kaggle. <https://www.kaggle.com/datasets/datafiniti/consumer-reviews-of-amazon-products>

Neal, R. M. (n.d.). *1.1. Linear Models — scikit-learn 1.3.0 documentation*. Scikit-learn. Retrieved August 3, 2023, from <https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression>

*1.9. Naive Bayes — scikit-learn 1.3.0 documentation*. (n.d.). Scikit-learn. Retrieved August 3, 2023, from <https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes>

*1.11. Ensemble methods — scikit-learn 1.3.0 documentation*. (n.d.). Scikit-learn. Retrieved August 3, 2023, from <https://scikit-learn.org/stable/modules/ensemble.html#forests-of-randomized-trees>

*1.10. Decision Trees — scikit-learn 1.3.0 documentation*. (n.d.). Scikit-learn. Retrieved August 3, 2023, from <https://scikit-learn.org/stable/modules/tree.html>

*XGBoost Documentation — xgboost 2.0.0-dev*. (n.d.). XGBoost Documentation. Retrieved August 3, 2023, from <https://xgboost.readthedocs.io/en/latest/>