SENTIMENT ANALYSIS ON AMANZON REVIEWS USING MACHINE LEARNING

Project report submitted in partial fulfillment for the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

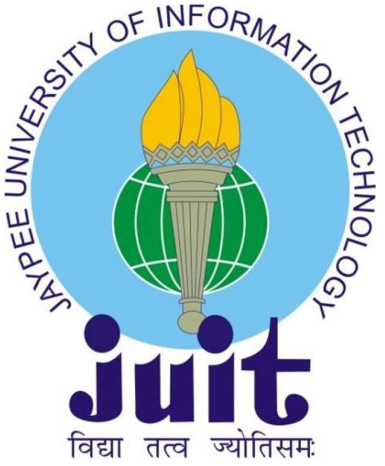
By

Ananya Joshi (191218)
Vipasha Rana (191226)

Under the supervision of

Dr. Ekta Gandotra

to



Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

CERTIFICATE

This is to certify that the work which is being presented in the project report titled "Sentiment Analysis on Amazon Reviews using Machine Learning" in partial fulfilment of the requirements for the award of the degree of B. Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by "Ananya Joshi, 191218 and Vipasha Rana, 191226" during the period from January 2022 to May 2022 under the supervision of Dr. Ekta Gandotra, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Ananya Joshi
(191218)

Vipasha Rana
(191226)

The above statement made is correct to the best of my knowledge.

(Dr. Ekta Gandotra)
Associate Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat, Solan, HP.

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><th>Copy Received on</th><th>Excluded</th><th>Similarity Index (%)</th><th colspan="2">Generated Plagiarism Report Details (Title, Abstract & Chapters)</th></tr>
<tr><td rowspan="2"></td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td rowspan="2">Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes it possible to complete the project work successfully.

I am really grateful and wish my profound indebtedness to Supervisor **Dr. Ekta Gandotra Associate Professor**, Department of CSE Jaypee University of Information Technology,Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of "**Sentiment Analysis on Amazon Reviews using Machine Learning**" to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages have made it possible to complete this project.

It is my radiant sentiment to place on record my best regards, deepest sense of gratitude to all those individuals who have helped me straightforwardly or in a roundabout way in making this internship easier. In this unique situation, I might want to thank the various team members, colleagues , who have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Ananya Joshi
(191218)

Vipasha Rana
(191226

TABLE OF CONTENT

LIST OF ABBREVIATIONS

S.No.	Abbreviation	Full Form
1.	CNN	Convolutional Neural Network
2.	ML	Machine Learning
3.	DL	Deep Learning
4.	NLP	Natural Language Processing
5.	XGA	Extended Graphics Array
6.	GHz	Gigahertz
7.	HTTP	HyperText Transfer Protocol
8.	XML	eXtensible Markup Language
9	HTML	Hyper Text Markup Language

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

With the advent of social media, people are now more comfortable than ever to express their thoughts, opinions, and emotions online. The proliferation of these comments, whether positive or negative, makes it crucial to analyse them accurately in order to grasp the true intentions of the writer. To achieve this, sentiment analysis is used to decipher the perspective of the text. In our study, we propose a novel approach that takes into account the sentimental aspects of the item being reviewed. To validate our approach, we utilized Amazon consumer reviews, specifically the Amazon musical Instruments Reviews dataset collected from the Kaggle repository by Eswar Chand. In this dataset, user ratings were initially detected in each analysis, after which we conducted pre-processing operations, such as creating a sentiment column, tokenization, reviewing text-punctuation cleaning, and eliminating stop-words to extract meaningful information such as the positivity or negativity of the feedback. Our main goal was to analyse this data on an aspect level, which would be highly beneficial to marketers in comprehending consumer preferences and adapting their strategies accordingly. Furthermore, we also provide insights into possible future work for text classification. Ultimately, our study presents a new approach to sentiment analysis that can enhance our understanding of online feedback and facilitate more effective marketing practices.

KEYWORDS: Amazon Product Reviews, Sentiment Analysis, Machine Learning, Natural Language Processing

Chapter 01: INTRODUCTION

1.1 Introduction

In today's digitalized world, eCommerce is quickly taking over as the primary mode of product availability for customers. eCommerce websites provide an ideal platform for people to express their thoughts and feelings about products they have purchased or are considering buying. In fact, customers are increasingly relying on the experiences of other customers to make informed purchasing decisions. Our opinions and purchasing decisions are often shaped by the experiences of others and the feedback they provide. Therefore, the importance of reviews has grown significantly.

However, with the abundance of reviews available, it is almost impossible for customers to read them all, which is where sentiment analysis plays an essential role in analyzing these reviews. Sentiment analysis is the process of evaluating text data and extracting the sentiment element from that field. This research proposes a sentiment analysis approach to predict the polarity of Amazon mobile phone dataset reviews using supervised machine learning algorithms. The goal of this approach is to help customers make better-informed purchasing decisions based on the experiences of others. Additionally, it can help companies improve their products by understanding their customers' opinions and needs.

Online stores such as Amazon provide a website where consumers can express their opinions about various products. In fact, it has been established that approximately 90% of consumers test different websites and channels to determine the quality of their purchase before making a final decision. With most people expressing their ideas, views, and opinions over social media, these feedbacks or comments carry an emotion that requires careful analysis to extract meaningful insights. With sentiment analysis, companies can quickly understand the sentiment of their customers and use this information to make

more informed decisions that benefit both their business and their customers.

While sarcasm can often be detected in person, it may not be as evident in online comments or headlines. This poses a challenge for computers, as they attempt to identify sarcastic language through linguistic cues or context. Our team is tackling this issue by developing a sarcasm detector using advanced machine learning techniques through neural networks. The project involves analyzing a collection of newspaper articles labeled as either sarcastic or non-sarcastic, including current sarcastic renditions of events from The Onion. By utilizing these resources, we aim to improve the accuracy of detecting sarcasm in text-based communication.

1.2 Problem Statement

The digitalization of the world has brought about significant changes, and one of the most noticeable changes is the increasing popularity of eCommerce. With the rise of eCommerce, customers now have access to a vast range of products that are within their reach. Additionally, eCommerce websites enable customers to express their thoughts and feelings about products. In fact, customers are increasingly relying on the experiences of other customers when making purchasing decisions. The opinions and feedback of others have a significant impact on our purchasing decision-making processes. We ask for opinions and experiences of others to benefit from their knowledge, hence the growing importance of product reviews.

However, with the vast number of product reviews available online, it is almost impossible for customers to read them all. Therefore, sentiment analysis plays a crucial role in analyzing these reviews. Sentiment analysis uses natural language processing and machine learning algorithms to determine the emotional tone of text data. By analyzing the sentiment of customer reviews, we can gain valuable insights into customers' opinions and experiences. This information can then be used by companies to improve their products and

services to meet the needs and preferences of their customers better.

This research proposes a sentiment analysis approach to predict the polarity of Amazon mobile phone dataset reviews using supervised machine learning algorithms. The approach focuses on analyzing the emotional tone of the reviews and predicting whether they are positive or negative. Sentiment analysis can help customers make informed purchasing decisions by providing them with a comprehensive understanding of the product's strengths and weaknesses. Moreover, it can help companies identify areas of improvement in their products by analyzing the feedback of customers. Overall, sentiment analysis represents an essential tool for eCommerce businesses to better understand their customers and improve their products and services to meet their needs and preferences.

1.3 Objectives

The aim of customer reviews and ratings is to convey the writer's attitude towards a product, which can be either positive, negative, or neutral. Some individuals award products with four or five stars to express their complete satisfaction, while others give one or two stars to convey their dissatisfaction. This poses no challenge in sentiment analysis. However, some people award three stars, despite expressing their satisfaction with the product, which can be confusing for businesses and other customers who seek to understand their genuine opinion. Therefore, analyzing reviews and comprehending customer satisfaction becomes challenging for both businesses and customers. Consequently, the three-star rating may not truly represent a neutral sentiment since those who assign a 3-star rating to a product may not necessarily have a balanced opinion between positive and negative.

Based on this premise, this research proposes the use of sentiment analysis to predict the polarity of Amazon mobile phone dataset reviews. The three-star rating will be considered neutral, with the intention of increasing the difficulty of the study and measuring the efficacy of state-of-the-art NLP models, such as

BERT, in solving complex classification problems. Moreover, four machine-learning models with diverse feature extraction techniques, namely Logistic Regression, Naïve Bayes, Random Forest, and Bi-LSTM, will be utilized in this research. Subsequently, the best-performing model will be analyzed to investigate its sentiment classification. Finally, the top-performing model will be retrained on the dataset with the neutral class removed, converting the problem into a binary-classification task. The purpose is to assess the extent to which this change affects the model's performance

Sarcasm is a well-known and widely used form of communication, yet even humans can struggle to quickly detect it. This difficulty poses a challenge not only for people but also for computers, as sarcasm can interfere with machine learning tasks such as sentiment analysis and opinion mining if not detected and accounted for. Detecting sarcasm is therefore an important task. To address this challenge, automatic sarcasm detection can be framed as a text classification problem. However, text data is inherently unstructured, and machine learning algorithms cannot process non-numerical data without appropriate feature extraction. Extracting useful information from text data and representing it in a way that supports accurate classification is critical to achieving high classification accuracy.

1.4 Language Used

Python is a high-level, interpretive general-purpose programming language that was created by Guido Van Rossum and was made available in 1991. The design of Python places a strong emphasis on the readability of its code and makes liberal use of whitespace. Its language features and object-oriented methodology are made to help programmers create logical, unambiguous code for both little and major projects. Python has dynamic typing and garbage collection. Programs can be written using procedural, object-oriented, and functional programming paradigms.

1.5 Technical Requirements (Hardware)

Processor: Intel core i5 or above. 64-bit, quad-core, 2.5 GHz minimum per core

Ram: 4 GB or more

Hard disk: 10 GB of available space or more.

Display: Dual XGA (1024 x 768) or higher resolution monitors

Operating system: Windows

Chapter 02: Literature Survey

Sentiment analysis is a fascinating and important field in natural language processing that focuses on understanding human emotions and opinions from text data. In recent years, sentiment analysis has gained significant attention in the e-commerce industry, where online reviews and ratings are critical in influencing consumers' purchasing decisions. Companies need to know how customers feel about their products to improve their offerings and stay competitive in the market.

Various studies have been conducted to explore the use of sentiment analysis in e-commerce. One such study by AlQahtani et al. [1] used machine learning techniques to perform sentiment analysis on the Amazon Reviews Dataset. The authors used various text preprocessing methods such as stop-word elimination, tokenization, stemming, lemmatization, and POS tagging to convert the review text into numerical representations. After training the models with different machine learning algorithms, they selected the best performing model based on multiclass classification analysis and retrained it for binary classification. The study demonstrated that effective text preprocessing is essential in sentiment analysis and can significantly affect the model's performance.

Another study by Aashutosh Bhatt et al. [2] also highlighted the importance of text preprocessing in sentiment analysis. The authors removed stop words from each review as the first step in their preprocessing process and added additional steps based on their specific task at hand. These steps included removing repeated letter strings, stemming, lowercasing, punctuation removal, and tokenization. The authors demonstrated that text preprocessing techniques can vary based on the dataset and the specific task and that a customized preprocessing pipeline can improve model performance. The authors performed tokenization and a spelling check, using the procedures outlined in reference [3], to further refine the gathered data. In [4] K.S. Kumar et. al. performed sentiment analysis on Online customer reviews and did Opinin Mining on the same.

One critical aspect of sentiment analysis is the selection of the appropriate classification algorithm. The study by Sanjay Dey et al. [5] compared the performance of SVM and Naive Bayes classifiers for sentiment analysis of Amazon product reviews. The study found that SVM outperformed Naive Bayes in terms of accuracy and F-measure for both positive and negative sentiments. Similarly, Rabnawaz Jansher [6] presented a machine learning approach for sentiment analysis of Amazon product reviews, using features such as bag-of-words and n-grams, combined with classification algorithms like Naive Bayes and Support Vector Machines. The study reported an accuracy of up to 85% for classifying reviews as positive, negative, or neutral.

There are two primary categories of sentiment categorization methods: machine learning methods and lexicon-based approaches. In [7], the Naive Bayes classifier performed better in classifying reviews as positive or negative than Logistic Regression and SentiWordNet. The experiment's findings were evaluated using Precision, Recall, and F1 Score. In [8], Moraes, R. et. al. subjected the dataset to tokenization and stemming and converted full-width numerals and alphabetic characters to half-width characters. The authors of the passage first extracted the meaning of individual words by referencing SentiWordNet. Afterwards, they divided phrases into sets of sentences and then sentences into sets of words. To parse the Chinese comments, they utilized the ICTCLAS 4 system, which separated words and assigned their corresponding parts of speech, as specified in reference [9]. They also removed any stop words and punctuation marks from the text, so that the analysis could focus on more meaningful content [10].

In [11], NB was conducted at the phrase and review level of granularity, and the relative frequency of each word was computed using the TF-IDF algorithm. The experiment's findings were assessed at the review level using Accuracy, Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). In [12], the dataset was subjected to stemming, lowercasing, punctuation removal, and tokenization, while spelling checks and lemmatization were performed by Aljuhani, S. A. et. al. [13].

In sentiment analysis and opinion mining, the detection of sarcasm holds significant importance. Machine learning algorithms have been employed to solve this problem. Earlier studies have implemented the Naive Bayes Classifier and Support Vector Machines for data analysis on social media in Indonesia [26]. The researchers applied vector support machines along with Tf-IDF and Bag-Of-Words techniques to detect sarcasm. Davidov et.al [27] also worked on detecting sarcasm through semi-supervised methods on two datasets consisting of Amazon product reviews and tweets from Twitter. They primarily focused on various features including punctuation, sentence syntax, and hashtags used in the text. Their system demonstrated an accuracy rate of over 75% [27]. The classifiers employed in the study [26] were described as effective and the decision to use Naive Bayes and SVM for the analysis was commended [28]. The methodology employed in this study yielded an accuracy rate of approximately 80%. However, other researchers have also attempted to tackle this problem using deep learning techniques [29]. Collecting data for the purpose of detecting sarcasm is a challenging task, especially when dealing with social media data. This study builds upon some of the methods described in previous works and yields even better accuracy results, while also implementing additional steps. In the study conducted by the researcher [30], Support Vector Machines were utilized to detect sarcasm in media news headlines.

Extracting features in this manner is valuable for analyzing text data across all domains. In [31], researchers explore the impact of pre-processing on text extraction and consider factors such as slang usage and spelling accuracy. They utilize an SVM classifier in their experiments. Another researcher proposes a solution to emotional analysis through the use of vector representations, achieving a high accuracy rate of 86% [32]. Other studies examine the use of four different data sets and explore features such as Word Bag, Dictionary, and section-based speech features [33]. These studies employ an integrated model that utilizes SVM, Logistic Regression, and Naive Bayes for analysis. n [34], the authors utilized three different techniques for removing features at varying levels and incorporated three-dimensional dividers into their analysis. In a separate study outlined in [35], a group of researchers employed ten distinct

methods for feature extraction in their work on emotion analysis. They concluded that the problem-solving techniques employed in a given task have the potential to enhance the performance of a segment separator. Their analysis showed that TF-IDF yielded the best results with a difference of up to 4%. Standard monitoring software applications commonly use SVM, Logistic Regression, and Naive Bayes algorithms in research studies, leading to their adoption in this analysis. Additionally, the analysis employed a CNN model that utilized features generated by both Count Vectorizer and word2vec. The process of converting words from text data into useful numeric features for machine learning classifiers is referred to as text extraction [36].

In this paper, we compared five machine learning algorithms for text classification on Amazon product reviews and observed which model gives us the best outcomes. Our study included multiple steps, such as data collection, data preprocessing, feature selection, model training, and evaluation. We used a dataset of 10,000 reviews and compared the performance of Random Forest, Multinomial Naive Bayes, Logistic Regression, Linear Support Vector Machine, and Decision Tree algorithms. We found that Logistic Regression gives the highest accuracy of 94%. Few of the research papers studied for this project have been reviewed below in Table 1.

Table I: Literature Review

S. No.	Author(s)	Contribution	Performance Parameter	Limitations
1	Arwa S.M. AlQahtani [1]	Analysis of the Amazon mobile phone product sentiment using four machine-learning models and various feature extraction	Accuracy, F Measure and Cross-Entropy	Double meaning connotation of words has not been taken into consideration.

		techniques.		
2	Aashutosh Bhatt, Ankit Patel, and Kiran Gawande [2]	Amazon Review Classification and Sentiment Analysis.	Accuracy, Recall, Precision, F1-score	Traditional sentiment analysis approaches apart from SVM have not been comparatively studied.
3	Sobia Wassana et. al. [3]	Performed Sentiment Analysis on Amazon Products using certain Machine Learning Techniques.	Recall, Accuracy	This study may be expanded to apply to the other Amazon datasets for further review system-related topics.
4	K.S. and D.J. and M.J. Kumar [4]	Online customer reviews: Opinion Mining and Sentiment Analysis	Precision, Accuracy.	The task of mining reviews from many websites might be expanded. to include more classification methods.
5	Sanjay Dey,	Use of SVM and Naive Bayes	Precision, Recall, f-	Other models can be

	Sarhan Wasif and Subrina Sultana [5]	Classifier for Sentiment Analysis of Amazon Product Reviews: A Comparative Study	measure, AUC	experimented on this dataset.
6	Rabnawaz Jansher [6]	Sentimental Analysis of Amazon Product Reviews Using Machine Learning Approach	Accuracy, precision, recall, F1	Apart from ratings, reviews could also be given weightage.

Chapter 03: System Development

3.1 Date Set Used

It's common for web portals to receive a lot of user feedback. It might be laborious to read through every comment. Opinions shared in discussion boards must be categorized. For a feedback management system, this can be used. Individual comments and reviews are categorized, and we use individual comments and reviews to get the total rating. Consequently, the business will have a thorough understanding of the client feedback and will be able to handle those specific fields. As a result, the firm grows in size, notoriety, brand value, and revenues, and its customers become more devoted.

3.2 Date Set Features

The dataset used in the study consists of reviews of various musical instruments from Amazon, and it was obtained from a Kaggle repository by Eswar Chand. The reviews were originally extracted from web portals such as Bhuvan.

The dataset contains several columns, including the reviewer ID, user ID, reviewer name, reviewer text, helpfulness rating, summary (obtained from the reviewer text), overall rating on a scale of 5, and review time. The reviewer ID and user ID columns are unique identifiers for each reviewer and user, respectively. The reviewer name column contains the name of the person who wrote the review. The reviewer text column contains the actual text of the review, while the helpfulness rating column indicates how helpful the review was to other users.

The summary column contains a summary of the review text, which is usually a brief sentence or phrase that captures the main point of the review. The overall rating column provides a rating of the product on a scale of 1 to 5. Finally, the

review time column indicates when the review was written.

This dataset is useful for studying sentiment analysis because it provides a large number of reviews with ratings and text data, which can be used to train machine learning models to predict the sentiment of the reviews. The dataset can also be used to study other aspects of customer reviews, such as helpfulness and the impact of summary text on overall ratings.

- ID of the reviewer: reviewerID (e.g. A2SUAM1J3GNN3B)
- ID of the product: asin (e.g. 0000013714)
- Name of the reviewer: reviewerName
- Helpfulness rating of the review: helpful (e.g. 2/3)
- Text of the review: reviewText
- Rating of the product: overall
- Summary of the review: summary
- Time of the review in unix format: unixReviewTime
- Time of the review in raw format: reviewTime

```
raw_reviews = pd.read_csv('/content/drive/MyDrive/sentiment_analysis/Musical_instruments_review.csv')
## print shape of dataset with rows and columns and information
print ("The shape of the  data is (row, column):"+ str(raw_reviews.shape))
print (raw_reviews.info())
```

```
The shape of the  data is (row, column):(10261, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10261 entries, 0 to 10260
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   reviewerID      10261 non-null  object 
 1   asin            10261 non-null  object 
 2   reviewerName    10234 non-null  object 
 3   helpful         10261 non-null  object 
 4   reviewText      10254 non-null  object 
 5   overall         10261 non-null  int64  
 6   summary         10261 non-null  object 
 7   unixReviewTime  10261 non-null  float64
 8   reviewTime      10261 non-null  object 
dtypes: float64(1), int64(1), object(7)
memory usage: 721.6+ KB
None
```

Figure. 3.1. Dataset features shown through code

3.3 Data-Flow Diagram (DFD)

The proposed methodology has been depicted in Figure 1. Customer reviews are becoming more widely available for a variety of goods and services. Star ratings and review content are the two elements that make up most customer reviews. In this study, we will classify reviews as Positive, Negative, or Neutral using consumer feedback and ratings.

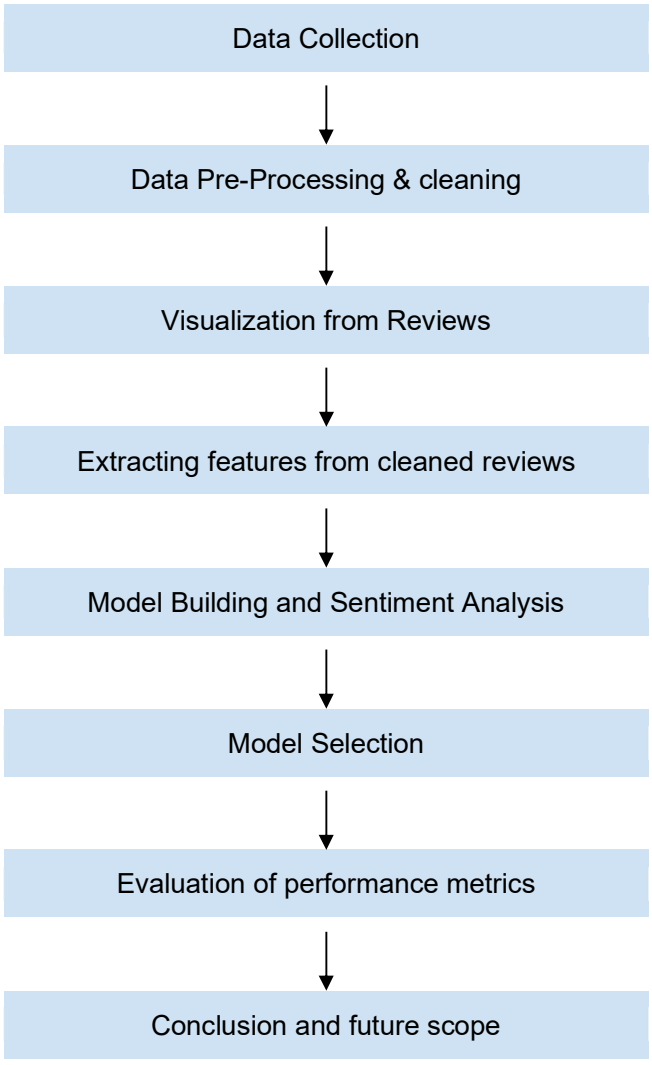


Fig. 3.2. Flowchart of methodology used

3.4 Design of Problem Statement

A. Data Collection

In this work, the data used for Amazon product reviews sentiment analysis was obtained from the Amazon Musical Instruments Reviews dataset available on Eswar Chand's Kaggle repository [11]. However, in order to increase the amount of data, web scraping was also performed. The two datasets were integrated into the final dataset that was used for the analysis.

Web scraping is a technique that involves extracting data from websites. The process of data extraction involves several steps, which are illustrated in Figure 3.2.

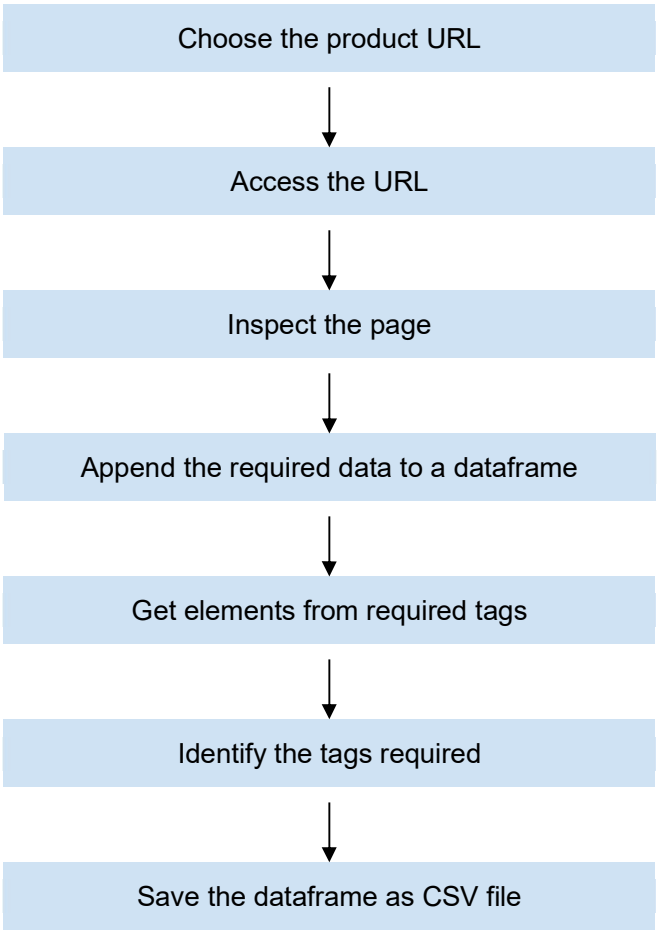


Fig. 3.3. Data Collection

Web scraping involves extracting data from websites and is a useful tool for collecting large amounts of data for analysis. To perform web scraping, several Python libraries are commonly used, including re, requests, and BeautifulSoup.

The re library is a powerful tool for performing regular expression operations in Python. It allows users to search for specific types of words or numbers in a string, making it an essential package for string manipulation in web scraping.

With the re library, users can quickly extract relevant data from HTML and XML files.

The requests library is another important package for web scraping in Python. It allows users to send HTTP requests, making it easy to access data from websites. The requests library provides several functions that range from providing arguments in URLs to delivering custom headers and SSL verification. These features make it a flexible and powerful tool for web scraping. Finally, the BeautifulSoup library is an excellent choice for parsing HTML and XML files in Python. It provides an intuitive way to discover, locate, and modify the parse tree, making it easy to extract the required data from web pages. With BeautifulSoup, users can identify the HTML tags that contain the data they need and extract it quickly and efficiently.

Overall, these three packages are essential tools for web scraping in Python. They provide the necessary functionality to extract data from websites, manipulate strings, and parse HTML and XML files.

B. Data Pre-processing

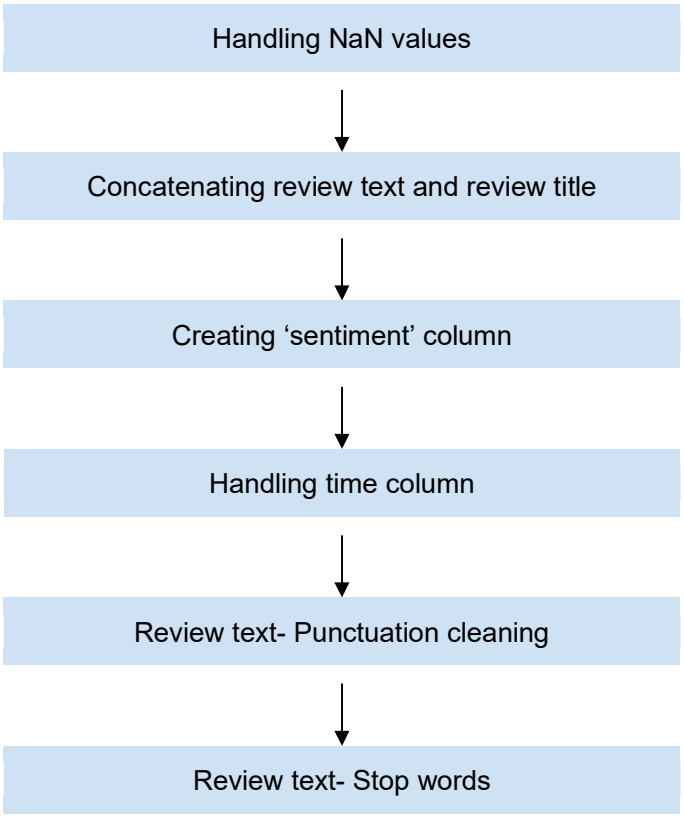


Fig. 3.4. Data Pre-processing and Cleaning

The data has got to be rigorously pre-processed before sending the reviews to the model. The flowchart of data preprocessing is represented in Figure 3.4.

1. Handling NaN values: Firstly, we handle null values which were essentially present in review name and review text as seen below in Figure 3.5. Review text is what is of importance to us as that is what helps us determine the sentiment of a customer while purchasing the product.

```
[ ]  #Creating a copy
     process_reviews=raw_reviews.copy()

     #Checking for null values
     process_reviews.isnull().sum()
```

```
reviewerID         0
asin               0
reviewerName      27
helpful            0
reviewText         7
overall            0
summary            0
unixReviewTime     0
reviewTime         0
dtype: int64
```

Fig. 3.5. Handling NaN values

2. Concatenating Review text and Review Title: Review text and review title are merged into one column so that the sentiments won't be contradicting in nature which is visualized in Figure 3.6.

```
[ ]  process_reviews['reviews']=process_reviews['reviewText']+process_reviews['summary']
     process_reviews=process_reviews.drop(['reviewText', 'summary'], axis=1)
     process_reviews.head()
```

	reviewerID	asin	reviewerName	helpful	overall	unixReviewTime	reviewTime
0	A2IBPI20UZIR0U	1.38E+09	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	5	1.390000e+09	02 28, 2014
1	A14VAT5EAX3D9S	1.38E+09	Jake	[13, 14]	5	1.360000e+09	03 16, 2013
2	A195EZSQDW3E21	1.38E+09	Rick Bennette "Rick Bennette"	[1, 1]	5	1.380000e+09	08 28, 2013
3	A2C00NNG1ZQQG2	1.38E+09	RustyBill "Sunday Rocker"	[0, 0]	5	1.390000e+09	02 14, 2014
4	A94QU4C90B1AX	1.38E+09	SEAN MASLANKA	[0, 0]	5	1.390000e+09	02 21, 2014

Fig. 3.6. Concatenating Review text and Review Title

3. Creating ‘sentiment’ column: Creating a sentiment column is a very important phase of pre-processing in which the outcome column, which determines the sentiment of reviews, is decided based on the overall score. We classify the reviews as positive, negative or neutral. If the overall score is greater than 3, we take that as positive, if it is less than 3 it is considered negative. If it is equal to 3 then it is taken as a neutral sentiment. This can be seen as depicted in Figure 3.7 and Figure 3.8.

```
[ ] #Figuring out the distribution of categories
    process_reviews['overall'].value_counts()
```

```
5    6938
4    2084
3     772
2     250
1     217
Name: overall, dtype: int64
```

```
[ ] def f(row):

        '''This function returns sentiment value based on the overall ratings from the user'''

        if row['overall'] == 3.0:
            val = 'Neutral'
        elif row['overall'] == 1.0 or row['overall'] == 2.0:
            val = 'Negative'
        elif row['overall'] == 4.0 or row['overall'] == 5.0:
            val = 'Positive'
        else:
            val = -1
        return val
```

```
[ ] process_reviews['sentiment'].value_counts()
```

```
Positive    9022
Neutral      772
Negative     467
Name: sentiment, dtype: int64
```

Fig. 3.7-3.8. Creating ‘sentiment’ column

4. Handling Time Column: Further time column is handled, it has date and year which once done split it further into month and date as can be seen below in Figure 3.9 and Figure 3.10.

```
# new data frame which has date and year
new = process_reviews["reviewTime"].str.split(",", n = 1, expand = True)

# making separate date column from new data frame
process_reviews["date"]= new[0]

# making separate year column from new data frame
process_reviews["year"]= new[1]

process_reviews=process_reviews.drop(['reviewTime'], axis=1)
process_reviews.head()
```

	reviewerID	asin	reviewerName	helpful	overall	unixReviewTime	reviews
0	A2IBPI20UZIR0U	1.38E+09	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	5	1.390000e+09	Not much to write about here, but it does exac...
1	A14VAT5EAX3D9S	1.38E+09	Jake	[13, 14]	5	1.360000e+09	The product does exactly as it should and is q...
2	A195EZSQDW3E21	1.38E+09	Rick Bennette "Rick Bennette"	[1, 1]	5	1.380000e+09	The primary job of this device is to block the...
3	A2C00NNG1ZQQG2	1.38E+09	RustyBill "Sunday Rocker"	[0, 0]	5	1.390000e+09	Nice windscreen protects my MXL mic and preven...
4	A94QU4C90B1AX	1.38E+09	SEAN MASLANKA	[0, 0]	5	1.390000e+09	This pop filter is great. It looks and perform...

Fig. 3.9. Handling Date and Time column

```
[ ] # Splitting the date
    new1 = process_reviews["date"].str.split(" ", n = 1, expand = True)

    # adding month to the main dataset
    process_reviews["month"]= new1[0]

    # adding day to the main dataset
    process_reviews["day"]= new1[1]

    process_reviews=process_reviews.drop(['date'], axis=1)
    process_reviews.head()
```

	reviewerID	asin	reviewerName	helpful	overall	unixReviewTime	reviews
0	A2IBPI20UZIR0U	1.38E+09	cassandra tu "Yeah, well, that's just like, u...	[0, 0]	5	1.390000e+09	Not much to write about here, but it does exac...
1	A14VAT5EAX3D9S	1.38E+09	Jake	[13, 14]	5	1.360000e+09	The product does exactly as it should and is q...
2	A195EZSQDW3E21	1.38E+09	Rick Bennette "Rick Bennette"	[1, 1]	5	1.380000e+09	The primary job of this device is to block the...
3	A2C00NNG1ZQQG2	1.38E+09	RustyBill "Sunday Rocker"	[0, 0]	5	1.390000e+09	Nice windscreen protects my MXL mic and preven...
4	A94QU4C90B1AX	1.38E+09	SEAN MASLANKA	[0, 0]	5	1.390000e+09	This pop filter is great. It looks and perform...

Fig. 3.10. Handling Month and Day column

5. Review text- Punctuation cleaning: Review text punctuation cleaning removes all punctuations which are un-necessary for calculating our sentiment. Figure 3.11 is a screenshot depicting the same.

```
[ ] #dropping the helpful column from main dataframe
    process_reviews=process_reviews.drop(['helpful'], axis=1)
```

```
[ ] #Removing unnecessary columns
    process_reviews=process_reviews.drop(['reviewerName','unixReviewTime'], axis=1)
    #Creating a copy
    clean_reviews=process_reviews.copy()
```

```
[ ] def review_cleaning(text):
        '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
        and remove words containing numbers.'''
        text = str(text).lower()
        text = re.sub('\[.*?\]', '', text)
        text = re.sub('https?://\S+|www\.\S+', '', text)
        text = re.sub('<.*?>+', '', text)
        text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
        text = re.sub('\n', '', text)
        text = re.sub('\w*\d\w*', '', text)
        return text
```

Fig. 3.11. Review text- Punctuation cleaning

6. Review text- Stop words: When it comes to stop words, the common nltk list includes terms like not, hasn't, and wouldn't, as shown in Figure 3.12 and Figure 3.13, which really express a bad attitude. That will conflict with the target variable if we remove it (sentiment). Therefore, I have chosen stop words that don't have any bad connotations or negative alternatives.

```
[ ] process_reviews['reviews']=process_reviews['reviews'].apply(lambda x:review_cleaning(x))
    process_reviews.head()
```

	reviewerID	asin	overall	reviews	sentiment	year	month	day
0	A2IBPI20UZIR0U	1.38E+09	5	not much to write about here but it does exact...	Positive	2014	02	28
1	A14VAT5EAX3D9S	1.38E+09	5	the product does exactly as it should and is q...	Positive	2013	03	16
2	A195EZSQDW3E21	1.38E+09	5	the primary job of this device is to block the...	Positive	2013	08	28
3	A2C00NNG1ZQQG2	1.38E+09	5	nice windscreen protects my mxl mic and preven...	Positive	2014	02	14
4	A94QU4C90B1AX	1.38E+09	5	this pop filter is great it looks and performs...	Positive	2014	02	21

```
stop_words= ['yourselves', 'between', 'whom', 'itself', 'is', "she's", 'up', 'herself', 'here', 'your', 'each',
             'we', 'he', 'my', "you've", 'having', 'in', 'both', 'for', 'themselves', 'are', 'them', 'other',
             'and', 'an', 'during', 'their', 'can', 'yourself', 'she', 'until', 'so', 'these', 'ours', 'above',
             'what', 'while', 'have', 're', 'more', 'only', "needn't", 'when', 'just', 'that', 'were', "don't",
             'very', 'should', 'any', 'y', 'isn', 'who',  'a', 'they', 'to', 'too', "should've", 'has', 'before',
             'into', 'yours', "it's", 'do', 'against', 'on',  'now', 'her', 've', 'd', 'by', 'am', 'from',
             'about', 'further', "that'll", "you'd", 'you', 'as', 'how', 'been', 'the', 'or', 'doing', 'such',
             'his', 'himself', 'ourselves',  'was', 'through', 'out', 'below', 'own', 'myself', 'theirs',
             'me', 'why', 'once',  'him', 'than', 'be', 'most', "you'll", 'same', 'some', 'with', 'few', 'it',
             'at', 'after', 'its', 'which', 'there','our', 'this', 'hers', 'being', 'did', 'of', 'had', 'under',
             'over','again', 'where', 'those', 'then', "you're", 'i', 'because', 'does', 'all']
```

```
[ ] process_reviews['reviews'] = process_reviews['reviews'].apply(lambda x: ' '
    .join([word for word in x.split() if word not in (stop_words)]))
    process_reviews.head()
```

	reviewerID	asin	overall	reviews	sentiment	year	month	day
0	A2IBPI20UZIR0U	1.38E+09	5	not much write but exactly supposed filters po...	Positive	2014	02	28
1	A14VAT5EAX3D9S	1.38E+09	5	product exactly quite affordablei not realized...	Positive	2013	03	16
2	A195EZSQDW3E21	1.38E+09	5	primary job device block breath would otherwis...	Positive	2013	08	28
3	A2C00NNG1ZQQG2	1.38E+09	5	nice windscreen protects mxl mic prevents pops...	Positive	2014	02	14
4	A94QU4C90B1AX	1.38E+09	5	pop filter great looks performs like studio fi...	Positive	2014	02	21

Fig. 3.12-3.13. Review text- Stop words

C. Story Generation and Visualization

In this phase, we do a thorough exploratory data analysis on texts and other elements to determine which characteristics collectively contribute to the sentiment. It is also depicted by Figures 3.14-3.26 under each point below.

1. **Year vs Sentiment count**: This block will display the number of reviews based on sentiments that were posted in each year from 2004 to 2014. Year Vs Sentiment count is depicted below in Fig. 3.16. The plot makes it evident that the number of favorable reviews has increased since 2010. All the review rates declined at this point as the trend peaked about 2013 and began to decline.

Reviews that are neutral or negative are far less common than reviews that are good.

```
process_reviews.groupby(['year','sentiment'])['sentiment'].count()
.unstack().plot(legend=True)
plt.title('Year and Sentiment count')
plt.xlabel('Year')
plt.ylabel('Sentiment count')
plt.show()
```

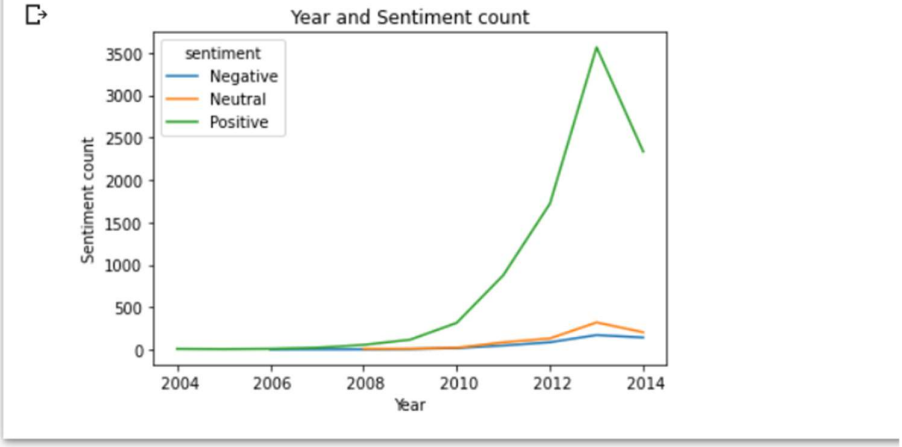


Fig. 3.14. Year vs Sentiment Count

2. **Day of Month vs Review Count**: To investigate any possible relationship between the day of the month and the number of reviews, a plot was created. The graph illustrates that the number of positive reviews has been consistently increasing since 2010. However, all review rates, including those for neutral and negative reviews, have decreased since their peak in 2013. Despite this decrease, positive reviews remain the most common type of review. Reviews with neutral or negative sentiments are relatively rare when compared to reviews with a positive sentiment.

```
[ ]  #Creating a dataframe
     day=pd.DataFrame(process_reviews.groupby('day')['reviews']
                      .count()).reset_index()
     day['day']=day['day'].astype('int64')
     day.sort_values(by=['day'])

     #Plotting the graph
     sns.barplot(x="day", y="reviews", data=day)
     plt.title('Day vs Reviews count')
     plt.xlabel('Day')
     plt.ylabel('Reviews count')
     plt.show()
```

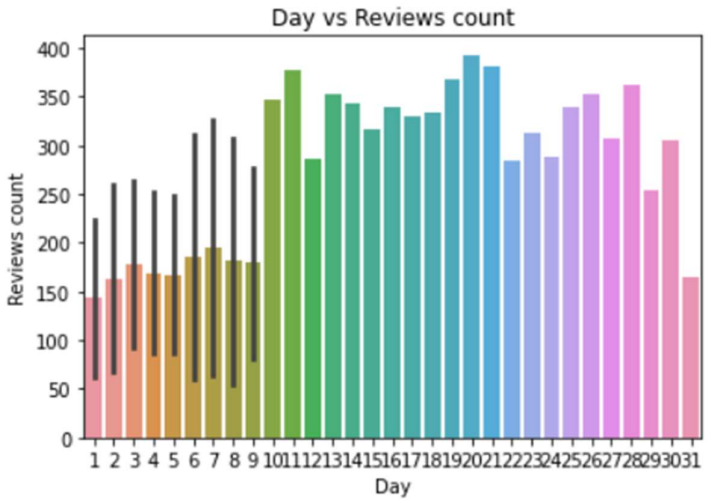


Fig. 3.15-3.16. Day of Month vs Review Count

3. **Creating few more features for text analysis:** In order to analyze the reviews, we calculated several metrics including polarity, review duration, and word count. Polarity was calculated using Textblob, which assigns a sentiment score between -1 and 1, with -1 indicating a negative sentiment and 1 indicating a positive sentiment. Review duration was determined by measuring the time interval between the review creation date and the review date. Lastly, word count was calculated by counting the total number of words in each review, including all letters and spaces. Word length was also calculated to indicate the average number of words in the reviews. These metrics provide valuable insights into the sentiment and structure of the reviews, which can be used to gain a better understanding of customer feedback.

```
process_reviews['polarity'] = process_reviews['reviews'].map(lambda text: TextBlob(text).sentiment.polarity)
process_reviews['review_len'] = process_reviews['reviews'].astype(str).apply(len)
process_reviews['word_count'] = process_reviews['reviews'].apply(lambda x: len(str(x).split()))
```

```
process_reviews.head()
```

	reviewerID	asin	overall	reviews	sentiment	year	month	day
0	A2IBPI20UZIR0U	1.38E+09	5	not much write but exactly supposed filters po...	Positive	2014	02	28
1	A14VAT5EAX3D9S	1.38E+09	5	product exactly quite affordablei not realized...	Positive	2013	03	16
2	A195EZSQDW3E21	1.38E+09	5	primary job device block breath would otherwis...	Positive	2013	08	28
3	A2C00NNG1ZQQG2	1.38E+09	5	nice windscreen protects mxl mic prevents pops...	Positive	2014	02	14
4	A94QU4C90B1AX	1.38E+09	5	pop filter great looks performs like studio fi...	Positive	2014	02	21

```
[ ] cf.go_offline()
    cf.set_config_file(offline=False, world_readable=True)
```

Fig. 3.17. Creation of extra Features

4. **Sentiment Polarity Distribution:** Upon analyzing the data, it is evident that there are significantly more positive sentiments expressed than negative sentiments. This distribution of polarity confirms the prevalence of positive reviews. However, it should be noted that this distribution is not a standard normal distribution, but rather a normal distribution.

```
[ ] process_reviews['polarity'].iplot(
        kind='hist',
        bins=50,
        xTitle='polarity',
        linecolor='black',
        yTitle='count',
        title='Sentiment Polarity Distribution')
```

Fig. 3.18. Sentiment Polarity Distribution

5. **Review Rating Distribution**: The distribution of overall ratings reveals a significant number of ratings with a score of 5, amounting to nearly 7000. This is followed by ratings of 4, 3, 2, and 1, in a linear manner.

```
process_reviews['overall'].iplot(
    kind='hist',
    xTitle='rating',
    linecolor='black',
    yTitle='count',
    title='Review Rating Distribution')
```

Fig. 3.19. Review Rating Distribution

6. **Review Text Length Distribution:** Upon analyzing the dataset, it became apparent that the distribution of review text lengths was skewed towards the right. This means that the majority of reviews were relatively short in length, with fewer reviews being longer in comparison. Specifically, the data revealed that the majority of reviews were between 0 and 1000 characters in length, indicating that customers generally preferred to provide concise feedback rather than lengthy reviews. This information could be useful for companies looking to improve their customer feedback mechanisms by encouraging customers to provide more detailed reviews.

7. **Review Text Word Count Distribution:** Upon analyzing the reviews, it was found that the majority of the reviews contained between 0 and 200 words. This was evidenced by a right-skewed distribution of word count, with fewer reviews containing higher numbers of words. This finding suggests that customers generally preferred to provide concise feedback, with only a small percentage of customers taking the time to write lengthy reviews. Businesses could use this information to improve their customer feedback mechanisms by encouraging customers to provide more detailed feedback, perhaps by offering incentives or rewards for more extensive reviews. Additionally, this data could be used to improve the analysis of customer feedback by identifying common themes and sentiments expressed within concise reviews.

8. **N-gram Analysis:** As part of our deep text analysis, we utilize N-grams to gain further insight into the sentiment expressed in customer reviews.

a. Monogram Analysis: Here, we'll map the one term that appears most frequently in reviews based on sentiments. As can be seen, the words rarely reflect the sentiment. We cannot evaluate a monogram only on a single word to convey a sentiment. So let's try using two words that are used frequently.

b. Bigram Analysis: Here, we'll map the two words that appear most frequently in reviews based on sentiments. As seen below we can get a clear idea about sentiments from the bi-words.

c. Trigram Analysis: Here, we'll map the three words that appear most frequently in reviews based on sentiments.

```
#Filtering data
review_pos = process_reviews[process_reviews["sentiment"]=='Positive'].dropna()
review_neu = process_reviews[process_reviews["sentiment"]=='Neutral'].dropna()
review_neg = process_reviews[process_reviews["sentiment"]=='Negative'].dropna()

## custom function for ngram generation ##
def generate_ngrams(text, n_gram=1):
    token = [token for token in text.lower().split(" ")
    if token != "" if token not in STOPWORDS]
    ngrams = zip(*[token[i:] for i in range(n_gram)])
    return [" ".join(ngram) for ngram in ngrams]

## custom function for horizontal bar chart ##
def horizontal_bar_chart(df, color):
    trace = go.Bar(
        y=df["word"].values[::-1],
        x=df["wordcount"].values[::-1],
        showlegend=False,
        orientation = 'h',
        marker=dict(
            color=color,
        ),
    )
    return trace
```

Fig. 3.20. N-gram Analysis

9. **Word Cloud Positive Reviews:** Upon conducting N-gram analysis on customer reviews, we identified several frequently occurring positive words, such as "great", "pop", "all five", and "exactly". These words were typically associated with positive sentiment and were often used to describe positive experiences or features of the product or service being reviewed. By identifying these frequently occurring positive words, businesses can gain insight into what aspects of their products or services are resonating with customers and use this information to make data-driven decisions to further improve the customer experience. It is important to note, however, that while positive words can provide valuable insight into customer sentiment, it is also essential to analyze

negative words and sentiments to gain a more comprehensive understanding of the customer experience.

```
text = review_pos["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Fig. 3.21. Code Screenshot for Word cloud Positive Reviews

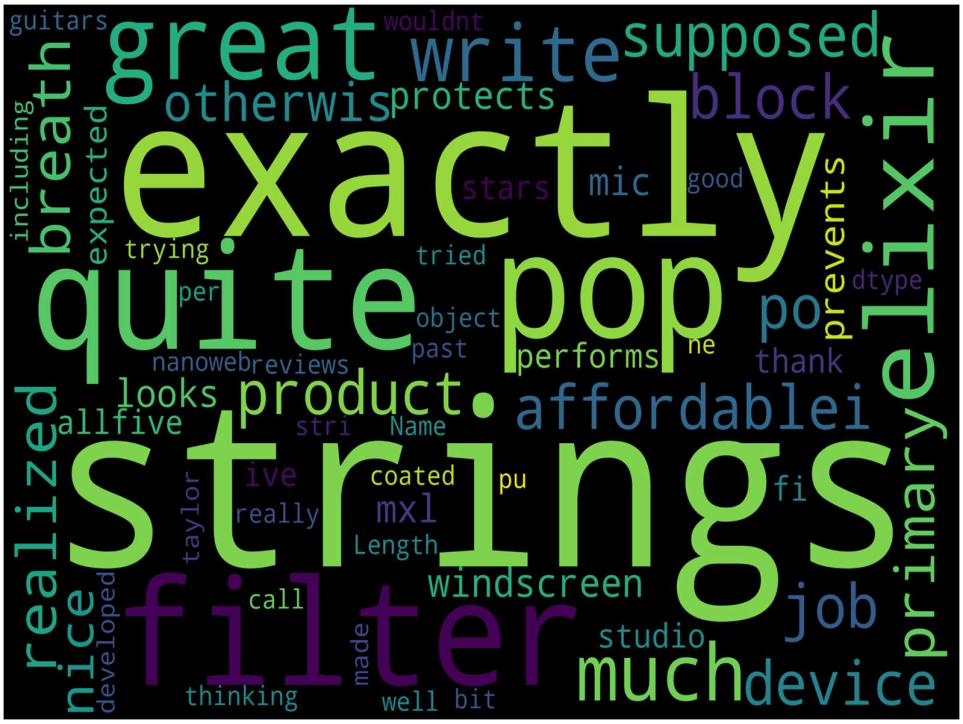


Fig. 3.22. Word Cloud for Positive Reviews

10. **Wordcloud Neutral Reviews:** After analyzing the text of neutral reviews, we generated a word cloud to identify the most frequently occurring words. It became apparent that the majority of neutral reviews focused on the product and

its potential areas of improvement. Customers often used words related to functionality, usability, and performance when discussing the product in neutral reviews. While these reviews did not express either positive or negative sentiment, they still provided valuable feedback to businesses on how they could improve their product or service to better meet customer needs and expectations. By identifying common themes and sentiments expressed in neutral reviews, businesses can make data-driven decisions to improve their products and services, and ultimately provide a better customer experience.

```
text = review_neu["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Fig. 3.23. Code Screenshot for Word cloud Neutral Reviews

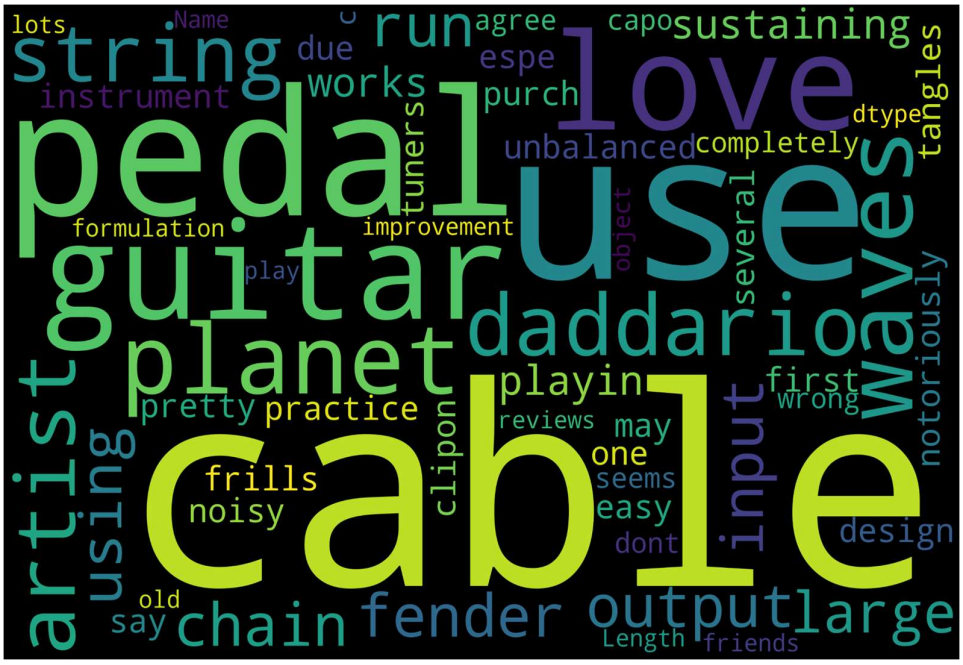


Fig. 3.24. Word Cloud for Neutral Review

11. **Word cloud Negative Reviews:** Upon analyzing the reviews, it is apparent that certain negative words frequently appear in the reviews. These negative words include "noisy," "didn't," "noise," "wasn't," "snap," "problems," "tension," and others. These words suggest that customers have experienced issues with the product or service provided. These negative sentiments can be used to identify areas that require improvement to enhance customer satisfaction. By analyzing the frequency and context of these negative words, we can gain a better understanding of the issues and take appropriate actions to address them.

```
text = review_neg["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = stop_words).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```

Fig. 3.25. Code Screenshot for Word cloud Negative Reviews

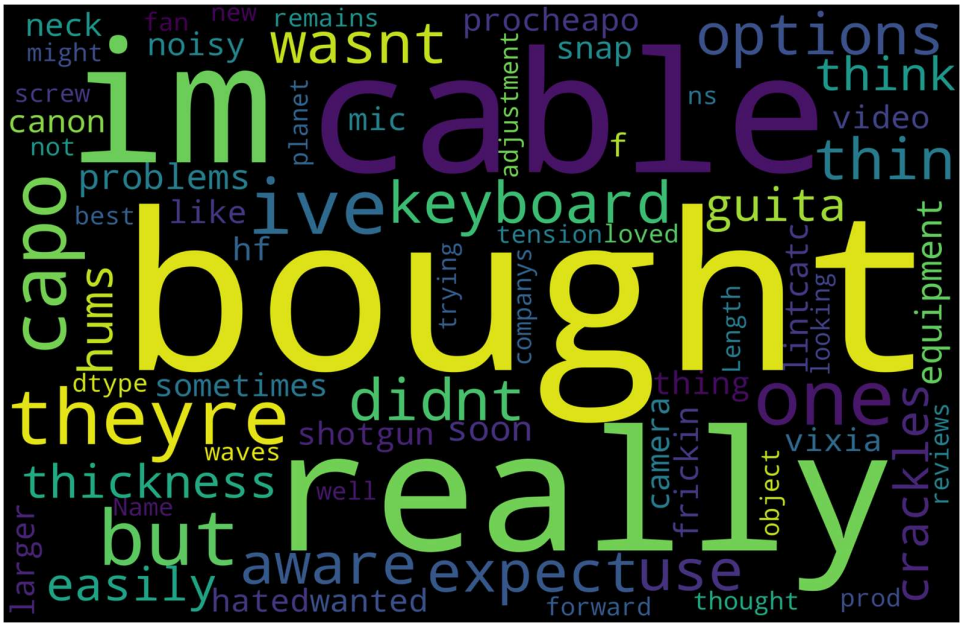


Fig. 3.26. Word Cloud for Negative Reviews

D. Extracting features from Cleaned reviews

In order to develop a model for sentiment analysis, it is necessary to convert the review texts into a vector format, as computers cannot understand words or their sentiment in their natural language form. In this project, we will be using the TF-IDF method to transform the texts into a numerical format that can be easily processed by the machine learning algorithm. TF-IDF stands for "term frequency-inverse document frequency," and it is a commonly used technique for transforming text data into a vector format. This method calculates a weight for each word in the text, based on how frequently it appears in the document and how rare it is across all the documents. The resulting vector representation of the text can then be used as input for the sentiment analysis model, allowing us to classify the sentiment of the reviews with a high degree of accuracy.

1. Encoding Target variable-sentiment: Scikit-learn's LabelEncoder function is used to encode the labels in the 'sentiment' column of a dataframe called 'process_reviews'. This encoding process assigns unique numerical values to each unique label in the 'sentiment' column. The LabelEncoder object is first initialized, and then the fit_transform method is used to encode the labels in the 'sentiment' column. The fit_transform method learns the mapping between the unique labels and their corresponding numerical values. Finally, the unique numerical values that were assigned to the labels in the 'sentiment' column are printed using the unique method. This encoding process is commonly used in machine learning to convert categorical data into numerical data, which can then be used as input to train machine learning models.

```
[ ]  # calling the label encoder function
     label_encoder = preprocessing.LabelEncoder()

     # Encode labels in column 'sentiment'.
     process_reviews['sentiment']= label_encoder.fit_transform(process_reviews['sentiment'])

     process_reviews['sentiment'].unique()

     array([2, 1, 0])

     process_reviews['sentiment'].value_counts()

     2    9022
     1     772
     0     467
     Name: sentiment, dtype: int64
```

Fig. 3.27. Target variable-sentiment encoding

2. Stemming Reviews: One way to get the root word from the inflected term is by stemming. Here, we extract the terms from the reviews and change them to their root words. for instance, going to go and finally to fina. You'll note that the root words don't necessarily need to have a semantic meaning. Another method is called lemmatization, which reduces words to their root words, each of which has a semantic meaning. Since it requires time. We are utilizing stemming. As a machine cannot understand words or their sentiment, we must translate them into 1s and 0s. This is how a line now appears. We use TFIDF to encrypt it.

3. TFIDF(Term Frequency — Inverse Document Frequency): TF-IDF is a widely used technique in natural language processing and information retrieval. It is particularly useful in tasks such as document classification, text summarization, and search engine ranking. By assigning weights to each word based on its relevance to a document and the corpus as a whole, it can help identify important words or phrases and differentiate them from less important ones. Bigrams (pairs of adjacent words) are used instead of individual words. This can help capture more meaningful and informative phrases and expressions that might be missed when analyzing individual words alone. For example, "information retrieval" is a bigram that represents a specific concept that might be more important than the individual words "information" and "retrieval" when

analyzing a text about search engines or document retrieval. Additionally, the top 5000 words from the reviews are used for analysis. This is a common technique in natural language processing to remove stop words and other common words that do not carry much meaning or contribute to the understanding of a text. By focusing only on the most frequent and important words, we can reduce noise and increase the signal-to-noise ratio in our analysis, which can lead to more accurate and meaningful results.

```
[ ] corpus[3]

    'nice windscreen protect mxl mic prevent pop thing gooseneck margin abl hold screen posit requir c

[ ] tfidf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
    # TF-IDF feature matrix
    X= tfidf_vectorizer.fit_transform(review_features['reviews'])

[ ] X.shape

    (10261, 5000)

[ ] #Getting the target variable(encoded)
    y=process_reviews['sentiment']
```

Fig. 3.28. Target Variable Encoding TFIDF

4. We can validate that we have 5000 columns because we took 5000 words into account. The reason why we have 5000 columns in the TF-IDF matrix is because we selected the top 5000 words from the reviews. This is a common technique in natural language processing and text mining to reduce the dimensionality of the matrix while still focusing on the most important and informative words in the corpus. The TF-IDF weight of each word represents its significance to the document and the corpus as a whole, and by using the top 5000 words, we can extract meaningful insights from the data while reducing computational complexity.

5. Handling Imbalance target feature-SMOTE: We saw that there were far more favorable comments about our desired feature than negative or indifferent ones.

In such a situation, it is imperative to balance the classes.In this case, SMOTE (Synthetic Minority Oversampling Technique) is used to address the unbalanced dataset problem. It aims to balance the distribution of classes by randomly increasing minority class samples and duplicating them. SMOTE combines existing minority instances to produce new minority instances. It uses linear interpolation to provide virtual training records for the minority class. These synthetic training records are selected at random from one or more of the k-nearest neighbors for each example in the minority class. The data is recreated after the oversampling process and can then be exposed to a variety of classification algorithms. The screenshot of the same is visible in Figure 3.29.

```
[ ] corpus[3]

    'nice windscreen protect mxl mic prevent pop thing gooseneck margin abl hold screen posit requir c

[ ] tfidf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
    # TF-IDF feature matrix
    X= tfidf_vectorizer.fit_transform(review_features['reviews'])

[ ] X.shape

    (10261, 5000)

[ ] #Getting the target variable(encoded)
    y=process_reviews['sentiment']
```

Fig. 3.29. Target Variable Encoding SMOTE

6. Train- TEST Split: To train and evaluate the performance of our machine learning model, we have divided the dataset into a 75:25 ratio for the train and test sets. This was done using the train-test split function, which randomly splits the dataset into two subsets: one for training the model and the other for testing its performance. By reserving a portion of the dataset for testing, we can evaluate how well the model generalizes to new, unseen data. This approach helps to prevent overfitting, where the model memorizes the training data and performs poorly on new data. By using a train-test split, we can optimize the performance of our machine learning model and ensure that it accurately predicts the sentiment of customer reviews.

E. Model Building

In the field of machine learning, one of the most critical tasks is to classify data into different categories or classes. One common approach is to split the dataset into a training set and a test set using the train test split function. The dataset was split into a training set and a test set using the train test split function, with a 75:25 ratio for the training and test sets, respectively.

Several machine learning methods have been developed to classify data into multiple categories, and these have different strengths and weaknesses. In this context, a variety of methods have been applied, including multinomial Naive Bayes, Gaussian Naive Bayes, Bernoulli Naive Bayes, logistic regression, decision trees, support vector machines, and k-nearest neighbors.

1. For review classification

The following ML algorithms have been used for multiclass classification:

i. Multinomial Naive Bayes: It is a supervised learning algorithm[25] that uses the principles of probability theory to analyze data. The algorithm is particularly useful when the data is represented by a multinomial distribution, which is a scaled version of the data in which each value indicates the quantity of a phrase or its relative frequency. For example, if each of the n feature vectors may take on k distinct values with probability pk, then:

$$P(X_1 = x_1 \cap X_2 = x_2 \cap \ldots \cap X_k = x_k) = \frac{n!}{\prod_i x_i!} \prod_i {p_i}^{x_i}$$

Fig. 3.30. Multinomial Naive Bayes

ii. Gaussian Naive Bayes: it can provide fast and accurate classification results. However, it is important to note that Gaussian Naive Bayes[25] may not perform well when the data does not follow a Gaussian distribution, or when the features are highly correlated. In such cases, other machine learning algorithms may be more appropriate. Additionally, it is important to carefully

select and preprocess the features before applying any machine learning algorithm, including Gaussian Naive Bayes.:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Fig.3. 31. Gaussian Naive Bayes

iii. Bernoulli Naive Bayes: This approach can be useful in tasks such as text classification, where the presence or absence of a certain word in a document can be represented as a binary value. In Bernoulli Naive Bayes, each feature is assumed to be independent and the probability of a class given a set of features is calculated using Bayes' theorem. Like Gaussian Naive Bayes, the parameters of the Bernoulli distribution (i.e., the probability of each feature taking on the value 1 or 0) are estimated from the training data. However, it is important to note that Bernoulli Naive Bayes may not perform well when the features are not binary or when the features are not independent. In such cases, other machine learning algorithms may be more appropriate.

$$p(x) = P[X = x] = \begin{cases} q = 1 - p & x = 0 \\ p & x = 1 \end{cases}$$

Fig. 3.32. Bernoulli Naive Bayes

iv. Logistic Regression: This supervised learning algorithm's main goal is to find the model that best illustrates the relationship between a group of predictor factors and the outcome [25].

v. Decision Tree: A Decision Tree is a machine learning algorithm that predicts the target variable by recursively splitting the data into subsets based on the input features[25]. Decision rules are applied at each node to determine the feature to split the data on, continuing until a stopping criterion is met. Decision Trees are useful for modeling complex relationships between variables and are easily interpretable and visualized.

vi. Random Forest: Random Forest is a machine learning algorithm that combines multiple decision trees to produce more accurate predictions. It is useful for both classification and regression problems, and reduces overfitting while handling high-dimensional data. By aggregating the outputs of multiple trees, Random Forest[25] improves the accuracy and robustness of the model, and is widely used in various domains.

vii. k-Nearest Neighbors: TK-Nearest Neighbor (KNN)[25] is a supervised machine learning algorithm for binary classification problems. It determines the class of a data point by computing the distance between it and its k nearest neighbors, where k is a user-defined parameter. KNN is simple, effective, and can handle non-linear boundaries and noisy data. It finds applications in diverse domains such as pattern recognition, image processing, and bioinformatics.

As there is no straightforward method to compute how a change in the hyperparameters might affect your model, hyperparameter tuning is challenging. In the majority of situations, we turn to experimentation.

Model Building in Sentiment Analysis: After conducting exploratory data analysis and preprocessing, we move on to the model building phase in our project. We observe various outcomes from the previous steps and note that the results of the different models we have trained are distinct. Our focus is primarily on the Naive Bayes model that we have used for training. As the text input has been correctly processed, it becomes a typical machine learning challenge. We anticipate the classes in the target feature using the sparse matrix, which is a matrix containing mostly zeros and a few non-zero values. This matrix represents the feature space of the text input and enables us to efficiently perform operations such as matrix multiplication and vector addition. By using the sparse matrix, we can effectively classify the text input into the desired categories or classes.

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
```

Fig. 3.33. Dataset features

Model Selection: By applying cross validation, first choose the model that performs the best. Let's run the model selection procedure while taking into account all the classification methods.

```
[ ] #creating the objects
    logreg_cv = LogisticRegression(random_state=0)
    dt_cv=DecisionTreeClassifier()
    knn_cv=KNeighborsClassifier()
    svc_cv=SVC()
    nb_cv=BernoulliNB()
    mn_cv=MultinomialNB()
    gn_cv=GaussianNB()

    cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree',2:'KNN',3:'SVC',4:'Bernoulli Naive Bayes',5:'Multinomial Naive Bayes',6:'Gaussian Naive Bayes'}
    cv_models=[logreg_cv,dt_cv,knn_cv,svc_cv,nb_cv,mn_cv,gn_cv]


    for i,model in enumerate(cv_models):
        print("{} Test Accuracy: {}".format(cv_dict[i],cross_val_score(model, X, y, cv=10, scoring ='accuracy').mean()))
```

```
Logistic Regression Test Accuracy: 0.8804211247582334
Decision Tree Test Accuracy: 0.8179521344744529
KNN Test Accuracy: 0.8716505235825689
SVC Test Accuracy: 0.8795439317757772
Bernoulli Naive Bayes Test Accuracy: 0.8029408694298767
Multinomial Naive Bayes Test Accuracy: 0.8787642046802606
Gaussian Naive Bayes Test Accuracy: nan
```

Fig. 3.34. Model selection

The findings show that logistic regression outperformed the other methods, and all of the accuracy levels are more than 80%. That's fantastic. Consequently, let's use logistic regression with hyperparameter adjustment.

2. For Sarcasm Detection

Tokenization: We created a tokenizer as seen below in Figure 3.35 and put it to use on a sizable collection of more than 25,000 sentences from the sarcasm dataset. We may inspect the word-index and the words discovered from the dataset with the aid of the tokenizer. We changed the words in the phrases into their corresponding tokens, turning them into sequences. After that, we applied post-padding to the sequences, setting it to the length of the largest word in the dataset, to make sure all sentences had the same length. If we want to examine the sequences, we may print one of them to see how the padded data is structured overall. This code initializes a tokenizer object with an out-of-vocabulary (OOV) token, fits it on a text data, and creates a word-index dictionary that maps each unique word to its numerical index. The length of this dictionary and the entire dictionary are then printed to the console. This code is useful for text preprocessing and creating a numerical representation of text for machine learning tasks such as sentiment analysis or text classification.

```
[ ]  tokenizer = Tokenizer(oov_token = "<OOV>")
     tokenizer.fit_on_texts(process_reviews['reviews'])

     word_index = tokenizer.word_index
     print(len(word_index))
     print(word_index)

     35628
     {'<OOV>': 1, 'but': 2, 'not': 3, 'guitar': 4, 'great': 5, 'one': 6,
```

Fig. 3.35. Tokenization

This code as shown in Figure 3.36 takes a list of sentences and converts them into sequences of numerical indices using the previously created tokenizer object. It then pads the sequences to a maximum length using the "pad_sequences" function from Keras. Padding ensures that all sequences have the same length, which is necessary for feeding the data into a neural network.

The resulting padded sequences are printed to the console along with their shape. This code is useful for preparing text data for use in machine learning models such as neural networks for sentiment analysis or text classification.

```
[ ]  sequences = tokenizer.texts_to_sequences(sentences)
     padded = pad_sequences(sequences, padding='post')
     print(padded[0])
     print(padded.shape)

     [   3   25 1615 ...    0    0    0]
     (10261, 1129)
```

Fig. 3.36. Padding and Sequencing

The below code declares several variables that will be used to configure the neural network for training on text data.

- "vocab_size" specifies the maximum number of words to keep based on word frequency.
- embedding_dim" is the dimension of the dense embedding space where the words will be represented as vectors.
- ""max_length" is the maximum length of the padded sequences.
- "trunc_type" specifies how to handle sequences that exceed the specified length, in this case, by truncating them at the end of the sequence.
- "padding_type" specifies where to add padding to the sequences, in this case, at the end of the sequence.
- "oov_tok" is the out-of-vocabulary token used to represent words that are not in the tokenizer's vocabulary.
- "training_size" is the number of training samples to use when training the neural network.

These variables are important for configuring the neural network architecture and preprocessing the input text data to ensure that it is in the appropriate format for training.

```
vocab_size = 10000
embedding_dim = 16
max_length = 100
trunc_type = 'post'
padding_type = 'post'
oov_tok = "<OOV>"
training_size = 8000
```

```
[ ] training_sentences = sentences[0:training_size]
    testing_sentences = sentences[training_size:]
    training_labels= labels[0:training_size]
    testing_labels = labels[training_size:]
```

Fig. 3.37. Variable Configurations

This code block preprocesses the text data for use in a neural network model for natural language processing tasks such as sentiment analysis or text classification. The text data is tokenized, which involves converting each word in the text into a numerical index using a dictionary-like structure called a tokenizer. The tokenizer is initialized with a maximum vocabulary size and an out-of-vocabulary token to handle words not in the vocabulary. The tokenizer is then fitted on the training sentences to create a word-index dictionary. The training and testing sentences are then converted into sequences of numerical indices using the word-index dictionary and padded to a fixed length using the Keras "pad_sequences" function. The resulting preprocessed data is in a suitable format for training and evaluating a neural network model. Splitting the data into training and testing sets and preprocessing the text data are important steps to ensure the model learns the underlying patterns in the data and performs well on unseen data. We can see this in Figure 3.38 below.

```
tokenizer = Tokenizer(num_words=vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)

word_index = tokenizer.word_index

training_sequences = tokenizer.texts_to_sequences(training_sentences)
training_padded = pad_sequences(training_sequences, maxlen=max_length, padding= padding_type, truncating=trunc_type)

testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(testing_sequences, maxlen=max_length, padding= padding_type, truncating=trunc_type)
```

Fig. 3.38 Training and Testing Tokenizer

F. Model Testing

1. For review classification

75% of the dataset has been used for *training* and the remaining 25% has been used for *testing* purposes as shown in Figure 3.39. In this project, a significant portion of the dataset, specifically 75%, was allocated for training the sentiment analysis model. This process involved feeding the machine with the encoded reviews and their corresponding sentiments, enabling it to learn from the patterns and relationships within the data. The remaining 25% of the dataset was reserved for testing purposes. This portion was used to evaluate the model's performance and determine its accuracy in predicting the sentiments of new, previously unseen reviews.

The results obtained from the training and testing process were analyzed in the next chapter, where we discussed the model's effectiveness in accurately predicting sentiment based on the review text. The analysis involved comparing the predicted sentiment with the actual sentiment of the reviews and calculating metrics such as precision, recall, and F1 score to evaluate the model's performance. Additionally, the analysis involved identifying any areas where the model could be improved to achieve better performance. Overall, the results of the sentiment analysis model's training and testing phases were vital in determining its effectiveness and ensuring its ability to accurately predict sentiment in new reviews. The results that are obtained have been discussed in the next chapter.

```
[ ]  print(f'Original dataset shape : {Counter(y)}')

     smote = SMOTE(random_state=42)
     X_res, y_res = smote.fit_resample(X, y)

     print(f'Resampled dataset shape {Counter(y_res)}')
```

```
Original dataset shape : Counter({2: 9022, 1: 772, 0: 467})
Resampled dataset shape Counter({2: 9022, 1: 9022, 0: 9022})
```

```
[ ]  ## Divide the dataset into Train and Test
     X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.25, random_state=0)
```

Fig. 3.39. Model Testing

2. For Sarcasm Detection

The below code defines a sequential neural network model for text classification. The model consists of an embedding layer, which maps each word in the vocabulary to a high-dimensional vector representation, a global average pooling layer, which averages the vectors for each word in the sequence, a dense layer with 24 units and a rectified linear unit (ReLU) activation function, and a final dense layer with one unit and a sigmoid activation function, which outputs a probability value between 0 and 1 for binary classification.

The model is compiled with the binary cross-entropy loss function, which is commonly used for binary classification tasks, and the Adam optimizer, which is an efficient stochastic gradient descent algorithm. The model is also configured to calculate and report the accuracy metric during training and evaluation. Once the model is defined and compiled, it is ready to be trained on the preprocessed text data.

```
[ ]  model = tf.keras.Sequential([
         tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
         tf.keras.layers.GlobalAveragePooling1D(),
         tf.keras.layers.Dense(24, activation='relu'),
         tf.keras.layers.Dense(1, activation='sigmoid')
     ])
     model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])
```

Fig. 3.40. Sequential Neural Networks

This code, as shown below in Figure 3.41 trains and evaluates the previously defined neural network model for text classification. The model is trained on the preprocessed training data for 50 epochs, with the training and validation accuracy tracked and displayed during training. After training, the model is evaluated on the preprocessed testing data using the "evaluate" method and the loss and accuracy metrics are computed. Finally, the accuracy metric is printed to the console as a percentage. This code allows us to assess the performance of

the model on unseen data and determine if the model has learned the underlying patterns in the data.

```
[ ] history = model.fit(training_padded, training_labels, epochs=50, validation_data=(testing_padded, testing_labels))

    # Evaluate the model on the testing data
    loss, accuracy = model.evaluate(testing_padded, testing_labels)

    # Print the model's accuracy
    print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Epoch 23/50
250/250 [==============================] - 2s 7ms/step - loss: 0.0977 - accuracy: 0.9689 - val_loss: 3.1255 - val_accura
Epoch 24/50
250/250 [==============================] - 2s 7ms/step - loss: 0.0898 - accuracy: 0.9712 - val_loss: 3.2575 - val_accura
Epoch 25/50
250/250 [==============================] - 2s 7ms/step - loss: 0.0846 - accuracy: 0.9724 - val_loss: 3.4939 - val_accura
Epoch 26/50
250/250 [==============================] - 2s 8ms/step - loss: 0.0765 - accuracy: 0.9771 - val_loss: 3.7304 - val_accura
Epoch 27/50
```

Fig. 3.41. Training and Evaluating Neural Networks

Chapter 04: Experiments and Results Analysis

4.1 Performance Parameters and Analysis

The following parameters have been used to assess the prediction results:

Precision: It is the measure of points that are actually positive, out of all the points in a model predicted positive.

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

Recall: It is the measure of points predicted to be positive out of those which are actually positive in a model.

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

F1 Score: The harmonic mean between Precision and Recall is coined as F1 score .

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

Accuracy: Out of the total number of points in a model, the points which are correctly classified is called accuracy.

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

Where, TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

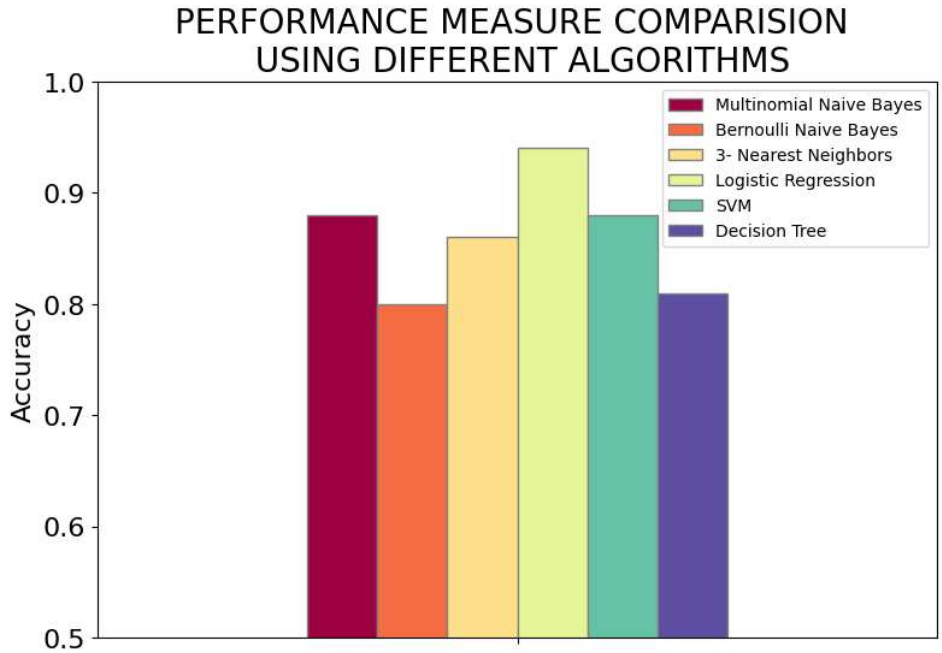


Fig. 4.1. Logistic Regression Classifier

Table II: Performance Measure Comparision

S. No.	Model Used	Precision	Recall	F-1 Score	Accuracy
	Logistic Regression	94 %	94%	94%	94%
2	K Nearest Neighbors	87 %	86 %	86 %	86 %
3	SVC	87 %	87 %	87 %	88 %
4	Bernoulli Naive Bayes	80 %	81 %	80 %	80 %
5	Multinomial Naive Bayes	85 %	85 %	86 %	86 %
6	Decision Tree	82 %	81 %	81 %	81 %

ROC-AUC Curve: The AUC (Area Under The Curve) ROC (Receiver

Operating Characteristics) curve is used to assess or depict the performance of the multi-class classification issue. It is one of the most crucial assessment criteria for assessing the effectiveness of any classification model. It is also spelled AUROC (Area Under the Receiver Operating Characteristics). The model is more accurate at classifying 0 classes as 0, and classifying 1 classes as 1, the higher the AUC. By example, the model is more effective at differentiating between individuals with the condition and those who do not have it the higher the AUC. TPR is plotted against FPR on the ROC curve, with FPR on the x-axis and TPR on the y-axis.

4.2 Discussion on the Results

In this project, we have performed Hyper Parameter Tuning to enhance the performance of our logistic regression model, which is one of the most commonly used machine learning algorithms for sentiment analysis. The Hyper Parameter Tuning process involves adjusting the model's hyperparameters to achieve better accuracy and optimize its performance. In our case, we have focused on tuning the regularization and penalty parameters, which are key hyperparameters that significantly affect the model's performance.

The Hyper Parameter Tuning process has been meticulously carried out to ensure that the model's accuracy is maximized, while also avoiding overfitting, which is a common challenge in machine learning. The results of the Hyper Parameter Tuning process are depicted in Figure 4.2, where we can observe that the accuracy of the logistic regression classifier has been increased to 94% after the tuning process.

This high accuracy rate is a significant achievement, as it ensures that our model can accurately predict the sentiment of new reviews with a high degree of certainty. Additionally, the results of the Hyper Parameter Tuning process have provided us with valuable insights into the optimal hyperparameter settings for our logistic regression model, which can be used in future sentiment analysis projects to achieve better performance. Overall, the Hyper Parameter Tuning

process has been a crucial step in enhancing the effectiveness of our logistic regression model for sentiment analysis.

```
[ ] param_grid = {'C': np.logspace(-4, 4, 50),
                  'penalty':['l1', 'l2']}
    clf = GridSearchCV(LogisticRegression(random_state=0), param_grid,cv=5, verbose=0,n_jobs=-1)
    best_model = clf.fit(X_train,y_train)
    print(best_model.best_estimator_)
    print("The mean accuracy of the model is:",best_model.score(X_test,y_test))

    LogisticRegression(C=10000.0, random_state=0)
    The mean accuracy of the model is: 0.9379340919166543

[ ] logreg = LogisticRegression(C=10000.0, random_state=0)
    logreg.fit(X_train, y_train)
    y_pred = logreg.predict(X_test)
    print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))

    Accuracy of logistic regression classifier on test set: 0.94
```

Fig. 4.2. Logistic Regression Classifier

```
[ ] cm = metrics.confusion_matrix(y_test, y_pred)
    plot_confusion_matrix(cm, classes=['Negative','Neutral','Positive'])
```

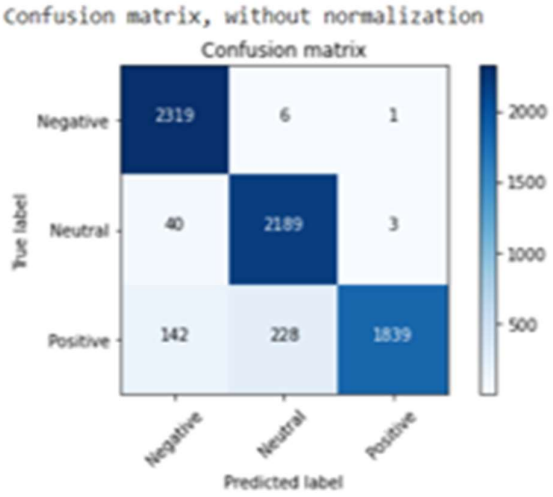


Fig. 4.3. Confusion Matrix with ROC

Here we plot a confusion matrix with ROC and check our f1-score that has been derived from Precision and Recall. As shown in Figure 4.3.

Confusion Matrix is visible between True label and Predicted label. The diagonal elements which are darker are correctly predicted records and the rest are incorrectly classified which can also be visualized.

We are taking this into consideration all three positive, negative and neutral reviews because it is crucial to forecast both favorable and negative evaluations. We received a decent f1 score. As can be seen in Figure 4.4, it received high scores in all categories of even recall precision and accuracy.

```
[ ]  print("Classification Report:\n",classification_report(y_test, y_pred))

     Classification Report:
                   precision    recall  f1-score   support

                0       0.93      1.00      0.96      2326
                1       0.90      0.98      0.94      2232
                2       1.00      0.83      0.91      2209

         accuracy                           0.94      6767
        macro avg       0.94      0.94      0.94      6767
     weighted avg       0.94      0.94      0.94      6767
```

Fig. 4.4. Performance outcome

The ROC-AUC curve plays a significant role in evaluating the performance of a classification model by showing the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) for different classification thresholds. To determine the optimal classification threshold based on specific objectives, we plotted the ROC curves for each class. Additionally, we generated the micro and macro averages of the ROC curves to gain a better understanding of which class was classified more accurately. By analyzing these curves, we can make more informed decisions about the performance of our classification model.

```
#Binarizing the target feature
y = label_binarize(y, classes=[0, 1, 2])
n_classes = y.shape[1]

#Train-Test split(80:20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2,
                                                    random_state=0)

#OneVsRestClassifier
classifier = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True,
                                 random_state=10))
y_score = classifier.fit(X_train, y_train).decision_function(X_test)

#Computing TPR and FPR
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

# aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
```

Fig. 4.5. ROC Curve code screenshot Part 1

```
# interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])

# Finally average it and compute AUC
mean_tpr /= n_classes

fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])

# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)

plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)

colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=4,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))
```

Fig. 4.6. ROC Curve code screenshot Part 2

```
[ ]  plt.plot([0, 1], [0, 1], 'k--', lw=4)
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Receiver operating characteristic to multi-class')
     plt.legend(loc="lower right")
     plt.show()
```

Fig. 4.7. ROC Curve code screenshot Part 3

The ROC Curve displayed in Figure 4.8 provides us with valuable insights into the classification performance of the model. We can observe that the classes 0 and 2 have been accurately classified due to their high area under the curve. To achieve the best true positive rate (TPR) and false positive rate (FPR), we can choose any threshold value between 0.6 and 0.8. This threshold selection depends on the objective standards set for the classification task.

Furthermore, it is apparent that the macro average doesn't perform as well as the micro average. The micro-average, which gives equal weightage to each sample, has shown better scores than the macro-average. The latter average is computed by averaging the scores of all classes equally. This implies that the model is better at identifying the minority classes than the majority class. Overall, the ROC curve provides us with a clear understanding of the model's classification performance, enabling us to make informed decisions about selecting the optimal threshold value for a given task.

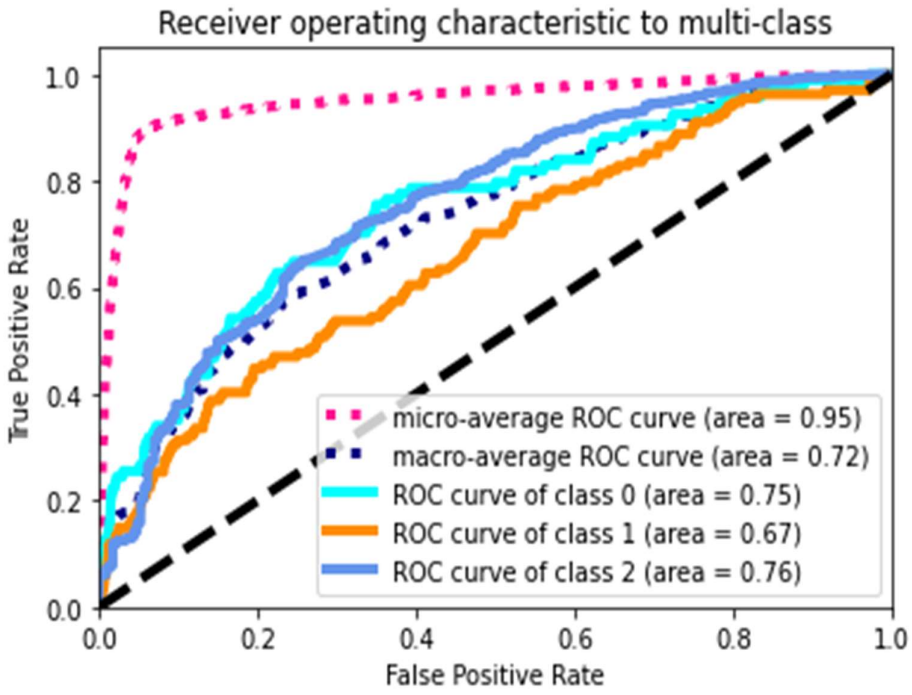


Fig. 4.8. ROC Curve

Chapter 05: Conclusion

5.1 Conclusion

Sentiment analysis is a popular technique for extracting insights from text data in eCommerce platforms, including comments, reviews, feedback, and tweets. The use of emoticons, ratings, and reviews helps to convey user opinions, and this information can be used by customers to make informed purchasing decisions.

In this project, we have used supervised machine learning methods such as Logistic Regression, Naive Bayes, and Random Forest to classify Amazon musical instruments into different categories based on the sentiment expressed in the reviews. We have also taken into account n-grams since one word alone may not provide accurate results, and stop words have been manually examined to ensure that negative terms are not included.

We found that balancing the dataset was beneficial in improving the performance of our sentiment analysis model, and the F1-score was used as the primary performance metric, which was found to be 94%. Our analysis also revealed that most of the neutral reviews were genuine complaints from customers, which can be used by Amazon to provide feedback to the vendors and help them improve their products.

Overall, sentiment analysis is a valuable technique that can help businesses improve their products by understanding customer opinions and feedback. The use of machine learning algorithms to classify sentiment in text data can help eCommerce platforms make better decisions and provide more accurate recommendations to customers.

5.2 Applications

Sentiment analysis of Amazon product reviews provides numerous advantages, as it allows both customers and sellers to gain valuable insights about the

product. Customers can easily identify the overall rating or impact of the product based on the reviews, while sellers can analyze the response generated by the reviews of their product. In the case of musical instruments, sentiment analysis helps to understand how users feel about a particular product, and what aspects they like or dislike.

In today's world, where everyone seeks out other people's opinions in order to learn from their experiences, the importance of reviews has increased significantly. However, customers hardly ever read all of the reviews available, and sentiment analysis plays a crucial role in interpreting them. By analyzing the sentiments expressed in reviews, customers can make informed decisions about their purchases, based on the opinions of others.

Furthermore, businesses can benefit greatly from sentiment analysis by learning about the preferences and requirements of consumers. By understanding what customers like or dislike about their products, businesses can improve their products and offer better customer service. Overall, sentiment analysis of Amazon product reviews is a valuable tool for both customers and businesses alike, helping to improve the shopping experience for everyone involved.

5.3 Future Scope and Limitations

In our upcoming work as part of future scope , we intend to use word2vec to extract features from our models and identify bogus reviews. We also plan on adding more layers and further performing hyper-parameter tuning on our Naive Bayes Model as it is considered a benchmark model for textual data analysis and further improved its results. A hybrid and multiple supervised learning approach could also be used to construct this system. Sarcasm detection would be a key goal of ours to further expand the scope of this project and also try and determine reviews with a hidden meaning. In addition, different classifiers besides those already stated could be employed, such as Gated Recurrent Unit (GRU) and Support Vector Machine. Additionally, since the

dataset focuses on Amazon mobile evaluations, it may also be used to analyze Amazon reviews in general. Due to the restricted resources of my personal laptop, the study's constraint is the implementation in Google Colab to speed up implementation.

References

[1] AlQahtani, Arwa SM. "Product sentiment analysis for amazon reviews." International Journal of Computer Science & Information Technology (IJCSIT) Vol 13 (2021).

[2] U. Norinder and P. Norinder, “Predicting Amazon customer reviews with deep confidence using deep learning and conformal prediction”, Journal of Management Analytics, Vol. 9, No. 1, 1–16, 2022

[3] S. Wassana , X. Chenb , T. Shenc, M. Waqard , N. Jhanjhie, “Amazon Product Sentiment Analysis using Machine Learning Techniques”, Revista Argentina de Clínica Psicológica, Vol. XXX, 2021

[4] K. S. Kumar, J. Desai and J. Majumdar, “Opinion mining and sentiment analysis on online customer review,” in IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, 2016.

[5] S. Dey, S. Wasif1 and S. Sultana, "A Comparative Study of SVM and Naive Bayes Classifier for Sentiment Analysis on Amazon Product Reviews”, International Conference on Contemporary Computing and Applications 2020.

[6] R. Jansher, “Sentimental Analysis of Amazon Product Reviews Using Machine Learning Approach”, Proceedings of the Computational Methods in Systems and Software, 2020.

[7] K. S. Kumar, J. Desai and J. Majumdar, “Opinion mining and sentiment analysis on online customer review,” in IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), Chennai, 2016.

[8] M. Rodrigo, J. F. Valiati, and W. P. GaviãO Neto. "Document-level sentiment classification: An empirical comparison between SVM and ANN." Expert Systems with Applications 40.2 (2013): 621-633

[9] Z. Dongwen, H. Xu, Z. Su, and Y. Xu. "Chinese comments sentiment classification based on word2vec and SVMperf." Expert Systems with Applications 42, no. 4 (2015): 1857-1863.

[10] Rathor, A. Singh, A. Agarwal, and P. Dimri. "Comparative study of machine learning approaches for Amazon reviews." Procedia computer science 132 (2018): 1552-1561.

[11] S. Yoichi, and V. Klyuev. "Classifying user reviews at sentence and review levels utilizing Naïve Bayes." 2019 21st International Conference on Advanced Communication Technology (ICACT). IEEE, 2019.

[12] B. Bansal, and S. Srivastava. "Sentiment classification of online consumer reviews using word vector representations." Procedia computer science 132 (2018): 1147-1153.

[13] A. S. Ashour, and N. S. Alghamdi. "A comparison of sentiment analysis methods on Amazon reviews of Mobile Phones." International Journal of Advanced Computer Science and Applications 10.6 (2019).

[15] A. Cernian, S. Valentin, and M. Bogdan. "Sentiment analysis from product reviews using SentiWordNet as lexical resource." 2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). IEEE, 2015.

[16] https://www.kaggle.com/datasets/eswarchandt/amazon-music-reviews

[17] https://github.com/BenRoshan100/Sentiment-analysis-Amazon-reviews

[18] O. Tsur, A. Rappoport, and D. Davidov, "Semi-supervised recognition of sarcastic sentences in twitter and amazon," in Proc. Fourteenth Conference on Computational Natural Language Learning, pp. 107–116.

[19] B. Faltings, M. Boia, C.-C. Musat, and P. Pu, "A:) Is worth a thousand words: How people attach sentiment to emoticons and words in tweets," in 2013 International Conference on Social Computing, pp. 345–350, 2013.

[20] K. V. Indukuri, K. Manuel, and P. R. Krishna, "Analyzing internet slang for sentiment mining," in 2010 Second Vaagdevi International Conference on Information Technology for Real World Problems, pp. 9–11, 2010.

[21] M. J. Carman, A. Joshi, and P. Bhattacharyya, "Automatic sarcasm detection: A survey," 2016 [Online]. Available: https://arxiv.org/abs/ 1602.03426.

[22] L. Lillian, V. Shivakumar, and B. Pang, "Thumbs up?: Sentiment classification using machine learning techniques," in Proc. ACL Conference Empirical Methods Natural Language Processing, vol. 10, pp. 79–86, 2002.

[23] D. Maynard and M. A. Greenwood, "Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis," in Proceedings. 9th Int. Conference. Language Resources and Evaluation, pp. 4238–4243, 2014.

[24] T. Finin and J. Marineau, "Delta TFIDF: An improved feature space for sentiment analysis," in Proc. AAAI International Conference on Weblogs and Social Media, 2009.

[25] A. Joshi, V. Rana, A.Sharma, "Brain Tumor Classification using Machine Learning and Deep Learning Algorithms: A Comparison: Classifying brain MRI images on thebasis of location of tumor and comparing the various Machine Learning and Deep LEARNING models used to predict best performance," InProceedings of the 2022 Fourteenth International Conference on Contemporary Computing 2022 Aug 4 (pp. 15-21).

[26] E. Lunando and A. Purwarianti, “Indonesian Social Media Sentiment Analysis With Sarcasm Detection,” In: Int. Conf. Adv. Comput. Sci. Inf. Syst. ICACSIS, pp. 195-198. September (2013).

[27] D. Davido, O. Tsura and A. Rappoport, “Semi-Supervised Recognition of Sarcasm in Twitter and Amazon,” In: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, pp.107–116, July (2010)

[28] R. Ahuja, A. Chug, S. Kohli, S. Gupta, P. Ahuja, “The Impact of Features Extraction on the Sentiment Analysis,” In: Procedia Computer Science, vol. 152, pp. 341-348. Elsevier, January (2019).

[29] P. Mehndiratta, S. Sachdeva and D. Soni, “Detection of Sarcasm in Text Data using Deep Convolutional Neural Networks,” In: Scientific International Journal for Parallel and Distributed Computing, vol. 18, September (2017).

[30] P. Mandal and R. Mahto, “Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection” (2019).

[31] A. Agarwal., B. Xie, I. Vovsha, O. Rambow, and R. Passonneau,”Sentiment Analysis of Twitter Data,” Proceedings of the Workshop on Language in Social Media, pp. 30-38, June (2011).

[32] D. Das and A. Clark, “Sarcasm detection on flickr using a cnn”, in International Conference on Computing and Big Data, (2018).

[33] G. Angulakshmi and R. Manicka Chezian, “Three level feature extraction for sentiment classification.” In: International Journal of Innovative Research in Computer and Communication Engineering 2, vol. 8,pp. 5501-5507 (2014).

[34] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “Pre- training of deep bidirectional transformers for language understanding”, (2018).

[35] R.N. Waykole, A.D. Thakare, “A Review Of Feature Extraction Methods For Text Classification.” ,International Journal of Advance Engineering and Research Development,vol. 5, April (2018).

[36] O. Chudi-Iwueze, and H. Afli. "Detecting Sarcasm in News Headlines." CERC. 2020.

Appendix

```
[ ]  import pandas as pd
     import requests
     from bs4 import BeautifulSoup
```

```
[ ]  search_query="nike+shoes+men"
```

```
[ ]  base_url="https://www.amazon.com/s?k="
     url=base_url+search_query
```

```
[ ]  url

     'https://www.amazon.com/s?k=nike+shoes+men'
```

Fig. 7.1. Importing libraries for web scraping

```
header={'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36','
referer':'https://www.amazon.com/s?k=nike+shoes+men&crid=28WRS5SFLWWZ6&sprefix=
```

```
[ ]  search_response=requests.get(url,headers=header)
```

```
     search_response.status_code

     200
```

```
[ ]  search_response.text
```

Fig. 7.2 Defining parameters for data extraction

```
[ ] search_response=requests.get(url,headers=header)

[ ] search_response.status_code

    200

[ ] search_response.text

    '<!doctype html><html lang="en-us" class="a-no-js" data-19ax5a9jf="dingo"><!-- sp:feature:
    ="utf-8"/>\n<!-- sp:end-feature:head-start -->\n<!-- sp:feature:csm:head-open-part1 -->\n\
    e:csm:head-open-part1 -->\n<!-- sp:feature:cs-optimization -->\n<meta http-equiv=\'x-dns-p
    s-amazon.com">\n<link rel="dns-prefetch" href="https://m.media-amazon.com">\n<link rel="dn
    <!-- sp:feature:csm:head-open-part2 -->\n<script type=\'text/javascript\'>\nwindow.ue_ihb
    ow,\n    ue_hob = +new Date();\n(function(d){var e=d.ue=d.ue||{},f=Date.now||function(){r.

[ ] search_response.cookies

    <RequestsCookieJar[Cookie(version=0, name='i18n-prefs', value='USD', port=None, port_speci
    path_specified=True, secure=False, expires=1695831627, discard=False, comment=None, commen
    6047735', port=None, port_specified=False, domain='.amazon.com', domain_specified=True, dc
    discard=False, comment=None, comment_url=None, rest={}, rfc2109=False), Cookie(version=0,
    domain='.amazon.com', domain_specified=True, domain_initial_dot=True, path='/', path_speci
    rest={}, rfc2109=False)]>

[ ] cookie={} #insert request cookies within{}
```

Fig. 7.3. Response of the data request

```
[ ]  cookie={} #insert request cookies within{}
```

```
[ ]  def getAmazonSearch(search_query):
       url="https://www.amazon.com/s?k="+search_query
       print(url)
       page=requests.get(url,headers=header)
       if page.status_code==200:
         return page
       else:
         return "Error"
```

```
[ ]  def Searchasin(asin):
       url="https://www.amazon.com/dp/"+asin
       print(url)
       page=requests.get(url,cookies=cookie,headers=header)
       if page.status_code==200:
         return page
       else:
         return "Error"
```

```
     def Searchreviews(review_link):
       url="https://www.amazon.com"+review_link
       print(url)
       page=requests.get(url,cookies=cookie,headers=header)
       if page.status_code==200:
         return page
       else:
         return "Error"
```

Fig. 7.4. Setting definite path for each product request

```
[ ] len(product_names)
```

```
66
```

```
[ ] data_asin=[]
    response=getAmazonSearch('nike+shoes+men')
    soup=BeautifulSoup(response.content)
    for a in soup.findAll("div",{'data-component-type':"s-search-result"}):
      data_asin.append(a['data-asin'])
```

https://www.amazon.com/s?k=nike+shoes+men

```
[ ] response.status_code
```

```
200
```

```
data_asin
```

```
['B07NMZ6S6Z',
 'B093Y7Z61R',
 'B004IM1GHW',
 'B07ZJP7LWP',
 'B00XWNW11I',
 'B0047XCBJE',
 'B07RLYXS26',
 'B06XS8BVXH',
 'B093C3FF2B',
 'B0B558DGGN',
 'B00Q6ZBZDY',
```

```
[ ] len(data_asin)
```

```
48
```

```
link=[]
for i in range(len(data_asin)):
  response=Searchasin(data_asin[i])
  soup=BeautifulSoup(response.content)
  for b in soup.findAll("a",{'data-hook':"see-all-reviews-link-foot"}):
    link.append(b['href'])
```

```
https://www.amazon.com/dp/B07NMZ6S6Z
https://www.amazon.com/dp/B093Y7Z61R
https://www.amazon.com/dp/B004IM1GHW
https://www.amazon.com/dp/B07ZJP7LWP
https://www.amazon.com/dp/B00XWNW11I
https://www.amazon.com/dp/B0047XCBJE
https://www.amazon.com/dp/B07RLYXS26
https://www.amazon.com/dp/B06XS8BVXH
https://www.amazon.com/dp/B093C3FF2B
https://www.amazon.com/dp/B0B558DGGN
https://www.amazon.com/dp/B00Q6ZBZDY
https://www.amazon.com/dp/B078P4Q6MC
https://www.amazon.com/dp/B071K7818X
https://www.amazon.com/dp/B07RFNMHL5
https://www.amazon.com/dp/B005AHBLSS
https://www.amazon.com/dp/B08GJ3FWXW
https://www.amazon.com/dp/B07SKKSVQM
https://www.amazon.com/dp/B08QBTY9BG
https://www.amazon.com/dp/B07ZTVN8DS
https://www.amazon.com/dp/B07T7HZZKJ
```

Fig. 7.5. Generating product page link using html tags on amazon webpage

```
[ ] len(link)
```

```
92
```

```
for j in range(len(link)):
  print(link[j])
```

```
/Nike-AQ1189-Mens-Fitness-Shoes/product-reviews/B07VQGQVX5/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-AQ1189-Mens-Fitness-Shoes/product-reviews/B07VQGQVX5/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-AA2148-Mens-Cross-Trainers/product-reviews/B07NJLLSDG/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-AA2148-Mens-Cross-Trainers/product-reviews/B07NJLLSDG/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-AH8050-Mens-Gymnastics-Shoes/product-reviews/B07FCQGVJF/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-AH8050-Mens-Gymnastics-Shoes/product-reviews/B07FCQGVJF/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Running-Shoes-Sneaker-537732/product-reviews/B09NKLBR8N/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Court-Legacy-Gymnastics/product-reviews/B0947JKF14/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Court-Legacy-Gymnastics/product-reviews/B0947JKF14/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Track-Field-Running-Orange/product-reviews/B07TPYH8TL/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Track-Field-Running-Orange/product-reviews/B07TPYH8TL/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-CQ9380-001-Low-top-Sneakers/product-reviews/B08NY4R4P4/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-CQ9380-001-Low-top-Sneakers/product-reviews/B08NY4R4P4/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Killshot-Leather-Sneaker/product-reviews/B086H3GG86/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Killshot-Leather-Sneaker/product-reviews/B086H3GG86/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Basketball-Shoes-White/product-reviews/B07CYWJNNG/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Basketball-Shoes-White/product-reviews/B07CYWJNNG/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Air-Max-Mens-Sneakers/product-reviews/B096DJ4CVY/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Air-Max-Mens-Sneakers/product-reviews/B096DJ4CVY/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Revolution-Wide-Running/product-reviews/B085LTY56B/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Revolution-Wide-Running/product-reviews/B085LTY56B/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-CD5463-Mens-Sneaker/product-reviews/B0BFZRS4LD/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-CD5463-Mens-Sneaker/product-reviews/B0BFZRS4LD/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-415445-Mens-Sneaker/product-reviews/B0BG1GGCH5/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
```

```
[ ] mylist = list( dict.fromkeys(link) )
    print(mylist)

['/Nike-Mens-Court-Vision-Sneaker/product-reviews/B07NMZ5Q5N/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews', '/Nike-Mens-Monarch-Cr

[ ] mylist = list( dict.fromkeys(mylist) )
    print(mylist)

['/Nike-Mens-Court-Vision-Sneaker/product-reviews/B07NMZ5Q5N/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews', '/Nike-Mens-Monarch-Cr

    for j in range(len(mylist)):
      print(mylist[j])

/Nike-Mens-Court-Vision-Sneaker/product-reviews/B07NMZ5Q5N/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Monarch-Cross-Trainer/product-reviews/B07JQKM2SP/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-BQ9646-Mens-Running-Shoe/product-reviews/B085WQZCT5/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Sneakers-Breathable-Comfortable-Lightweight-Cushioning/product-reviews/B078H9Z1S5/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/NIKE-Mens-Free-Flyknit-Running/product-reviews/B07BFMNZ7K/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Control-Cross-Trainer/product-reviews/B07RDMBS9J/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/NIKE-Mens-2017-Running-Shoe/product-reviews/B01IB5JATM/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Precision-Basketball-Anthracite-CW3403-006/product-reviews/B096HXT1F3/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Force-07-Basketball/product-reviews/B0829L4VC6/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-684494-600-Mens-Lace-up/product-reviews/B0B4VB5Y7B/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Vapormax-Plus-Mens-Style/product-reviews/B079H3CZFG/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/NIKE-Mens-2018-Running-Shoe/product-reviews/B000M5CRC2/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Sneaker/product-reviews/B0BG141WW4/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-Mens-Sneaker-Running-Shoes/product-reviews/B07C5DJ6LC/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-CQ9246-103-Mens-Tennis-Shoe/product-reviews/B08R4RJTD9/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
/Nike-AR3762-Mens-Fitness-Shoes/product-reviews/B07VM1DW74/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews
```

Fig. 7.6. Generating the Review page link for all products

```
reviewslist=[]
for j in range(len(mylist)):
  response=Searchreviews(mylist[j]+'&pageNumber='+str(0))
  soup=BeautifulSoup(response.content)
  review={
      'productTitle': soup.title.text.replace("Amazon.com: Customer reviews: ", "")
      .replace("&#39;", "'").strip(),
      'Customer Name'  :soup.find("span",{'class':"a-profile-name"}).text,
      'Date'  :soup.find("span",{'data-hook':"review-date"}).text.partition('on')[2],
      'Review Title': soup.find('a', {'data-hook':"review-title"}).text.strip(),
      'Rating': soup.find('i', {'data-hook': 'review-star-rating'}).text.strip()[0],
      'Review Body': soup.find("span",{'data-hook':"review-body"}).text.strip()
  }
  reviewslist.append(review)
```

```
https://www.amazon.com/Nike-Mens-Court-Vision-Sneaker/product-reviews/B07NMZ5Q5N/ref=cm_cr_dp_
https://www.amazon.com/Nike-Mens-Monarch-Cross-Trainer/product-reviews/B07JQKM2SP/ref=cm_cr_dp
https://www.amazon.com/Nike-BQ9646-Mens-Running-Shoe/product-reviews/B085WQZCT5/ref=cm_cr_dp_d
https://www.amazon.com/Sneakers-Breathable-Comfortable-Lightweight-Cushioning/product-reviews/
https://www.amazon.com/NIKE-Mens-Free-Flyknit-Running/product-reviews/B07BFMNZ7K/ref=cm_cr_dp_
https://www.amazon.com/Nike-Mens-Control-Cross-Trainer/product-reviews/B07RDMBS9J/ref=cm_cr_dp
https://www.amazon.com/NIKE-Mens-2017-Running-Shoe/product-reviews/B01IB5JATM/ref=cm_cr_dp_d_s
https://www.amazon.com/Nike-Precision-Basketball-Anthracite-CW3403-006/product-reviews/B096HXT
https://www.amazon.com/Nike-Mens-Force-07-Basketball/product-reviews/B0829L4VC6/ref=cm_cr_dp_d
https://www.amazon.com/Nike-684494-600-Mens-Lace-up/product-reviews/B0B4VB5Y7B/ref=cm_cr_dp_d_
https://www.amazon.com/Nike-Vapormax-Plus-Mens-Style/product-reviews/B079H3CZFG/ref=cm_cr_dp_d
https://www.amazon.com/NIKE-Mens-2018-Running-Shoe/product-reviews/B000M5CRC2/ref=cm_cr_dp_d_s
https://www.amazon.com/Nike-Mens-Sneaker/product-reviews/B0BG141WW4/ref=cm_cr_dp_d_show_all_bt
https://www.amazon.com/Nike-Mens-Sneaker-Running-Shoes/product-reviews/B07C5DJ6LC/ref=cm_cr_dp
https://www.amazon.com/Nike-CQ9246-103-Mens-Tennis-Shoe/product-reviews/B08R4RJTD9/ref=cm_cr_d
```

Fig. 7.7. Appending the extracted information into a list

```
df = pd.DataFrame(reviewslist)
df
```

	productTitle	Customer Name	Date	Review Title	Rating
0	Nike Men's Court Vision Low Sneaker	Cathy	September 24, 2022	Better than Af-1	5
1	Nike Men's Air Monarch IV Cross Trainer	pcrews	July 27, 2022	An accurate 4E	4
2	Nike Men's Running Shoe	Lifelong student	September 18, 2022	Been a while since I've worn a really good run...	5
3	NIKE Men's Tanjun Sneakers, Breathable Textile...	eddy munoz	September 16, 2022	I got a different version of the shoes.	5
4	Nike Men's Free RN Flyknit Running Shoe	Katy Dicky	July 19, 2022	Good shoes for distance	4
5	Nike Men's Flex Control Tr4 Cross Trainer	Franklbcav	September 14, 2022	Decent for the price	4
6	Nike Men's Air Max 2017 Cool Grey/Anthracite-D...	Daniel W.	September 15, 2022	Bubble popped after a few months. Right shoe i...	4
7	Nike Precision 5 Men's Basketball Shoes Black ...	Jeffrey SASEK	September 13, 2022	Better than expected! Super Happy with these	5
8	Nike Men's Air Force 1 '07 An20 Basketball Shoe	Patrick Jordan	September 6, 2022	Current Favorite AF1: the original	5
9	Nike Men's Lace-up	Michæl Grubb	August 17, 2022	Comfortable	5
10	Nike Air Vapormax Plus	amir	September 3, 2022	It isnot original the insole doesn't stay in p...	4
11	Nike Mens Free Rn 2018 Running Shoe	JARED	August 31, 2022	Great shoe	5
12	Nike Men's Sneaker	MICHAEL MCCAUSLIN	September 10, 2022	Great comfortable	5
13	Nike Men's Sneaker,Running Shoes	Kris Boyce	September 23, 2022	Great Product !!	5
14	Nike Men's Tennis Shoe	Mrsbrmr	August 28, 2022	Decent fit, overall satisfied	4

Fig. 7.8. Sample data after web scraping