

Sentiment Analysis using Machine Learning on Amazon Datasets

By Dipawoli Malla

Table of Contents

1. Introduction	1
1.1. Topic and Concept Visualization.....	1
1.2. Problem Overview.....	2
1.3. Importance of Sentiment Analysis.....	2
1.4. Aims and Objectives	2
1.5. Motivation.....	3
1.6. The relevance of AI and the Dataset.....	3
1.7. Description of the selected domain	4
2. Background	5
2.1. Research on the problem scenario	5
2.1.1. Research Paper 1	5
2.1.2. Research Paper 2	5
2.1.3. Research Paper 3	6
2.2. Algorithms and Evaluation Metrics Review	6
2.2.1. Research Paper 1	6
2.2.2. Research Paper 2	7
2.2.3. Research Paper 3	8
2.2.4. Comparative Analysis	9
2.2.5. Comparison of evaluation metrics.....	10
2.3. Block diagram	11
2.4. Advantages of working around the problem domain	11
2.5. Considered drawbacks revolving the problem domain.....	12
2.6. Research on dataset.....	12
2.6.1. Dataset description	12
2.6.2. Dataset analysis	15
3. Solution	17
3.1. Proposed solution to the problem	17
3.2. Explanation of the approach	17
3.3. Explanation of AI Algorithms	19
3.3.1. Naive Bayes Classifier	19

3.3.2.	Support Vector Machines (SVM)	19
3.3.3.	Logistic Regression	20
3.3.4.	K-Nearest Neighbors (KNN)	21
3.4.	Explanation of Evaluation Metrics	21
3.4.1.	Confusion Matrix	22
3.4.2.	Precision	22
3.4.3.	Recall.....	22
3.4.4.	Accuracy	23
3.4.5.	AUC-ROC curve	23
3.4.1.	Macro Avg.....	23
3.4.2.	Weighted Avg	23
3.5.	Pseudocode of algorithms.....	24
3.5.1.	Pseudocode of Naïve Bayes Classifier	24
3.5.2.	Pseudocode of SVM model	25
3.5.3.	Pseudocode of Logistic Regression Model	26
3.6.	Diagrammatical representation	27
3.6.1.	Flowchart of Naïve Bayes Classifier	27
3.6.2.	Flowchart of SVM	28
3.6.3.	Flowchart of Logistic Regression	29
4.	Development.....	30
4.1.	Development Tools	30
4.1.1.	Environment.....	30
4.1.2.	Programming Language	30
4.1.3.	Libraries	30
4.2.	Model development.....	31
4.2.1.	Data Preprocessing	32
4.2.2.	Data Splitting	35
4.2.3.	Vectorization	36
4.2.4.	Model Training	37
5.	Results.....	39
5.1.	Achieved Model Results	39
5.1.1.	Classification Report and Confusion Matrix	39

5.1.2.	Precision Comparison Table of Balanced data	42
5.1.3.	Recall Comparison Table.....	42
5.1.4.	F1-Score Comparison Table.....	43
5.1.5.	Accuracy Comparison Table.....	44
5.1.6.	Model Comparison.....	44
5.1.7.	ROC-AUC Curve of Balanced Dataset	45
5.1.8.	Comparison Table of unbalanced dataset	47
6.	Conclusion	48
6.1.	Analysis of the work done	48
6.2.	How the solution addresses real world problems.....	48
6.3.	Further work.....	48
7.	References.....	49

Tables of Tables

Table 1: Metrics comparison of research papers	10
Table 2: Dataset description table	13
Table 3: Count Value of Sentiments.....	15
Table 4: Precision Comparison Table	42
Table 5: Recall Comparison table of balanced dataset	43
Table 6: F1 score Comparison table of models in balanced dataset.....	43
Table 7: Accuracy Comparison Table	44

Tables of Figures

Figure 1: Types of sentiment analysis method (Wankhade, et al., 2022).....	1
Figure 2: Visualization of the machine learning approach (Taboada , 2016).....	1
Figure 3 : Image on Sentiment Analysis on customer data (Sedik, 2023).....	2
Figure 4: Block Diagram of the model development.....	11
Figure 5: Amazon Dataset snippet	12
Figure 6: Word cloud of positive reviews.....	14
Figure 7: Word cloud of negative reviews	15
Figure 8: Sentiment Distribution figure	16
Figure 9: Machine Learning Model Development cycle.....	18
Figure 10: Figure of how SVM works (Vasista Reddy, 2018).....	20
Figure 11: Evaluation metrics for classification (Anon., 2018).....	22
Figure 12: Flowchart of Naïve Bayes Classifier.....	27
Figure 13: Flowchart of SVM.....	28
Figure 14: Flowchart of Logistic Regression	29
Figure 15: Logo of Jupyter Notebook	30
Figure 16: Logo of Python	30
Figure 17: Oversampling data.....	32
Figure 18: Converting text to lowercase	33
Figure 19: Removing stopwords, special character using NLTK	33
Figure 20: Using Lemmantization to the text	34
Figure 21: Label encoded for sentiments	34
Figure 22: Data splitting	35
Figure 23: Model training of Naive Bayes.....	37
Figure 24: Model training of Logistic Regression	37
Figure 25: Linear SVC model training	38
Figure 26: Classification report of Logistic Regression with TFIDF	39
Figure 27: Confusion Matrix of Logistic Regression	39
Figure 28: SVM classification report.....	40
Figure 29: Confusion matrix of SVM.....	41
Figure 30: Classification report of Naive Bayes with TFIDF	41
Figure 31: Confusion matrix of Naive Bayes	42
Figure 32: ROC-AUC curve of Naive Bayes.....	45
Figure 33: ROC-AUC curve of Logistic Regression	46
Figure 34: ROC-AUC curve of SVC	46

1. Introduction

1.1. Topic and Concept Visualization

Sentiment Analysis is the process of identifying, classifying and analysing large amount of text into the emotional tone of the message i.e. positive, negative or neutral. (ibm, 2024) Sentiment analysis is a subset of natural language languages (NLP) method that helps to identify the emotions in text. There are various approaches for analysing sentiments from the data.

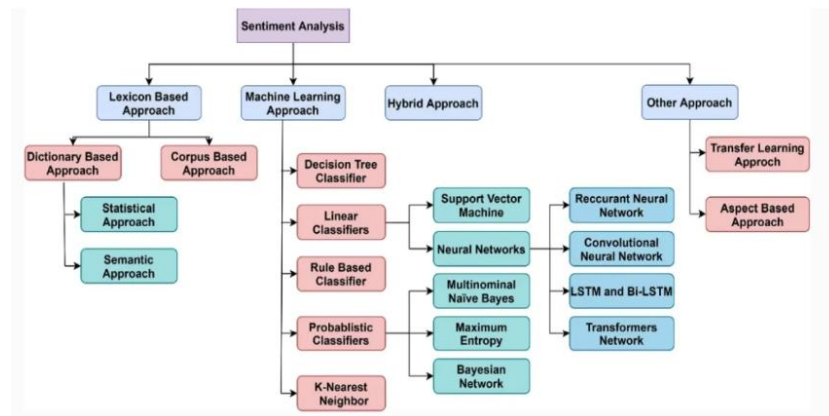


Figure 1: Types of sentiment analysis method (Wankhade, et al., 2022)

In this coursework, machine learning approach is used to analyse a corpus for sentiment analysis using machine learning models to train and test on the corpus.

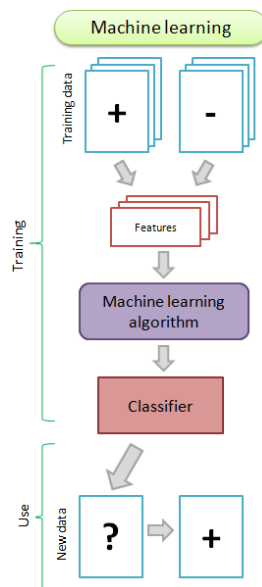


Figure 2: Visualization of the machine learning approach (Taboada , 2016)

1.2. Problem Overview

In today's world where everyone is shopping on digital ecommerce platforms like Amazon, large amount of customer data and reviews are generated. These customer reviews are very important for e-commerce giants like Amazon to understand their customer satisfaction and sentiment better. However, manually analysing such large volumes of text data can be challenging and inefficient for the companies. It can lead to favouritism, biasness and other operational costs which is why this process requires automation. In such cases, performing sentiment analysis can be very useful and effective.

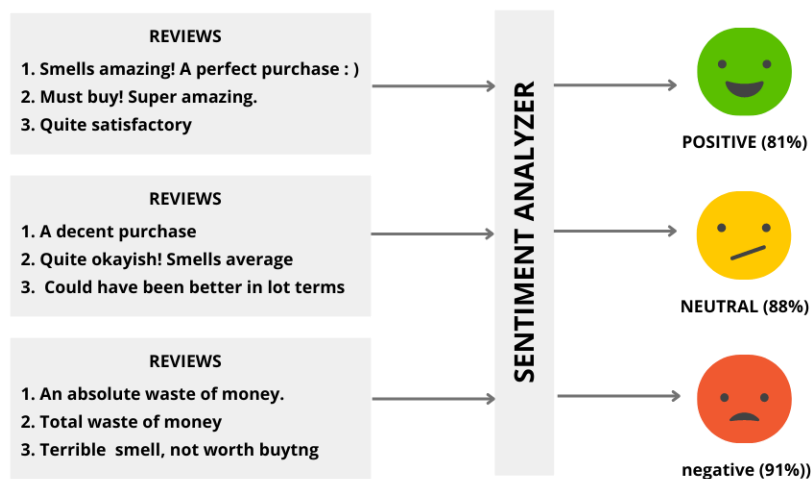


Figure 3 : Image on Sentiment Analysis on customer data (Sedik, 2023)

1.3. Importance of Sentiment Analysis

Performing sentiment analysis can be very beneficial to the business. It allows them to see what type of experience a customer is having with their products, in their platforms, websites either positive, negative or neutral. (Rosemin Anderson, 2024) It is a great indication for them to learn about their strengths and weaknesses in their services or experiences for them to take action with accurate sentiment analysis. It also helps in understanding customer emotions and customer in businesses like Amazon, Flipkart to curate a more personalized experience for them. (ibm, 2024)

1.4. Aims and Objectives

The aim of this project is to build and analyse sentiments in the product review dataset using machine learning models.

The objective of this project are as follows:

- To research and explore the dataset of product reviews of beauty products.
- To transform textual reviews into numerical features using techniques.
- To experiment with a variety of machine learning algorithms for sentiment classification.
- To train these models on the pre-processed dataset by optimizing model parameters through different techniques to enhance performance.
- To assess model performance using evaluation metrics.
- To document the development process.

1.5. Motivation

The motivation behind this project stems from several key considerations, deeply rooted in both academic curiosity and practical application such as

- To gain insights into consumer preferences and satisfaction through sentiment analysis.
- To apply and test the NLP technologies in sentiment analysis.
- To help businesses respond swiftly to consumer feedback for competitive advantage.
- To understand the technologies with sentiment analysis research

1.6. The relevance of AI and the Dataset

Sentiment Analysis utilizes technology with Artificial Intelligence (AI) that enables efficient processing and interpretation of large volumes of text data. For this project, the dataset collected by Mc Auley Lab with researchers that contain Amazon product review dataset of all beauty products 2023 (Hou, et al., 2024) is being used. The dataset offers rich, diverse textual data with various ratings and sentiments which is suitable for the development of sentimental analysis models.

1.7. Description of the selected domain

This coursework focuses on Natural Language Processing (NLP), a branch of Machine Learning that helps computers in comprehending and interacting with human language and relates to following domains.

- **Natural Language Processing (NLP)**

Natural Language Processing (NLP) is a part of artificial intelligence that allows computers to understand, interpret and utilize natural human language. It is the core concept required for working of this coursework.

- **Supervised Learning**

Supervised learning involves a process where models are trained on labelled datasets containing text and its respective sentiment labels. It is a part of machine learning process.

- **Machine Learning**

Machine Learning is a part of artificial intelligence that identifies patterns from data using algorithms. It is used for tasks like sentiment analysis, that allows computers to categorize text input into groups such as neutral, positive, and negative sentiments. (SAS, 2024)

- **Classification**

Classification is a supervised machine learning technique that attempts to predict the correct label from input data using a model. When it comes to classification, the model is thoroughly trained on training data, assessed on test data, and then utilized to make predictions on fresh, unseen data. (datacamp, 2024) Sentiment analysis is treated as classification problem as it assigns a predefined sentiment category to each input text based on patterns learned from the training data.

- **Tokenization**

In NLP, tokenization is the process of breaking up a text sequence into smaller units called tokens. This technique is important primarily because it breaks down human language into chunks that are simpler to analyse for machine to understand. (datacamp, 2024)

2. Background

2.1. Research on the problem scenario

2.1.1. Research Paper 1

Title: Sentiment Analysis of Yelp's Ratings Based on Text Reviews

Authors: Yun Xu, Xinhui Wu, Qinxia Wang

Journal: Project Paper, Stanford University

Published Date: 2015

Description: The goal of this research (Yun Xu, 2015) is to forecast Yelp star ratings by analysing the linguistic content of reviews. The authors investigated supervised machine learning methods like Multiclass SVM, Perceptron, and Naive Bayes. With preprocessing techniques including stemming and stop-word removal, the dataset had more than 1.1 million reviews. Several feature selection techniques, including as sentiment lexicons and stemming, were evaluated, and precision and recall measures were used for assessment. The algorithm with the most reliable performance was Binarized Naive Bayes.

2.1.2. Research Paper 2

Title: Sentiment Analysis on Large Scale Amazon Product Reviews

Authors: Sayyed Johar, Samara Mubeen

Journal: International Journal of Scientific Research in Computer Science and Engineering (IJSRCSE)

Published Date: February 2020

Description: In this paper (Sayyed Johar, 2020), a supervised learning model was employed to analyse the sentiment of Amazon product reviews. By combining Chi-Square selection with feature extraction techniques like Bag-of-Words and TF-IDF, the authors used algorithms like Naive Bayes and Support Vector Machines (SVM). First, unlabelled dataset was classified using active learning in the study, which produced good classification accuracy. They found out that SVM marginally outperformed

Naive Bayes. when used with TF-IDF. The models were evaluated with metrics like precision, recall, and F1-score.

2.1.3. Research Paper 3

Title: Thumbs Up? Sentiment Classification Using Machine Learning Techniques

Authors: Bo Pang, Lillian Lee, Shivakumar Vaithyanathan

Journal: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)

Published Date: July 2002

Authors: Bo Pang, Lillian Lee, Shivakumar Vaithyanathan (Cornell University and IBM Almaden Research Centre)

Description: This research (Bo Pang, 2002) investigated the effectiveness of machine learning techniques in sentiment classification using IMDb movie reviews. The authors applied Naive Bayes, Maximum Entropy Classification (MaxEnt), and Support Vector Machines (SVM), utilizing unigram and bigram features. Evaluation focused on accuracy through cross-validation, revealing that SVMs performed best overall. The study also explored feature engineering, including negation tagging, and demonstrated the importance of accounting for context in sentiment analysis.

2.2. Algorithms and Evaluation Metrics Review

2.2.1. Research Paper 1

Title: Sentiment Analysis of Yelp's Ratings Based on Text Reviews

- **Algorithms Used:**

Naive Bayes, Perceptron, Multiclass Support Vector Machines (SVM)

- **Evaluation Metrics:**

Precision, Recall

Runtime which was highlighted as a critical factor for evaluating performance due to the large dataset size.

- **Performance Results:**
 - Naive Bayes achieved the best balance of precision and recall across all star ratings, particularly excelling when binarized features were used.
 - Perceptron performed well for extreme ratings (1 and 5) but struggled with mid-range ratings (2, 3, and 4).
 - SVM delivered inconsistent accuracy due to complexity in classifying multiple classes.
- **Key Insights:**
 - Preprocessing, including stop-word removal and stemming, significantly improved prediction accuracy.
 - The runtime was a notable limitation for more computationally intensive models, especially on large datasets.

2.2.2. Research Paper 2

Title: Sentiment Analysis on Large Scale Amazon Product Reviews

- **Algorithms Used:**

Naive Bayes, Support Vector Machines (SVM), Decision Trees
- **Evaluation Metrics:**
 - Accuracy, Precision, Recall, and F1-Score
 - Chi-Square Statistic: Used during feature selection to improve relevance and model efficiency.
- **Performance Results:**
 - SVM slightly outperformed Naive Bayes when paired with TF-IDF, with an accuracy of over 85%.
 - Naive Bayes delivered robust results with minimal computation but showed limitations when handling complex sentiment nuances.
 - Decision Trees was less effective in comparison, with lower accuracy and higher variance.

- **. Key Insights:**

- Active learning successfully labeled large datasets, improving both model accuracy and efficiency.
- Feature extraction techniques, particularly TF-IDF combined with Chi-Square, significantly boosted classifier performance

2.2.3. Research Paper 3

Title: Thumbs Up? Sentiment Classification Using Machine Learning Techniques

- **Algorithms Used:**

Naive Bayes, Maximum Entropy Classification (MaxEnt), Support Vector Machines (SVM)

- **Evaluation Metrics**

- Accuracy: Derived from three-fold cross-validation for unbiased results.
- Additional Analysis: Comparison between feature frequency versus presence and unigram versus bigram performance.

- **Performance Results:**

- Despite its simplicity, the Naive Bayes algorithm demonstrated remarkable performance, achieving approximately 81% accuracy with binary presence features.
- MaxEnt needed a lot more processing power but demonstrated similar accuracy.
- With binary presence features, SVM performed the best (~83%) and was particularly good at handling high-dimensional data.

- **Key Insights**

- Bigrams did not significantly improve accuracy, indicating their limited contextual contribution.
 - For SVM, negation tagging somewhat improved results, but for other algorithms, it made little difference.

2.2.4. Comparative Analysis

Algorithms:

- Naive Bayes proved to be consistently successful in every study because of its ease of use and capacity to process text data. However, it struggled with sentiment distinctions in Papers 2 and 3.
- SVM proved to be the most reliable algorithm for high-dimensional text categorization as it consistently outperformed others when paired with appropriate feature extraction techniques.
- Results from Perceptron and MaxEnt were inconsistent, frequently needing additional processing power without appreciable improvements in accuracy.

Insights:

- Feature engineering, particularly TF-IDF and negation tagging, significantly impacted classifier performance.
- Simpler algorithms like Naive Bayes excelled with well-preprocessed data, while SVMs handled more complex features better.

Metrics and Results:

- Papers 2 and 3 emphasized precision, recall, and F1-score for a balanced evaluation, providing more detailed insights than overall accuracy alone.
- Paper 1 highlighted runtime as a practical consideration, particularly for scalable solutions.
- Naive Bayes consistently achieved solid results across all studies, especially when preprocessing was effective.
- SVM was the top performer in both Papers 2 and 3, particularly when paired with advanced feature engineering like TF-IDF or binary presence.
- Metrics such as precision and recall were highlighted in Paper 1 but were less emphasized in Papers 2 and 3, where accuracy dominated evaluations.

2.2.5. Comparison of evaluation metrics

Metric	Paper 1 (Yelp)	Paper 2 (Amazon)	Paper 3 (IMDb)
Precision	38-58% (Naive Bayes), 36-53% (Perceptron)	Higher for SVM (exact % not mentioned)	Not explicitly stated but inferred higher for SVM (~83% accuracy).
Recall	32-76% (varies by rating)	Higher for SVM with TF-IDF (no exact %)	Not mentioned; Accuracy used as primary metric.
Accuracy	Not reported	Naive Bayes ~80%, SVM ~85%	Naive Bayes ~81%, MaxEnt ~80.4%, SVM ~83%
F1-Score	Not reported	Used but exact values not stated	Not mentioned
Runtime	Highlighted as critical for large datasets	Not emphasized	Not emphasized
Precision	38-58% (Naive Bayes), 36-53% (Perceptron)	Higher for SVM (exact % not mentioned)	Not explicitly stated but inferred higher for SVM (~83% accuracy).

Table 1: Metrics comparison of research papers

2.3. Block diagram

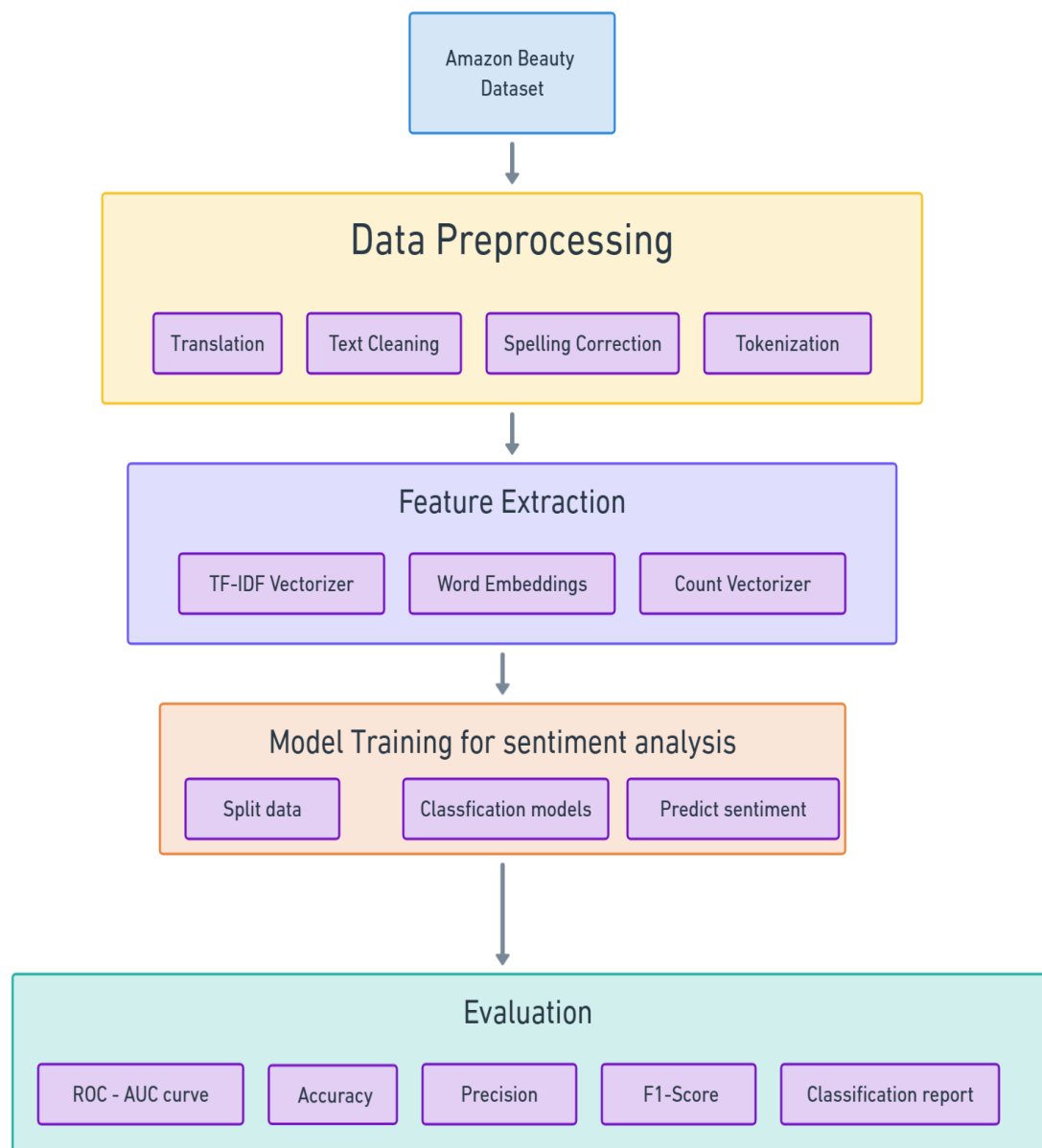


Figure 4: Block Diagram of the model development

2.4. Advantages of working around the problem domain

- Sentiment analysis gives actionable insight into user opinions, helping the enterprise move toward enhancement in customer experience, improving their products, and building focused marketing strategies.
- Logistic Regression, SVM, and Naive Bayes algorithms are computationally efficient for text classification. These algorithms can work efficiently with large datasets and provide good performance despite simple implementation.

2.5. Considered drawbacks revolving the problem domain

- The dataset has an uneven distribution of sentiments with more positive reviews than negative or neutral, it can lead to biased predictions.
- Algorithms like SVM or Logistic Regression may perform poorly on underrepresented classes without proper handling of imbalance.
- While models like Logistic Regression and Naive Bayes are computationally efficient, they may fail to capture complex relationships in text compared to deep learning approaches.
- Both TF-IDF and Count Vectorizer treat words independently, ignoring context and relationships between words. For example, "not good" might be treated similarly to "good," leading to misclassification.

2.6. Research on dataset

2.6.1. Dataset description

The dataset used for this project is the Amazon Reviews 2023 dataset (Hou, et al., 2024) that was collected in 2023 by McAuley Lab. The dataset was collected by researchers to pretrain BLAIR (Bridging Language and Items for Retrieval and Recommendation) for sentence embedding models specifically for recommendation scenarios. It includes customer reviews across various products, capturing diverse opinions and sentiments. The researchers have listed the dataset in various category from years ranging from 2013 to 2023 in a website [amazon review 2023](#).

The dataset chosen for this coursework falls into the category of All_Beauty with 633 thousand user's amazon review of the beauty products. The dataset is in JSON format. This data is essential for understanding customer behaviour, enabling businesses to refine their strategies and improve digital products.

rating		title	text	images	asin	parent_asin	user_id	timestamp	helpful_vote	verified_purchase
0	5	Such a lovely scent but not overpowering.	This spray is really nice. It smells really go...	[]	B00YQ6X8EO	B00YQ6X8EO	AGKHLEW2SOWHNMFQJGBEC7INQ	2020-05-05 14:08:48.923	0	True
1	4	Works great but smells a little weird.	This product does what I need it to do, I just...	[]	B081TJ8YS3	B081TJ8YS3	AGKHLEW2SOWHNMFQJGBEC7INQ	2020-05-04 18:10:55.070	1	True
2	5	Yes!	Smells good, feels great!	[]	B07PNNCSP9	B097R46CSY	AE74DYR3QUGVZJ3P7RFBGIX7XQ	2020-05-16 21:41:06.052	2	True
3	1	Synthetic feeling	Felt synthetic	[]	B09JS339BZ	B09JS339BZ	AFQLNQNQYFWQZPJQZS6V3NZU4QBQ	2022-01-28 18:13:50.220	0	True
4	5	A+	Love it	[]	B08BZ63GMJ	B08BZ63GMJ	AFQLNQNQYFWQZPJQZS6V3NZU4QBQ	2020-12-30 10:02:43.534	0	True

Figure 5: Amazon Dataset snippet

The dataset contains 701,528 rows and 10 columns, as summarized below:

Column Name	Description	Data Type
rating	Customer rating of the product (1-5 scale).	int64
title	Title of the review.	object
text	Full review text.	object
images	List of image URLs associated with the review.	object
asin	Amazon Standard Identification Number (unique product ID).	object
parent_asin	Parent product ID for grouped products.	object
user_id	Unique identifier of the reviewer.	object
timestamp	Date and time of the review submission.	datetime64[ns]
helpful_vote	Number of helpful votes received by the review.	int64
verified_purchase	Indicates whether the review is from a verified purchase (True/False).	bool

Table 2: Dataset description table

Reasons for choosing this dataset

There are various reasons for choosing this dataset for this coursework. Some are discussed below.

- The dataset is directly applicable to sentiment analysis tasks involving consumer feedback since the data is from Amazon.
- The dataset is reliable since a research paper is based on it to build a pretrained model.
- It contains 700,000 rows with diverse sentiments and features, providing enough data for effective training and testing of machine learning models.

- It includes the metadata such as ratings which can be encoded for sentiments.

Assumptions for the dataset

- The ratings in the dataset are reliable indicators of the sentiment expressed in the corresponding review.
- A rating of 5 indicates highly positive sentiment.
- A rating of 1 indicates highly negative sentiment.
- Middle ratings (e.g., 3) may indicate neutral or mixed sentiment.
- The sentiment expressed in the text aligns with the numerical rating.
- The dataset contains reviews in multiple languages (e.g., English, Spanish).
- The dataset contains user-generated content, which may have spelling errors, slang, and informal language.

Chart 1: Word Cloud for positive reviews.

The highest frequency has larger text which are visible such positive words like great product, work well, good quality in positive reviews as seen in the figure.



Chart 2: Word Cloud for negative reviews.

The word cloud for negative reviews shows that words associated with negative sentiment such as waste money, disappointment can be vividly seen in the figure.

Figure 6: Word cloud of positive reviews

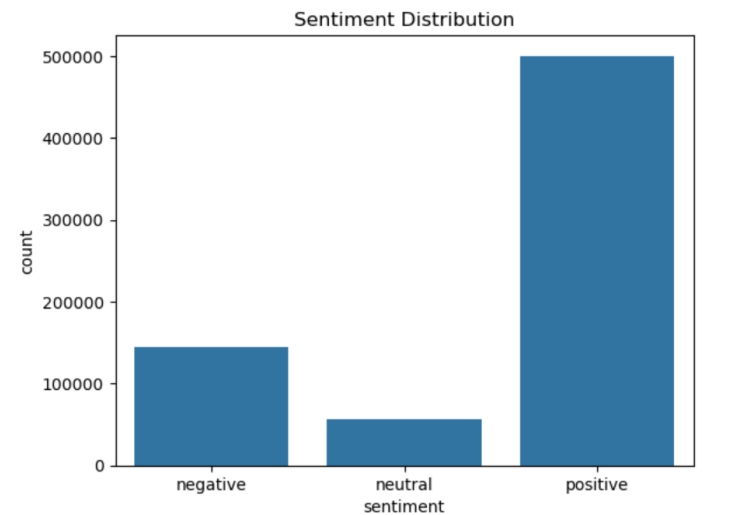


Figure 8: Sentiment Distribution figure

This distribution suggests that the dataset is skewed towards positive sentiments, which could reflect a generally satisfied or supportive community, or might indicate a bias in data collection or in how sentiments are classified, depending on the context of the data source.

3. Solution

3.1. Proposed solution to the problem

The proposed solution for this project involves developing a machine learning-based sentiment analysis system tailored for the Amazon Reviews 2023 dataset. For this project, algorithms such as Naive Bayes, Support Vector Machines (SVM), Logistic Regression, and K-Nearest Neighbors (KNN) are considered to classify sentiments while addressing the challenges of processing large-scale textual data effectively.

3.2. Explanation of the approach

The project will focus following approach after collection of data.

- **Data Exploration and Preprocessing**

After collection of data, data preprocessing is an important step in preparing dataset for building machine learning datasets. In this process, following steps are implemented.

- To maintain uniformity throughout the corpus, all text will be changed to lowercase.
- Ratings are encoded into labels of 0, 1, and 2.
- The text will be cleaned by removing unnecessary elements like special characters punctuation, and numbers that do not contribute to meaningful analysis.
- Then, data goes through a process of tokenization involves dividing the text into smaller parts called tokens and organizing it into forms like unigrams and n-grams.
- Libraries such as NLTK will be used to filter out common words like "and" "the," and so on to remove stopwords and filler Words.
- Stemming: To make the text simpler, words will be stripped of frequent suffixes (such as "playing" to "play") and reduced to their basic forms.
- Lemmatization: To guarantee that the output is consistent and understandable, words will be normalized to their basic forms (for example, "better" to "good").

- To make the text cleaner and more streamlined, unnecessary spaces will be eliminated.
- **Data Splitting**

The data will be split into train and testing dataset. If possible, validation set also be created for cross validation.
- **Vectorization**

To ensure uniformity and usability, the content using methods like TF-IDF (Term Frequency-Inverse Document Frequency) and Count vectorization for model training, the text will be tokenized and vectorized into numerical representations.
- **Model Training**

Training and validating the selected algorithms on pre-processed data.
- **Evaluation**

Using metrics such as accuracy, F1-score, and confusion matrix to assess performance.

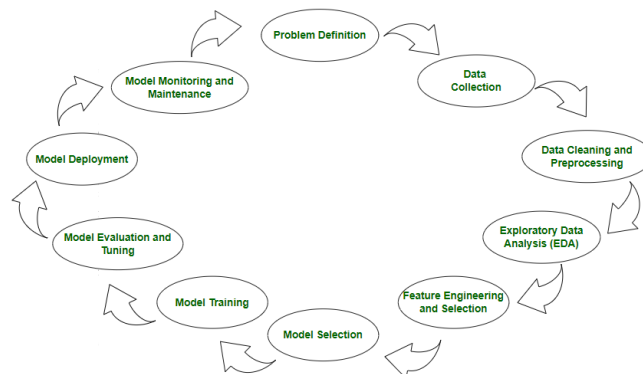


Figure 9: Machine Learning Model Development cycle

3.3. Explanation of AI Algorithms

3.3.1. Naive Bayes Classifier

The Naïve Bayes algorithm is a probabilistic algorithm based on Bayes' theorem, that assumes feature independence from its name 'naive'. Bayes' theorem is a formula that describes how to update probabilities based on new data. (Wohlwend, 2023)

The mathematical equation below shows the calculation of conditional probability of a class C, given predictor variable X.

$$P(C|X) = \frac{P(C|X) * P(C)}{P(X)}$$

Equation 1: Naive Bayes theorem

Reasons for choosing this algorithm for this coursework are as follows:

3.3.2. Support Vector Machines (SVM)

SVM is a classification model that identifies the optimal hyperplane to separate data into classes. SVMs are effective in high-dimensional spaces and are commonly used for text classification tasks, including sentiment analysis. (Vasista Reddy, 2018)

The reasons for choosing this algorithm for this coursework are as follows:

- By focusing on the optimal hyperplane, it reduces the risk of overfitting, especially in cases where the number of features is greater than the number of samples.
- It performs well with high-dimensional data, making it suitable for text classification tasks like sentiment analysis.
- It often achieves strong classification performance, ensuring reliable results in distinguishing different sentiments.

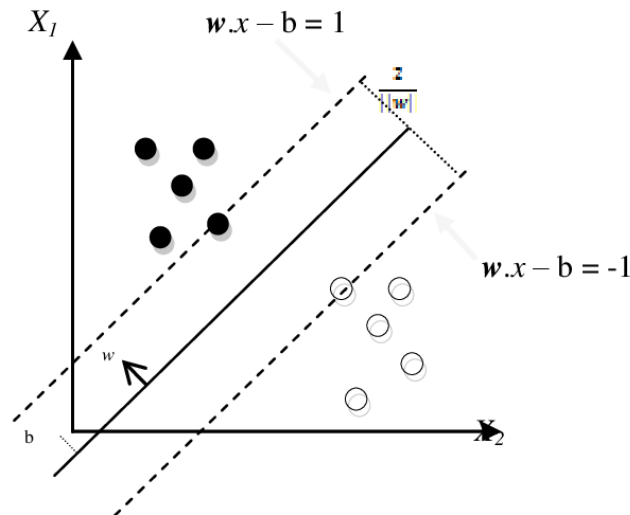


Figure 10: Figure of how SVM works (Vasista Reddy, 2018)

3.3.3. Logistic Regression

Logistic regression, despite its name suggesting regression, is a linear model designed for classification. It predicts the probability of a class label using input features and is simple to use, making it a common baseline for text classification tasks. (Wohllwend, 2023)

The basic formular of the logistic regression model where β_0 and β_1 are the parameters of the model:

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 * X)}}$$

Equation 2: Formula to calculate the function of Logistic regression

The reasons for choosing this algorithm for this coursework are as follows:

- It is straightforward to implement and provides clear insights into the relationship between features and the target class.
- It performs well as a baseline model, offering competitive results for text classification tasks.
- It provides clear insights into how each feature affects the probability of each class, that allows better understanding and explanation of the model's decisions.

3.3.4. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is an algorithm that classifies data points by comparing their distance to labelled examples within the feature space. KNN has K as parameter that is the number of nearest neighbours to include in the majority voting process. (Wohlwend, 2023)

The formula below calculates the distance between the new data point and every other data points in the training set.

$$\begin{aligned}d(p, q) &= d(q, p) \\&= \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} \\&= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}\end{aligned}$$

Equation 3: Distance formula used in KNN (Wohlwend, 2023)

Reasons for not choosing this KNN algorithm is due to following reasons:

- It struggles in high dimensional space as distance between the data points become more meaningful.
- It is insensitive to imbalance data, which was one of the main reasons, as the dataset is skewed towards positive reviews.
- It struggles with large datasets due to its computational complexity.

3.4. Explanation of Evaluation Metrics

Since the algorithm used for problem domain like sentiment analysis are based on classification model, the evaluation metrics are also be based on classification metrics.

The most popular metrics for measuring classification performance are accuracy, confusion matrix, precision-recall, macro avg, weighted avg and AUC-ROC.


		predicted condition			
		total population	prediction positive	prediction negative	Sensitivity  Recall = $\frac{\sum \text{TP}}{\sum \text{condition positive}}$
true condition	condition positive	True Positive (TP)	False Negative (FN) (Type II error)		
	condition negative	False Positive (FP) (Type I error)	True Negative (TN)		Specificity = $\frac{\sum \text{TN}}{\sum \text{condition negative}}$
	Accuracy = $\frac{\sum \text{TP} + \sum \text{TN}}{\sum \text{total population}}$	Precision= $\frac{\sum \text{TP}}{\sum \text{prediction positive}}$			F1 Score = $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$

Figure 11: Evaluation metrics for classification (Anon., 2018)

3.4.1. Confusion Matrix

A confusion matrix is a table that summarizes the performance of a classification model on test data with known true values. It includes:

- True Positives (TP) which is correct predictions of positive cases.
- True Negatives (TN) which is correct predictions of negative cases.
- False Positives (FP) which is incorrectly predicted positives (Type I error).
- False Negatives (FN) which is incorrectly predicted negatives (Type II error).

3.4.2. Precision

Precision measures proportion of positive identifications that were actually correct.

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

Equation 4: Precision formula

3.4.3. Recall

Recall, also known as sensitivity, measures the ability of a model to find all the relevant cases within the dataset.

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

Equation 5: Recall formula

3.4.4. Accuracy

Accuracy is the measure of overall correctness of prediction among the total number of cases examined.

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

Equation 6: Accuracy Formula

3.4.5. AUC-ROC curve

AUC means Area under the curve and ROC means Receiver operating Characteristic. The ROC curve is a probability curve that is graphed with the true positive rate (recall) and the false positive rate at different thresholds. The AUC (Area Under the Curve) measures how well the model distinguishes between classes, with 1 being perfect and 0.5 being random guessing. A higher AUC means the model is better at classification.

The ROC curve plots the True Positive Rate (Recall) against the False Positive Rate:

$$\text{Total Positive Rate (TPR)} = \text{Recall} = \frac{TP}{TP + FN}$$

Equation 7: True Positive Rate calculation

$$\text{False Positive Rate (FPR)} = \text{Recall} = \frac{FP}{FP + TN}$$

Equation 8: False Positive Rate calculation

3.4.1. Macro Avg

Macro Avg calculates the average of a metric such as precision, recall, or F1 score across all classes by giving each class equal weight regardless of its size. It treats all classes equally, which is useful for balanced datasets.

3.4.2. Weighted Avg

Weighted Avg calculates the average of a metric, weighted by the number of samples in each class. It gives more importance to larger classes, useful for imbalanced datasets.

3.5. Pseudocode of algorithms

3.5.1. Pseudocode of Naïve Bayes Classifier

START

Step 1: **IMPORT** necessary libraries

Step 2: **LOAD** training data with labelled sentiments

Step 3: **PREPROCESS** text data (lowercase, remove stopwords, etc.)

Step 4: Check if data is clean and processed

IF data is not clean or not processed

THEN

 Go back to Step 3

ELSE

 Proceed to Step 5

Step 5: **SELECT** features with sentiment label as the target variable (y)

Step 6: **SPLIT** dataset

Step 7: **TRAIN** Naive Bayes model using preprocessed training data

Step 8: **EVALUATE** the model (accuracy, precision, recall)

Step 9: **CHECK** model performance

IF model performance is not acceptable

THEN

 Perform hyperparameter tuning on the model

 Go back to Step 7

ELSE

 Proceed to Step 10

Step 10: **DISPLAY** sentiment predictions and evaluation metrics

END

3.5.2. Pseudocode of SVM model

START

Step 1: **IMPORT** necessary libraries

Step 2: **LOAD** training data with labelled sentiments

Step 3: **PREPROCESS** text data (lowercase, remove stopwords, etc.)

Step 4: Check if data is clean and processed

IF data is not clean or not processed

THEN

 Go back to Step 3

ELSE

 Proceed to Step 5

Step 5: **SELECT** features with sentiment label as the target variable (y)

Step 6: **SPLIT** dataset

Step 7: **TRAIN** SVM model using preprocessed training data

Step 8: **EVALUATE** the model (accuracy, precision, recall)

Step 9: **CHECK** model performance

IF model performance is not acceptable

THEN

 Perform hyperparameter tuning on the model

 Go back to Step 7

ELSE

 Proceed to Step 10

Step 10: **DISPLAY** sentiment predictions and evaluation metrics

END

3.5.3. Pseudocode of Logistic Regression Model

START

Step 1: **IMPORT** necessary libraries

Step 2: **LOAD** training data with labelled sentiments

Step 3: **PREPROCESS** text data (lowercase, remove stopwords, etc.)

Step 4: Check if data is clean and processed

IF data is not clean or not processed

THEN

 Go back to Step 3

ELSE

 Proceed to Step 5

Step 5: **SELECT** features with sentiment label as the target variable (y)

Step 6: **SPLIT** dataset

Step 7: **TRAIN** Logistic Regression Model using preprocessed training data

Step 8: **EVALUATE** the model (accuracy, precision, recall)

Step 9: **CHECK** model performance

IF model performance is not acceptable **THEN**

 Perform hyperparameter tuning on the model

 Go back to Step 7

ELSE

 Proceed to Step 10

Step 10: **DISPLAY** sentiment predictions and evaluation metrics

END

3.6. Diagrammatical representation

3.6.1. Flowchart of Naïve Bayes Classifier

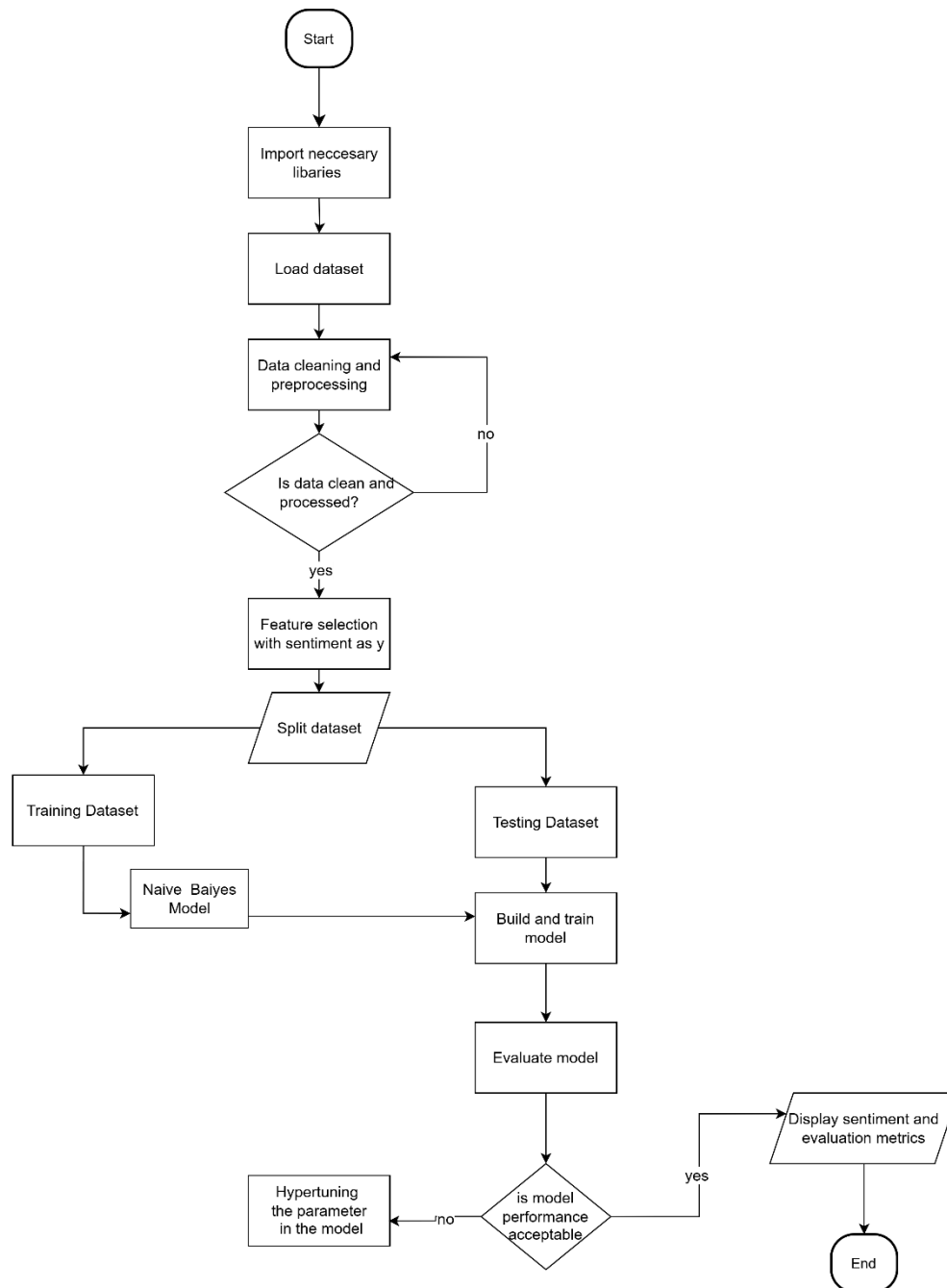


Figure 12: Flowchart of Naïve Bayes Classifier

3.6.2. Flowchart of SVM

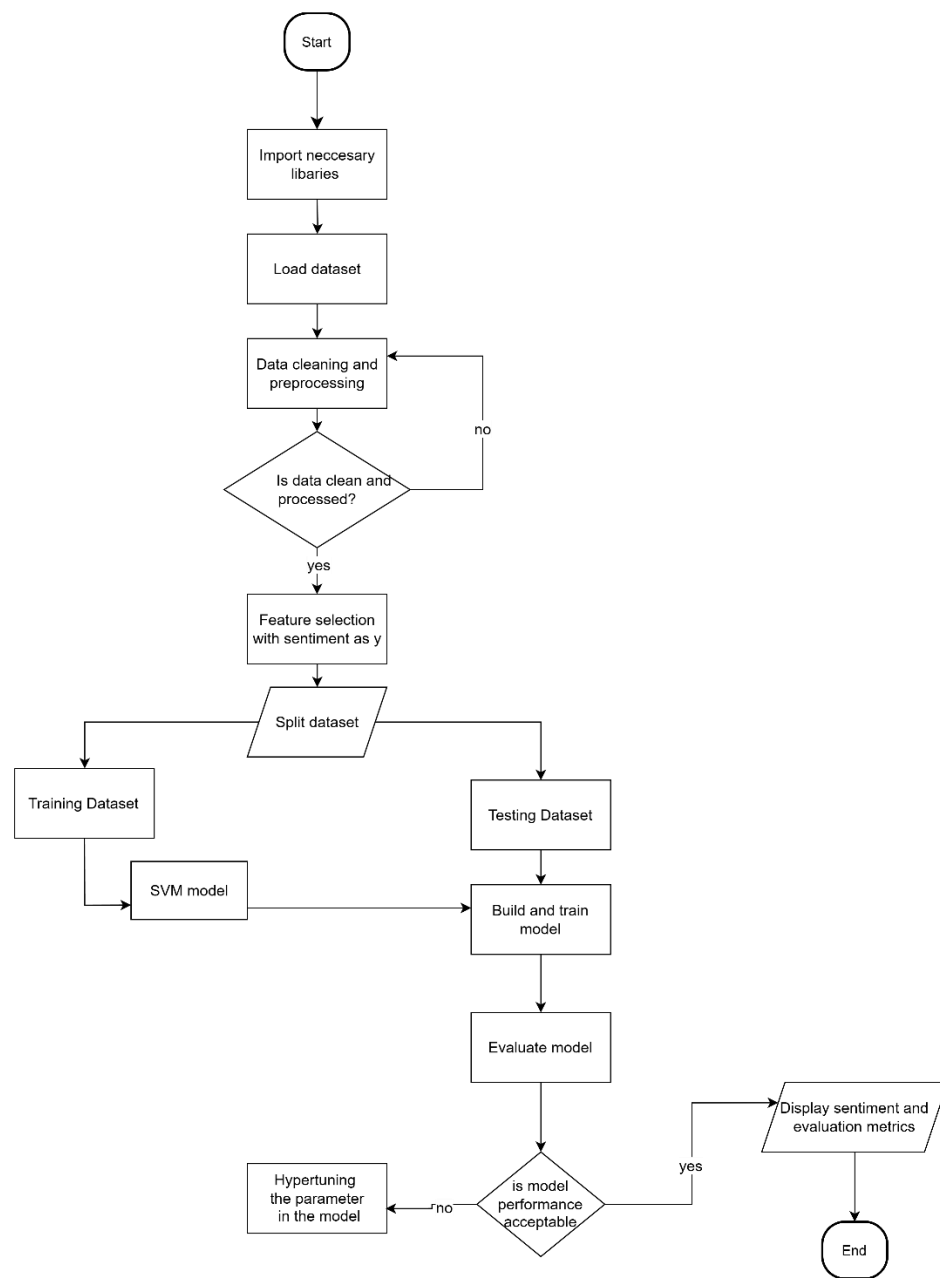


Figure 13: Flowchart of SVM

3.6.3. Flowchart of Logistic Regression

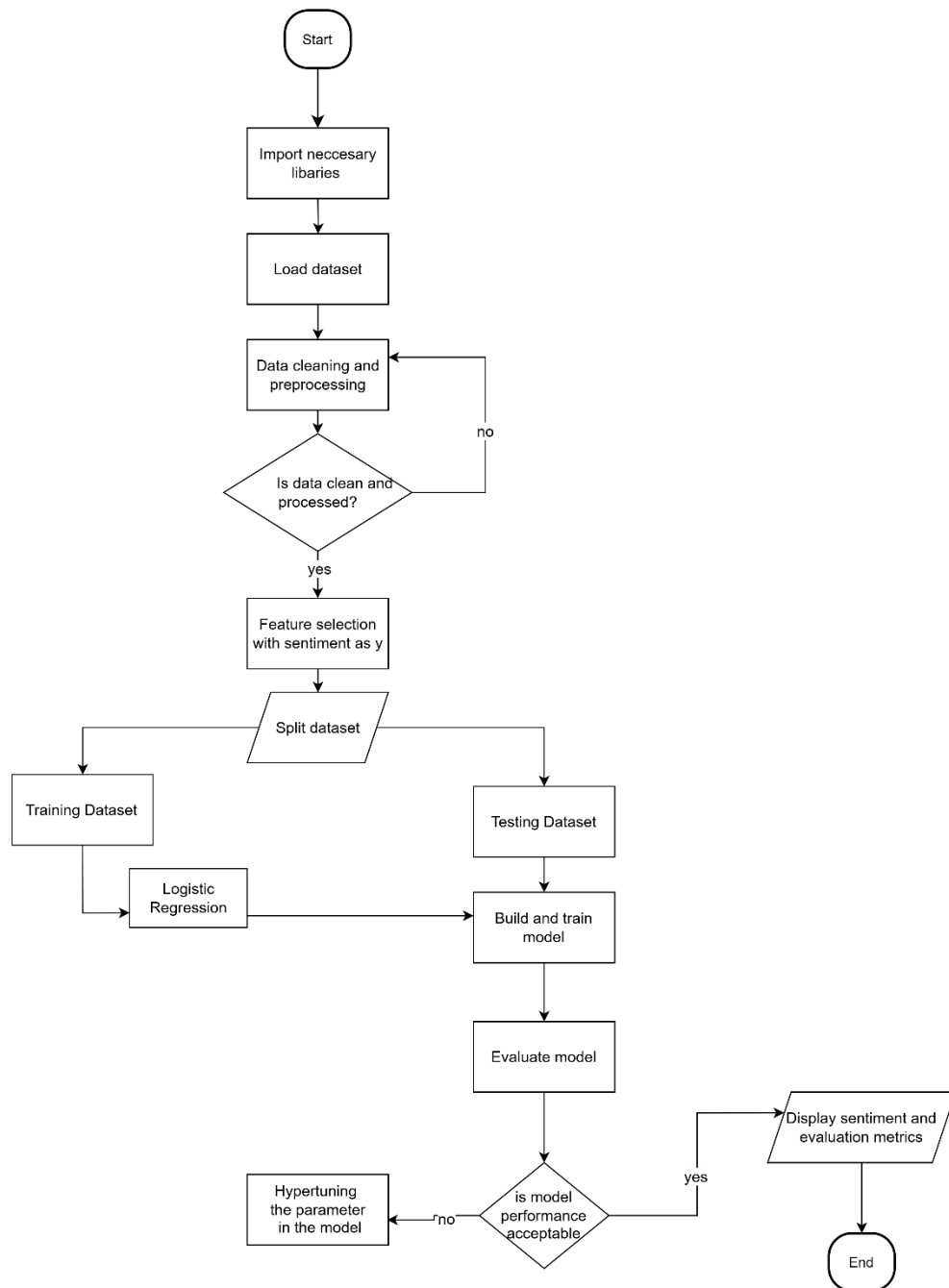


Figure 14: Flowchart of Logistic Regression

4. Development

4.1. Development Tools

4.1.1. Environment

- Jupyter Notebook

Jupyter Notebook (Jupyter, 2025) is a software used in this coursework because of its interactive computing across all programming languages. The development file can be exported into HTML file supported by this software. It is an integrated development environment for machine learning and data analysis.



Figure 15: Logo of Jupyter Notebook

4.1.2. Programming Language

- Python

Python (Python, 2025) is a programming language suitable for this coursework because of its libraries built for the machine learning. The large open-source community has built one of the largest machine learning communities supporting the tasks and models required to develop a machine learning solution. It also supports NLTK libraries.



Figure 16: Logo of Python

4.1.3. Libraries

There are important libraries supported by Python used in this coursework which are explained below:

- NumPy

NumPy is a library for numerical operations and handling arrays. It is useful for fast computations.

- Pandas

Pandas is a library used for data manipulation and analysis, particularly with tabular data (DataFrames).

- Matplotlib

Matplotlib is a library for plotting and visualization such as histograms, bars that are crucial for understanding data and errors.

- Seaborn

Seaborn is a library enhanced and aesthetically pleasing visualizations.

- WordCloud

WordCloud is a tool utilized for generating visual representations of word frequencies, allowing users to create engaging and informative word cloud visualizations.

- NLTK

NLTK is a library for processing language. It helps with tasks like breaking text into parts, removing common words, and reducing words to their base forms.

- Scikit-learn

Scikit-learn is a powerful machine learning library that facilitates the development of models, data preprocessing, and performance evaluation using metrics such as precision, recall, and F1 score. It supports the implementation of various machine learning algorithms, including Naive Bayes, Support Vector Machines (SVM), and Logistic Regression.

4.2. Model development

For the model development of sentiment analysis, the approach discussed in model development cycle are followed which are further explained.

- Data preprocessing
- Data Splitting
- Vectorization
- Model Training
- Evaluation

4.2.1. Data Preprocessing

- **Oversampling**

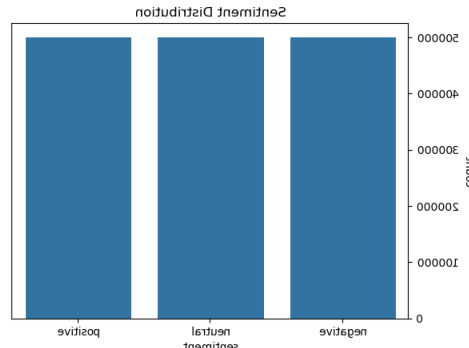
Oversampling is used in sentiment analysis when the dataset is imbalanced, meaning one sentiment has far more examples than another. It balances the dataset by duplicating minority class samples, helping the model learn equally from all sentiments and improving classification performance. This reduces bias towards the majority class.

```
# Oversample the minority classes
neutral_oversampled = resample(neutral,
                                replace=True, # Sample with replacement
                                n_samples=len(positive), # Match majority class size
                                random_state=42)

negative_oversampled = resample(negative,
                                 replace=True,
                                 n_samples=len(positive),
                                 random_state=42)
```

Figure 17: Oversampling data

```
The shape of the new dataframe is (1500321, 2)
sentiment
positive    500107
neutral     500107
negative    500107
Name: count, dtype: int64
```



- **Convert text to lowercase.**

`str.lower()` is a Python string method that converts all characters in the text to lowercase, ensuring uniformity. In pandas, `pandas.Series.str.lower()` applies this to each element in a series.

```

df['text'] = df['text'].str.lower()
print (df['text'].head())

0    this spray is really nice. it smells really go...
1    this product does what i need it to do, i just...
2                                     smells good, feels great!
3                                     felt synthetic
4                                     love it
Name: text, dtype: object

```

Figure 18: Converting text to lowercase

- **Remove stopwords, punctuation, and special characters using NLTK**

A function is created using `nltk.corpus.stopwords` that provides a list of common words to remove (like "the", "a", "an") and `string.punctuation` that gives you characters to exclude. This is the very important step for cleaner text data.

```

import re
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')

stop_words = set(stopwords.words('english'))

# Function to clean text
def clean_text(text):
    text = re.sub(r'^a-zA-Z\s', '', text) # Remove punctuation and special characters
    words = text.split() # Tokenize by splitting on whitespace
    words = [word for word in words if word not in stop_words] # Remove stopwords
    return ' '.join(words)

# Apply cleaning to the text column
df['text'] = df['text'].apply(clean_text)
print (df['text'].head())

[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\nepal\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
0    spray really nice smells really good goes real...
1    product need wish odorless soft coconut smell ...
2                                     smells good feels great
3                                     felt synthetic
4                                     love
Name: text, dtype: object

```

Figure 19: Removing stopwords, special character using NLTK

- **Tokenization and stemming/lemmatization.**

`nltk.word_tokenize()` breaks text into words or tokens. Stemming, done with `nltk.stem.PorterStemmer()`, reduces words to their root form (e.g.,

"running" to "run"). Lemmatization, using `nlk.stem.WordNetLemmatizer()`, does this more intelligently, considering the word's context (e.g., "better" to "good").

```
from nltk.stem import PorterStemmer, WordNetLemmatizer
nltk.download('wordnet')
nltk.download('omw-1.4')

stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()

# Function to apply stemming
def apply_stemming(text):
    words = text.split()
    words = [stemmer.stem(word) for word in words] # Apply stemming
    return ' '.join(words)

# Function to apply Lemmatization
def apply_lemmatization(text):
    words = text.split()
    words = [lemmatizer.lemmatize(word) for word in words] # Apply Lemmatization
    return ' '.join(words)

# Choose either stemming or Lemmatization (do not use both at the same time)
df['text'] = df['text'].apply(apply_lemmatization) # Use Lemmatization
print(df['text'].head())
```

Figure 20: Using Lemmatization to the text

- **Feature Engineering**

`sklearn.feature_extraction.text.CountVectorizer` turns text into a matrix counting word occurrences, while `TfidfVectorizer` does the same but weights words by their importance (TF-IDF). These are essential for transforming text into a format machine learning algorithms can process.

- **Sentiment Encoding**

`sklearn.preprocessing.LabelEncoder` converts categorical labels (like sentiment categories) into numerical labels, which is necessary for many algorithms that require numeric input. This process assigns a unique integer to each unique category (e.g., 0 for 'negative', 1 for 'neutral', 2 for 'positive').

```
Label encoding mapping:
negative: 0
neutral: 1
positive: 2
```

Figure 21: Label encoded for sentiments

4.2.2. Data Splitting

When splitting data for machine learning, functions like *train_test_split* from *sklearn.model_selection* are used to divide dataset into training and testing sets. The parameters for this function are the feature matrix (X), target vector (y), set a test size (e.g., 0.2 for 20% test data), and include a random state for reproducibility. This split helps in training the model on one subset of data and evaluating its performance on unseen data, ensuring the model generalizes well.

The process is broken down into following steps:

- Split raw data into training and testing sets.
- Fit the TF-IDF vectorizer on the training data.
- Transform both training and testing data using the fitted vectorizer.
- Train and evaluate the model.

Advantages of data splitting before vectorization are as follows:

- The TF-IDF vectorizer learns vocabulary and IDF values only from the training set. The model doesn't gain information from the test set during training.
- It allows realistic evaluation as the test set remains unseen during training. It provides a more honest evaluation of how the model will perform on new, unseen data.
- It reduces the risk of overfitting, as the model isn't influenced by the test set's distribution during feature extraction.

```
from sklearn.model_selection import train_test_split

X = df['text']          # Feature: review text
y = df['sentiment_encoded'] # Target Label: sentiment

# 80% training, 20% testing split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 22: Data splitting

4.2.3. Vectorization

Vectorization is the process of converting text data into numerical features that machine learning algorithms can understand. Here's an overview of two key vectorization techniques:

- **TF-IDF**

TF-IDF adjusts the raw term frequency (how often a word appears in a document) by the inverse document frequency (how common or rare a word is across all documents). This method helps in highlighting words that are important in a particular document but not too common across the entire corpus.

It is ideal for scenarios where word importance in context is crucial. It's particularly useful when common words need to be de-emphasized, and rare, document-specific terms need to be given more weight.

$$\text{TF}(\text{term}, \text{document}) = \frac{\text{Number of occurrences of term in document}}{\text{Total number of terms in document}}$$

$$\text{IDF}(\text{term}) = \log \left(\frac{\text{Number of documents}}{\text{Number of documents with term}} \right)$$

$$\text{TF-IDF}(\text{term}, \text{document}) = \text{TF}(\text{term}, \text{document}) \times \text{IDF}(\text{term})$$

Equation 9: Formula for calculating TF-IDF

- **Count Vectorizer**

Count Vectorizer converts a collection of text documents to a matrix of token counts. Each row represents a document, and each column represents a unique word in the vocabulary. The cell values are the counts of how many times each word appears in each document.

It is best suited for basic text analysis where the frequency of words is the primary concern without regard to their importance in the broader document set. It's a simpler approach and can be effective for models where word frequency is a significant feature.

4.2.4. Model Training

For training of the machine learning model in Sentiment analysis, three algorithms are used.

- **Naïve Bayes**

Naive Bayes is a family of probabilistic algorithms based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Multinomial Naive Bayes is used with `MultinomialNB()` in this coursework for discrete data, particularly for text classification tasks where the features represent the frequency of words.

The model is fitted on `X_train_count` which is X train after count vectorization.

```
: # Train Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train_count, y_train)

: ▼ MultinomialNB ⓘ ?
MultinomialNB()
```

Figure 23: Model training of Naive Bayes

- **Logistic Regression**

Logistic regression training involves fitting a model to binary outcome data using `LogisticRegression()` with its hyperparameters `multi_class`, `solver` and `max_iter`. The process includes defining the logistic function, optimizing parameters through maximum likelihood estimation.

```
print("Logistic Regression:")
lr_model = LogisticRegression(multi_class='multinomial',
                             solver='lbfgs',
                             max_iter=1000)
lr_model.fit(X_train_tfidf, y_train)

Logistic Regression:
▼ LogisticRegression ⓘ ?
LogisticRegression(max_iter=1000, multi_class='multinomial')
```

Figure 24: Model training of Logistic Regression

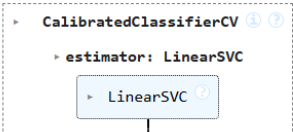
- **Support Vector Machine**

For SVM, Linear Support Vector Classifiers (LinearSVC) is trained with probability calibration. The LinearSVC is initialized with parameters; C=1.0 which is a regularization strength, maximum number of iterations for the optimization max_iter=1000 and random_state=42.

CalibratedClassifierCV wraps the LinearSVC to enable probability predictions using the sigmoid method. This is necessary because LinearSVC doesn't provide probabilities natively.

The LinearSVC learns to separate the classes with a hyperplane, and the calibration step ensures that the model can output probabilities for each class. This is useful for tasks like confidence scoring and decision-making.

```
# Train LinearSVC with calibration for probabilities
linear_svc = LinearSVC(C=1.0, max_iter=1000, random_state=42)
calibrated_svc = CalibratedClassifierCV(linear_svc, method="sigmoid")
calibrated_svc.fit(X_train_scaled, y_train)
```



```
graph TD
    CCV[CalibratedClassifierCV] -- estimator --> LSVC[LinearSVC]
```

Figure 25: Linear SVC model training

5. Results

5.1. Achieved Model Results

5.1.1. Classification Report and Confusion Matrix

a. Logistic Regression

```
y_pred_lr = lr_model.predict(X_test_tfidf)
print(classification_report(y_test, y_pred_lr))
```

	precision	recall	f1-score	support
0	0.72	0.73	0.73	100026
1	0.64	0.62	0.63	100567
2	0.79	0.80	0.79	99472
accuracy			0.72	300065
macro avg	0.72	0.72	0.72	300065
weighted avg	0.71	0.72	0.72	300065

Figure 26: Classification report of Logistic Regression with TFIDF

The classification report of Logistic Regression is generated using scikit-learn's `classification_report`. It shows that 72% of the predictions are correct for class 0. For class 2, the model captured 80% of positive sentiments.

The key insight from the report is that the model performed best on class 2 (positive sentiments) with an F1-score of 0.79, while it struggles slightly with class 1 (neutral) as seen from its lower precision, recall, and F1-score. The overall accuracy (72%) indicates moderate performance, with room for improvement, especially for underrepresented or challenging classes like class 1.



Figure 27: Confusion Matrix of Logistic Regression

The figure above shows the confusion matrix for a logistic regression model. The rows represent the actual (true) classes (0, 1, 2) and the columns represent the predicted classes (0, 1, 2). The true negative is 73338 which means they were correctly predicted as Class 0.

b. SVM

The classification report of SVM shows that precision, recall, and F1-score are all highest for class 2 (positive sentiment) at 0.78, 0.80, and 0.79, respectively. Class 1 (neutral sentiment) performs the worst, with a recall of 0.59, indicating difficulty capturing neutral sentiments. The overall accuracy of the model is 71%, and both the macro average and weighted average metrics are 0.71, showing a balanced performance across all classes.

	precision	recall	f1-score	support
0	0.71	0.74	0.72	100026
1	0.64	0.59	0.62	100567
2	0.78	0.80	0.79	99472
accuracy			0.71	300065
macro avg	0.71	0.71	0.71	300065
weighted avg	0.71	0.71	0.71	300065

Figure 28: SVM classification report

The confusion matrix visualizes the logistic regression model's predictions. For class 0, 73,828 predictions were correct, with 19,934 misclassified as class 1 and 6,264 as class 2. Class 1 has significant misclassification, with 24,299 instances wrongly predicted as class 0 and 16,553 as class 2. Class 2 performs well, with 79,840 correct predictions and fewer misclassifications compared to other classes. This highlights the model's strength in identifying positive sentiments and its struggles with neutral sentiment classification.



Figure 29: Confusion matrix of SVM

c. Naïve Bayes

The classification report of Naïve Bayes helps to derive information about precision of 0.75 and recall of 0.73 with an F1-score of 0.74 for Class 0. It also indicates that the model struggles more with Class 1 and has best performance for positive sentiment Class 2 with precision of 0.78, recall of 0.82, and an F1-score of 0.80.

```
# Evaluate Naive Bayes
y_pred_nb = nb_model.predict(X_test_count)
print(classification_report(y_test, y_pred_nb)) #to print prec
```

	precision	recall	f1-score	support
0	0.75	0.73	0.74	100026
1	0.66	0.65	0.65	100567
2	0.78	0.82	0.80	99472
accuracy			0.73	300065
macro avg	0.73	0.73	0.73	300065
weighted avg	0.73	0.73	0.73	300065

Figure 30: Classification report of Naive Bayes with TFIDF

The confusion matrix of Naïve Bayes shows that for Class 0, it correctly predicted 72726 times, for Class 1, it correctly predicted correctly predicted 65785 times and for Class 2, it correctly predicted 81852 times. It shows that the model performed well in Class 2 with positive sentiment.

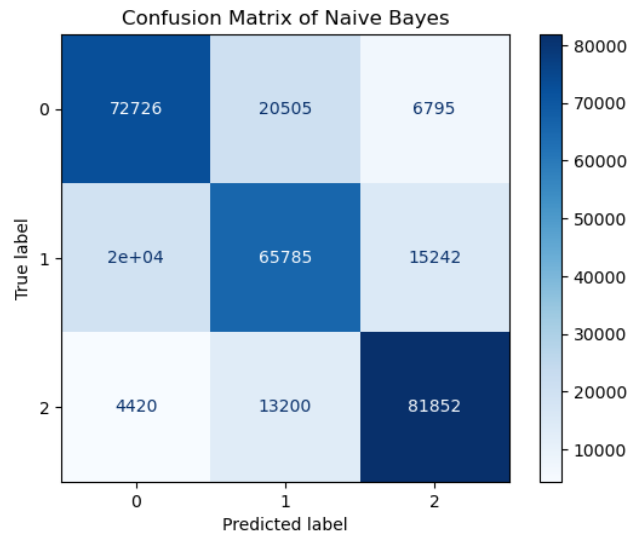


Figure 31: Confusion matrix of Naive Bayes

5.1.2. Precision Comparison Table of Balanced data

The precision comparison table for Naive Bayes, Logistic Regression, and SVM of the balanced data are as follows:

Class	Naive Bayes	Logistic Regression	SVM
Class 0	0.75	0.71	0.71
Class 1	0.66	0.64	0.64
Class 2	0.78	0.78	0.78

Table 4: Precision Comparison Table

It shows that all models perform similarly on class 2, while Naive Bayes generally outperforms Logistic Regression and SVM for Class 0 and Class 1.

5.1.3. Recall Comparison Table

The table below compares the recall of Naive Bayes, Logistic Regression, and SVM across classes. Naive Bayes performs slightly better overall, especially for class 1 and class 2, with the highest recall for class 2 at 0.82. Logistic Regression and SVM have similar performance, but SVM struggles more with class 1 at 0.59 recall, indicating difficulty in identifying neutral sentiments.

Class	Naive Bayes	Logistic Regression	SVM
Class 0	0.73	0.73	0.74
Class 1	0.65	0.62	0.59
Class 2	0.82	0.80	0.80

Table 5: Recall Comparison table of balanced dataset

5.1.4. F1-Score Comparison Table

The table below shows F1-scores of Naive Bayes, Logistic Regression, and SVM. Naive Bayes performs slightly better overall, achieving the highest F1-scores for all classes and macro/weighted averages, especially excelling in class 2 (positive sentiment) at 0.80. Logistic Regression and SVM are closely matched, with Logistic Regression performing marginally better than SVM in all metrics.

Class	Naive Bayes	Logistic Regression	SVM
Class 0	0.74	0.73	0.72
Class 1	0.65	0.63	0.62
Class 2	0.80	0.79	0.79
Macro Avg	0.73	0.72	0.71
Weighted Avg	0.73	0.72	0.71

Table 6: F1 score Comparison table of models in balanced dataset

Overall

- Naive Bayes has slightly better macro and weighted averages across precision, recall, and F1-score.
- Logistic Regression performs marginally better than SVM due to slightly higher averages and consistency across metrics.

Key Insights

- Class 2 (Positive Sentiment):

All models perform best for class 2, with Naive Bayes and Logistic Regression achieving F1-scores of 0.80 and 0.79, respectively. SVM slightly lags but still achieves strong results.

- **Class 1 (Neutral Sentiment):**

This class is the hardest for all models. Naive Bayes achieves the highest recall, while Logistic Regression and SVM show lower performance in recall and F1-scores.

- **Class 0 (Negative Sentiment):**

Naive Bayes has a slight edge in precision (0.75) compared to Logistic Regression and SVM, which are closely aligned at 0.72 and 0.71.

5.1.5. Accuracy Comparison Table

The comparison table of accuracy across all the tables shows that Naive Bayes achieves the highest accuracy at 73%, making it the most effective model for this task. Logistic Regression follows closely with an accuracy of 72%, showing competitive performance. SVM has the lowest accuracy at 71%, indicating that it struggles slightly more compared to the other models in this specific dataset.

Model	Accuracy
Naive Bayes	0.73
Logistic Regression	0.72
SVM	0.71

Table 7: Accuracy Comparison Table

5.1.6. Model Comparison

- **Naive Bayes:**

- It performs best overall with the highest macro (0.73) and weighted (0.73) averages with balanced dataset.
- It has strong recall and F1-scores for class 2 (positive sentiment), making it well-suited for datasets with clear sentiment distinctions.
- It handles neutral sentiment (class 1) better than the other models but still has room for improvement.

- **Logistic Regression:**

- It is a balanced performer with macro and weighted averages of 0.72.
- It slightly underperforms compared to Naive Bayes, particularly for class 1 (neutral sentiment).
- It is a reliable choice for general-purpose text classification tasks due to its simplicity and interpretability.

- **SVM:**

- It performs similarly to Logistic Regression but slightly lags behind with macro and weighted averages of 0.71.
- It struggles the most with class 1 (neutral sentiment), indicating difficulty in handling nuanced sentiments.
- It is best suited for datasets with high dimensionality or clear class separations.

5.1.7. ROC-AUC Curve of Balanced Dataset

- **ROC-AUC curve of Naïve Bayes**

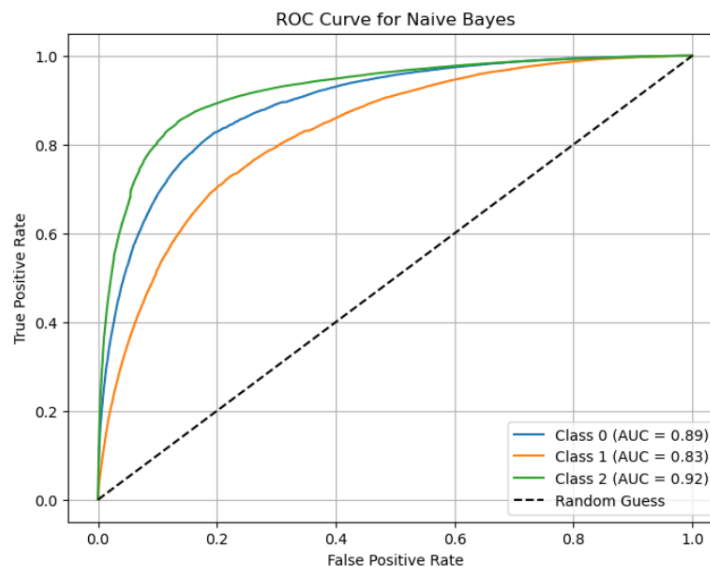


Figure 32: ROC-AUC curve of Naive Bayes

Class 0 (AUC = 0.89) performs well, showing a strong ability to distinguish negative sentiments from others.

Class 1 (AUC = 0.83) has moderate performance, indicating challenges in separating neutral sentiments.

Class 2 (AUC = 0.92) has best performance among the three classes, with excellent separability for positive sentiments.

- **ROC-AUC curve of Logistic Regression**

Class 0 (AUC = 0.89 matches Naive Bayes with strong performance for negative sentiment classification.

Class 1 (AUC = 0.82) slightly lower AUC compared to Naive Bayes, indicating similar struggles with neutral sentiment classification.

Class 2 (AUC = 0.93) outperforms Naive Bayes slightly, achieving the highest AUC for positive sentiments.

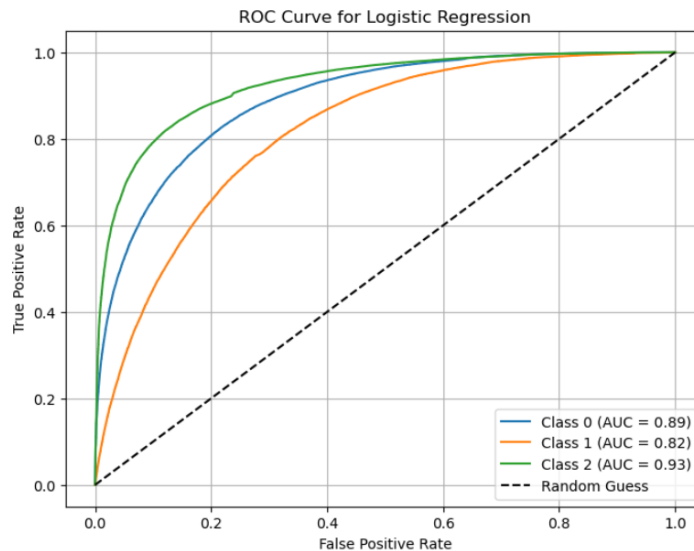


Figure 33: ROC-AUC curve of Logistic Regression

- **ROC-AUC curve of SVC**

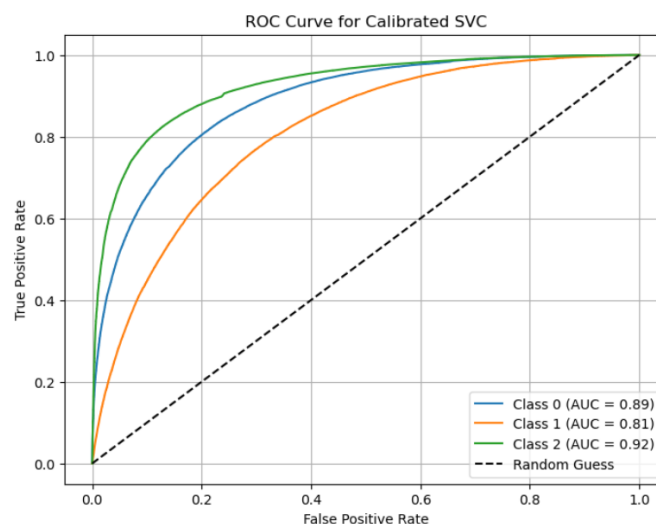


Figure 34: ROC-AUC curve of SVC

Class 0 (AUC = 0.89) has similar performance to the other models for negative sentiments.

Class 1 (AUC = 0.81) has lowest AUC among the three models for neutral sentiments, confirming struggles in distinguishing this class.

Class 2 (AUC = 0.92) matches Naive Bayes for positive sentiments, but slightly behind Logistic Regression.

5.1.8. Comparison Table of unbalanced dataset

In milestone 1 of this coursework, unbalanced dataset was used without any sampling performed where SVM had best overall performance, especially with TFIDF, due to its ability to find an optimal hyperplane in high-dimensional spaces. Naive Bayes also performed well with Count Vectorizer for simple, balanced datasets but struggled with skewed distributions or subtle distinctions in text. Logistic Regression achieved balanced performance across TFIDF and Count Vectorizer but slightly underperforms compared to SVM.

6. Conclusion

6.1. Analysis of the work done

This coursework focused on building a sentiment analysis system using machine learning models, by evaluating their performance on the Amazon Reviews 2023 dataset. The goal was to identify customer sentiments: positive, neutral, or negative through classification algorithms and to compare the performance of various machine learning models using Count Vectorizer and TFIDF vectorization techniques.

The comparison between model performance with different vectorization techniques and evaluation metrics showed that Support Vector Machine (SVM) and Logistic Regression outperformed Naive Bayes, especially when combined with TFIDF.

6.2. How the solution addresses real world problems

The project directly addresses the real-world challenges faced by e-commerce giants like Amazon. This project analyses the customer data to give customers insights and automates the sentiment analysis of Amazon reviews that can be adopted by other businesses. IT also aids in identifying top performing products and services that require improvement. Since it automates the operation, the company saves time and resources. Additionally, the solution can be adapted to other domain such as social media monitoring, product reviews, and customer feedback analysis, enhancing its versatility.

6.3. Further work

Future work in this project can involve hyper tuning of the parameters, using word embedding, more advanced models like LSTM, BERT, and further data preprocessing techniques.

7. References

- Akamai, 2024. *what is tokenization*. [Online]
Available at: <https://www.akamai.com/glossary/what-is-tokenization#:~:text=In%20the%20world%20of%20data,system%20that%20created%20the%20token.>
[Accessed 22 December 2024].
- AlmaBetter, 2025. *Metrics for Classification Model*. [Online]
Available at: <https://www.almabetter.com/bytes/tutorials/data-science/classification-metrics>
[Accessed 8 January 2025].
- Amazon, 2023. *Amazon Review 23*. [Online]
Available at: <https://amazon-reviews-2023.github.io/>
[Accessed 22 December 2024].
- Anon., 2018. *Shuzhan Fan*. [Online]
Available at: <https://shuzhanfan.github.io/2018/02/model-evaluation-metrics/>
[Accessed 8 January 2025].
- Bigblue, 2024. *sentiment-analysis*. [Online]
Available at: <https://bigblue.academy/en/sentiment-analysis>
[Accessed 22 December 2024].
- Bo Pang, L. L. S. V., 2002. *Thumbs up? Sentiment Classification using Machine Learning Techniques*. New york, Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002).
- Datacamp, 2024. *datacamp*. [Online]
Available at: <https://www.datacamp.com/blog/classification-machine-learning>
[Accessed 22 December 2024].
- Datacamp, 2024. *what is tokenization*. [Online]
Available at: <https://www.datacamp.com/blog/what-is-tokenization>
[Accessed 22 December 2024].
- Hou, Y. et al., 2024. *Bridging Language and Items for Retrieval and Recommendation*. [Online]
Available at: <https://arxiv.org/abs/2403.03952>
[Accessed 10 January 2025].
- Ibm, 2024. *What is sentiment analysis?*. [Online]
Available at: <https://www.ibm.com/think/topics/sentiment-analysis>
[Accessed 23 December 2024].

Rosemin Anderson, 2024. <https://www.qualtrics.com/>. [Online]
Available at: <https://www.qualtrics.com/experience-management/customer/customer-sentiment/#:~:text=It%20allows%20companies%20to%20see,excellent%20starting%20point%20for%20action.>
[Accessed 22 December 2024].

SAS, 2024. *Evolution of Machine Learning*. [Online]
Available at: https://www.sas.com/en_us/insights/analytics/machine-learning.html#:~:text=Machine%20learning%20is%20a%20method,decisions%20with%20minimal%20human%20intervention.
[Accessed 22 December 2024].

Sayed Johar, S. M., 2020. Sentiment Analysis on Large Scale Amazon Product Reviews. *International Journal of Scientific Research in Computer Science and Engineering*, 8(1), pp. 07-15.

Sedik, R. R., 2023. *Linkedin*. [Online]
Available at: <https://www.linkedin.com/pulse/decoding-emotions-using-text-data-natural-language-roy-rachman-sedik/>
[Accessed 23 December 2024].

Sklearn, 2025. *Pipeline*. [Online]
Available at: <https://scikit-learn.org/1.5/modules/generated/sklearn.pipeline.Pipeline.html>
[Accessed 10 January 2025].

Taboada, M., 2016. Sentiment Analysis: An overview from Linguistics. *Annual Review of Applied Linguistics*, 2(1), pp. 325-347.

Vasista Reddy, 2018. *Sentiment Analysis using SVM*. [Online]
Available at: <https://medium.com/scrapehero/sentiment-analysis-using-svm-338d418e3ff1>
[Accessed 22 December 2024].

Wankhade, M., Rao, A. C. S. & Kulkarni, C., 2022. A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(2022), p. 5731–5780.

Wohlwend, B., 2023. *Classification Algorithms: KNN, Naive Bayes, and Logistic Regression*. [Online]
Available at: <https://medium.com/@brandon93.w/classification-algorithms-knn-naive-bayes-and-logistic-regression-515bdb085047>
[Accessed 22 December 2024].

Yun Xu, X. W. Q. W., 2015. *Sentiment Analysis of Yelp's Ratings Based on Text Reviews*, California: Stanford University.