BAHRAIN POLYTECHNIC بوليتكنك البحرين

Assessment Cover Sheet

Assessment	Project(Individual)		
Assessment	Uncontrolled	Individual	Not must-pass
Due Date	11 December 2025	**Course Code**	IT9002
Course Title	Natural Language Processing		
Internal Moderator's	Dr. Abdelhameed Fawzy		
External Examiner's	Dr.		

Instructions:

1. This cover sheet must be completed (section in blue below) and attached to your assessment before submission in hard copy/soft copy.
2. The time allowed for this assessment is XXX minutes/hours/days.
3. This assessment carries XXX marks distributed to a total of XXX questions assessing CILO X and CILO X.
4. The materials allowed for use in this assessment are XXX, XXX, and XXX.
5. The **use of generative AI tools is strictly prohibited**.
6. References consulted (if any) must be properly acknowledged and cited.
7. The assessment has a total of XXX pages

Learner ID	202508993	**Date Submitted**	06/01/2026
Learner Name	Hatem Isa		
Programme	MSc in AI		
Programme	IT9002		
Lecturer's	Sini Raj Pulari		

By submitting this assessment for marking, I affirm that this assessment is my own work.

Learner	Hatem Isa

Do not write beyond this line. For assessor use

Assessor's Name	Sini Raj Pulari		
Marking Date		**Maks Obtained**	

Comments:

Contents

Natural Language Processing Project: General Information

Project Individual Submission Date: **Thursday ,11th December 2025 by 11:55 p.m.**
Assessment Type: Individual Project

Learning Outcomes Assessed

The following learning outcomes will be assessed in this assessment:

Project (Individual) include Tasks 1, 2, 3,4,5,6

Learning Outcomes	**Tasks**
LO1 - Understanding the critical knowledge of fundamental concepts, algorithms, and models in Natural Language processing (NLP) for performing. various linguistic NLP tasks.	Task1, Task 2, Task 3, Task 4, Task 5
LO2 - Demonstrate professional levels of insight, interpretation by utilizing the various NLP packages like Natural Language Tool Kit (NLTK) to apply, solve, implement, evaluate, and improve the real time significant applications of NLP	Task 2, Task 3, Task 4, Task 5, Task 6

Project Instructions

Instructions, use of software and deliverables are given in the sub-sections below.

Log Files
This Project has a research component related to a selection of datasets.
Each week the student needs to update the log file in Moodle (Excel file) on a weekly basis. The log file should specify the following details:

Project Name:				
Student Name:				**Student1 ID**
IT9002 Project				
Date	**Week No**	**Task Name**	**Work done During the week (Bullet points /Description)**	**Issues Experienced if any**

Every week, **Saturday 11:59 pm**, will be the deadline for updating the Log files.

Use of Software
The project must be presented in a professional manner. All the models should be produced by using **either** the **Jupyter Notebook or Google Colaboratory**. It is the student's choice to select the software which you would prefer for your project.

Deliverables

The **deliverables for** of the project (Individual):

- The **project report** produced by using Microsoft Word
- **GitHub links** to the Project which contains. ipynb files and dataset etc.

Place the files into a **single ZIP file named** as **IT9002_Student_ID** and submit the ZIP file to Moodle for **Project** Submission.

Please note a major evaluation will be based on the report! Please make the report appropriate with all relevant screenshots with proper explanations. Simply putting lots of screenshot might add value to the work. No restriction for word limit. However, better is a short and crisp one approach detailing the inferences and jusrtifications wherever necessary.

PROJECT DESCRIPTION

The project description is for the Natural Language Processing [NLP], IT9002 course. The tasks below give more information about what you are expected to do in this project. This does not limit you to explore and come up with novel and creative steps in the project. However, as a general and standard guideline, every student should follow these steps.

There is no restriction to the upper limit of the word count (however, minimum is 200 words) or no visualizations or screenshots used (minimum is 2 per section). However, if you use a screenshot or a visualization, make sure to provide **title or description for the image. Also, if you need to highlight any information specifically, you may do so.**

1. Task 1 - Problem Statement Formulation and definition

You are required to research and find problems from any business domain, which will be suitable for applying the Text based NLP tasks. This assumes your understanding of a business problem that you have selected. You should select the business problem in such a way that you should be able to bring out some valuable insights as the result. Thoroughly study the existing works and scope of the project topic and then finalize the project topic. Finally, students are required to come out with a "Problem Statement / Problem Definition".
They should have a clear idea of what the motivations and objectives of the Problem statement they have chosen to work with.

To be precise,

- Motivation
- Problem Statement / Project Definition
- Expected Result - What are you trying to achieve

is required by the end of this task.

2. Task 2 - Selection of an Appropriate Data Set (Data Collection)

You are required to select a data set from the chosen business domain, which will be suitable for the use of classification or Prediction tasks. This assumes your understanding of the data used for the business domain chosen. The data set should have a complexity that will be expressed through a detailed dataset justification.

- Source of Dataset – reason why you chose the dataset
- Visualize the initial data to understand the distribution and summary of the data and display the initial data
- Also provide the "shape" of the dataset
- If the dataset have labels mention how many and which are they and visualize based on labels, if labels are categorical-changing them to numbers.

Note:

There are many websites / resources on the Internet providing free access to large numbers of data sets. Some of them are listed below. However, do not feel limited by the ones listed: if time allows, you can even collect the data yourself.

- https://data.world/datasets/nlp
- https://imerit.net/blog/25-best-nlp-datasets-for-machine-learning-all-pbm/
- https://www.kdnuggets.com/datasets/index.html
- https://odsc.medium.com/20-open-datasets-for-natural-language-processing-538fbfaf8e38
- https://www.kaggle.com/datasets?tags=13204-NLP
- https://www.exxactcorp.com/blog/Deep-Learning/best-open-source-datasets-for-nlp
- https://archive.ics.uci.edu/ml/index.php

Please use Visualizations wherever necessary!

3. Task 3 -Text Preprocessing

Text pre-processing is one of the most important phases in Natural Language Processing, because a good quality of data is the essence for building a good NLP model. Hence, you are expected to use at least 3 techniques in detail with the right import statements and show the appropriate output for each steps

- Tokenization
- Stemming
- Lemmatization
- TF-IDF methods (or any other methods)

- Please use Visualizations wherever necessary!

4. Task 4- Text Representation

Having a cleaned data, decide on the type of text representation techniques you may use for the business problem that you have selected. You may at least use 3 techniques in this section for the business problem you chose,

- POs Tagging
- Named Entity Recognition
- Bag of Words
- Skip Gram Models
- Word Embeddings
- N Grams

Please use Visualizations wherever necessary!

5. Task 5 -Text Classification / Prediction

Having converted the data into proper text representation, you are required to

- Select the most appropriate Text Classification /Prediction methods based on the business problem selected. Why these methods? Justify
- At least apply two models like ML based Models Like Naïve Bayes, Logistic , Multinomial Naïve Bases or DL based Models like RNN, LSTM, GRU or any Transformer Based Models on the cleaned and readied data. Please understand

you may use any two Classifier based models.

- While applying for the models, you may use NLTK, SPACY SKLEARN, NUMPY, Pandas, TensorFlow, and Keras etc. Students may decide and choose which package to use where while proceeding with the business process implementation.

Please use Visualizations wherever necessary!

6. <u>Task 6 - Evaluation , Inferences, Recommendation and Reflection</u>

- Evaluate and compare how well at least two models are performed.
- The results to be evaluated by various metrics like Accuracy, precision, recall, F1 measure or can use Confusion Matrix or any other metrics like AUC -ROC suitable for your problem. At least use two metrics to evaluate.
  Also inferences and reflections on learning and improvements you could have done on the project.

Please use Visualizations wherever necessary!

Project Report Contents

The project report should contain these main topics as follows in order.
<u>Same as the contents in Marking Structure and in the same order.</u>

Marking

The following lines emphasize some important points about the assessment and marking approach. This is **not a 'Must Pass'** assessment. The parts of the project subject to marking are listed in the table below.

Project Parts to be Marked	**Marks**
Cover Page	**1**
Table of Contents	**2**
Task 1 - Problem Statement Formulation and definition	**9**
Task 2 - Selection of an Appropriate Data Set (Data Collection)	**9**
Task 3 - Text Preprocessing	**15**
Task 4- Text Representation	**15**
Task 5 -Text Classification / Prediction	**25**
Task 6 -Evaluation, Inferences, Recommendation and Reflection	**12**
Overall Quality Report	**3**
Extra Challenging Problems	**1**
Log files	**5**
GitHub Link	**2**
References [Atleast 3]	**1**
Total:	**100**

<u>All the Best!</u>

<u>*End of the Document*</u>

JANUARY 6, 2026

SENTIMENT ANALYSIS OF E-COMMERCE PRODUCT REVIEWS

GITHUB REPOSITORY

NATURAL LANGUAGE PROCESSING – IT9002

HATEM ISA HATEM
202508993 – BAHRAIN POLYTECHNIC
Lecturer: Dr. Sini Raj Pulari

Declaration

I hereby declare that this project is original to me and has not been turned in for credit toward any other course. Every source that was used has been correctly credited and acknowledged.

Table of Contents

Table of Figures

Objective of the Project

The proposed project explores the application of Natural Language Processing (NLP) and classical machine learning models for sentiment classification in e-commerce product review. The main goal is not only to classify sentiments, but also to assess the effect of varying text preprocessing and feature representation methods on the model performance. This research targets to evaluate the trade-off between the simplicity of the model, its interpretability, and predictive power on sentiment analysis problems by comparing Logistic Regression and Multinomial Naive Bayes. The project also aims at contemplating real-life constraints including the imbalance of classes and noisy text data of the real world, which sheds some light on the weaknesses and strengths of classical NLP pipelines.

Problem Statement & Motivation

This project is an attempt to automatically classify e-commerce product reviews as negative or positive through the use of machine learning models and natural language processing (NLP) techniques.

Expected Results/Hypothesis

The expected outcome is a sentiment classification system that is able to reasonably predict customer sentiment and provide informative information regarding customer reviews.

Dataset Used

Reviews of Amazon e-commerce products from Kaggle form the dataset used in this project. In addition to product star ratings, it contains customer-written review content that is used to generate sentiment labels. One to two-star reviews are classified as negative, while four to five-star reviews are classified as good. In order to maintain binary sentiment categorization, neutral reviews are removed. The data is already categorized and structured but still contains natural language noise, such as informal wording and spelling errors, which makes it semi-clean. In addition to it being in CSV format, which makes it easier to load and process in Google Colab.

Methods Utilized in the Project

In this project, several NLP techniques were used. Tokenization, stopword removal, and stemming were among the text preparation techniques, which assist in noise reduction and text standardization. Text was transformed into numerical features for text representation using Bag of Words, N-grams, and TF-IDF. Two models were deployed for sentiment analysis: Logistic Regression and Multinomial Naive Bayes. As for the model's performance, this project focused on accuracy, recall, F1-score, precision and confusion matrices.

Findings

The findings demonstrate that, with the right preprocessing, machine learning models can successfully classify sentiment in product reviews. Overall, Logistic Regression outscored

Multinomial Naive Bayes. Additionally, experiments showed that by generating more relevant feature representations, appropriate text preprocessing increases classification accuracy.

Limitations

Class imbalance in the dataset, which made it more difficult for the model to accurately forecast the minority sentiment class, is one of the project's limitations. Additionally, as deep learning techniques were outside the scope of this study, only conventional machine learning models were employed. Furthermore, neutral or mixed sentiments are not included in the analysis; it only concentrates on binary sentiment classification.

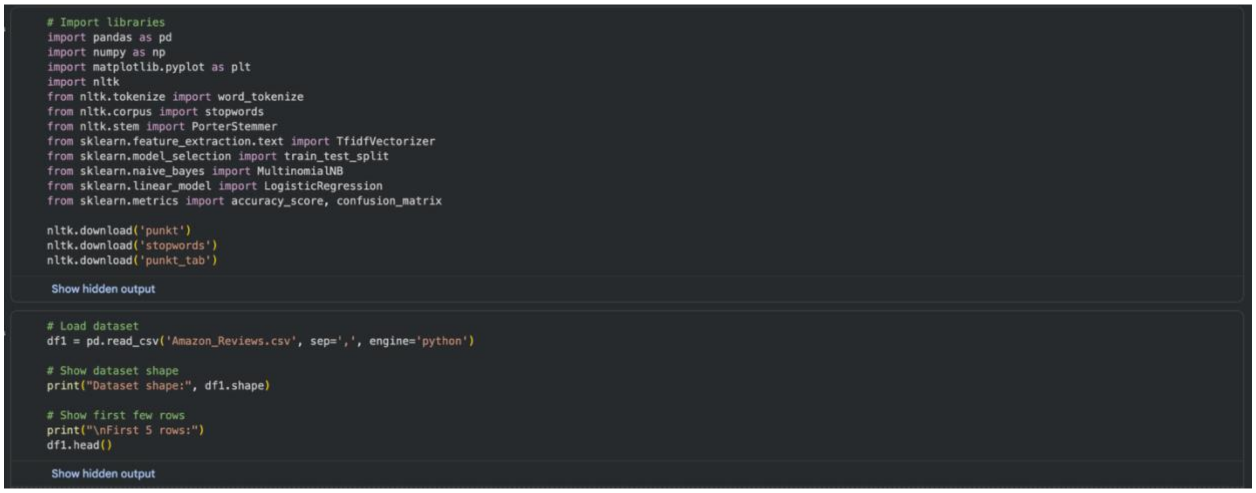
Literature Review

In e-commerce, sentiment analysis is frequently used to examine customer feedback and opinions (Zhang et al., 2020). According to earlier research, text preprocessing and feature representation are crucial for enhancing sentiment classification and analysis performance (Jiang et al., 2021). Additionally, studies demonstrate that when paired with appropriate preprocessing techniques, common machine learning models can still produce impressive outcomes (Minaee et al., 2021). Sentiment analysis tasks can be implemented with easily accessible capabilities from NLP libraries like NLTK (Bird et al., 2020).

Natural Language Processing

Text preprocessing was used to lower noise and enhance data quality, through Tokenization, stopword removal, stemming, and TF-IDF feature extraction. These steps enhanced model performance and helped standardize the content.

The first step is to import all the necessary Python libraries for data handling, text preprocessing, model training, and evaluation as seen in Figure 1 below. Next, a CSV file called Amazon_Reviews.csv that was manually uploaded to Google Colab is used to load the dataset. Before further processing, the form of the dataset and the first few rows are shown to help understand its structure.



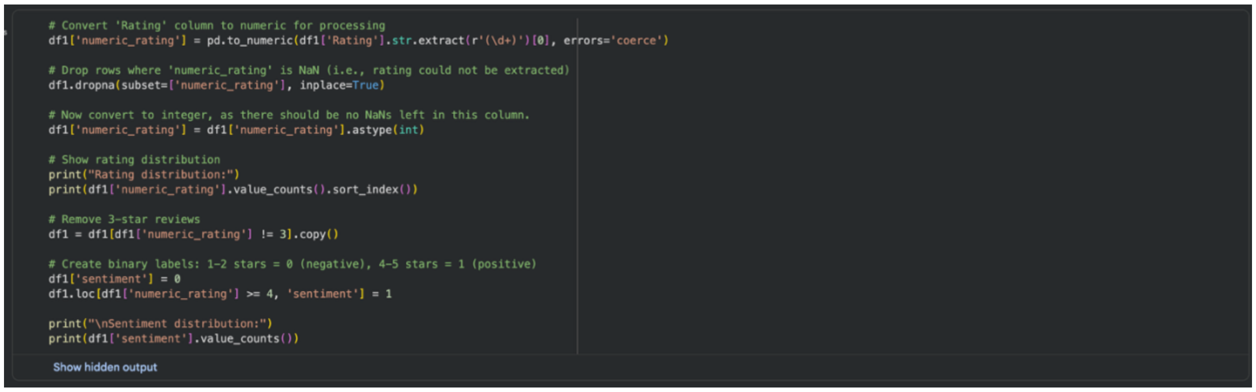
```python
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')
```

```python
# Load dataset
df1 = pd.read_csv('Amazon_Reviews.csv', sep=',', engine='python')

# Show dataset shape
print("Dataset shape:", df1.shape)

# Show first few rows
print("\nFirst 5 rows:")
df1.head()
```

Figure 1 - Initial Step of Installing Python Libraries & Uploading the Dataset

The second step as seen in Figure 2 below, prepares the rating values for sentiment analysis. The Rating column's numerical component has to be extracted out and then transformed into numbers. Rows with ratings that cannot be converted are eliminated and removed, this way ratings were converted to numerical values and mapped to binary sentiment labels.

The distribution of ratings is then printed to determine how many reviews correspond to each rating value. All 3-star reviews are then eliminated since they are considered neutral. Lastly, ratings of 1-2 are labeled as negative (0) and ratings of 4-5 as positive (1) in a newly constructed sentiment column. Since we converted to 1's and 0's it means we reached a binary state, we then finally display & print the sentiment labels.



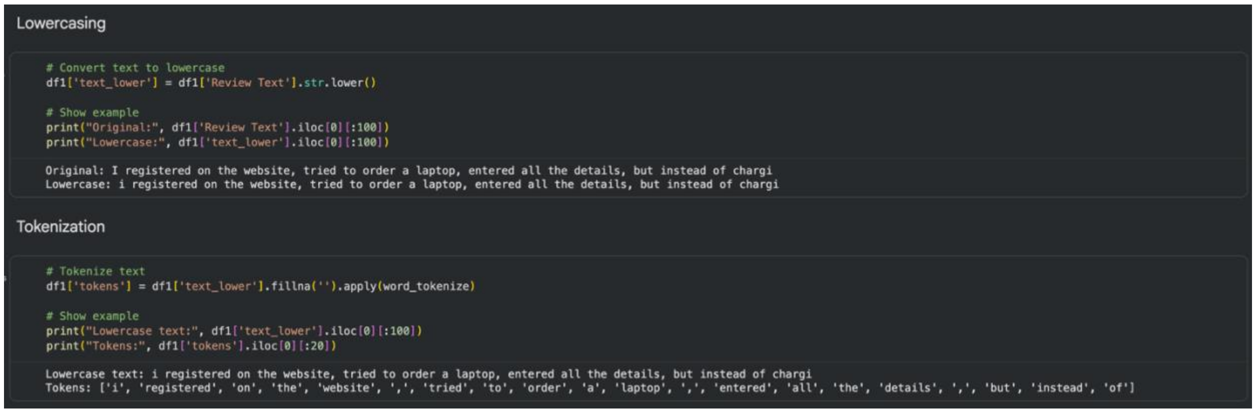
```
# Convert 'Rating' column to numeric for processing
df1['numeric_rating'] = pd.to_numeric(df1['Rating'].str.extract(r'(\d+)')[0], errors='coerce')

# Drop rows where 'numeric_rating' is NaN (i.e., rating could not be extracted)
df1.dropna(subset=['numeric_rating'], inplace=True)

# Now convert to integer, as there should be no NaNs left in this column.
df1['numeric_rating'] = df1['numeric_rating'].astype(int)

# Show rating distribution
print("Rating distribution:")
print(df1['numeric_rating'].value_counts().sort_index())

# Remove 3-star reviews
df1 = df1[df1['numeric_rating'] != 3].copy()

# Create binary labels: 1-2 stars = 0 (negative), 4-5 stars = 1 (positive)
df1['sentiment'] = 0
df1.loc[df1['numeric_rating'] >= 4, 'sentiment'] = 1

print("\nSentiment distribution:")
print(df1['sentiment'].value_counts())
```

Figure 2 - Sentiment label distribution following the elimination of neutral reviews

Text Preprocessing

In this step, we start with the Lowercasing technique. In order to treat words, like "good" and "good" as the same word, all letters in the review text are changed to lowercase. To display both the original text and the lowercase version, an example is displayed.

Tokenization is then used. NLTK's tokenizer divides the lowercase text into discrete words, called tokens. This turns a full phrase into a list of words, which makes the text easier to handle in later processes such as stopword removal and feature extraction. To demonstrate the result, an example of the tokens is printed as seen in Figure 3 below.


Lowercasing

```
# Convert text to lowercase
df1['text_lower'] = df1['Review Text'].str.lower()

# Show example
print("Original:", df1['Review Text'].iloc[0][:100])
print("Lowercase:", df1['text_lower'].iloc[0][:100])
```

```
Original: I registered on the website, tried to order a laptop, entered all the details, but instead of chargi
Lowercase: i registered on the website, tried to order a laptop, entered all the details, but instead of chargi
```

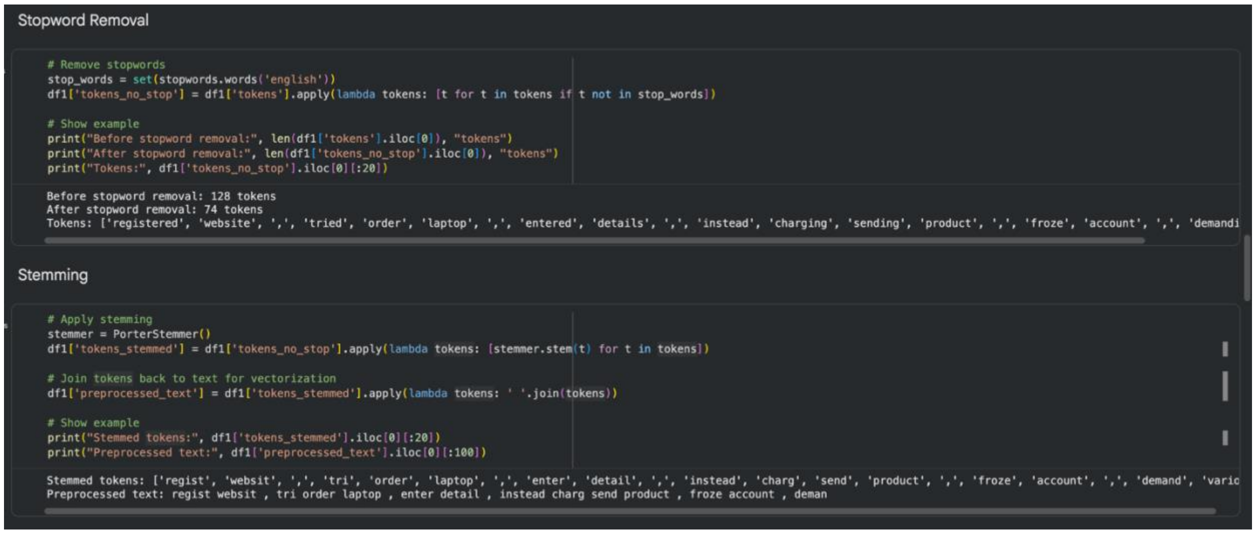
Tokenization

```
# Tokenize text
df1['tokens'] = df1['text_lower'].fillna('').apply(word_tokenize)

# Show example
print("Lowercase text:", df1['text_lower'].iloc[0][:100])
print("Tokens:", df1['tokens'].iloc[0][:20])
```

```
Lowercase text: i registered on the website, tried to order a laptop, entered all the details, but instead of chargi
Tokens: ['i', 'registered', 'on', 'the', 'website', ',', 'tried', 'to', 'order', 'a', 'laptop', ',', 'entered', 'all', 'the', 'details', ',', 'but', 'instead', 'of']
```

Figure 3 - An example of tokenization and lowercasing used in a customer review

To get rid of common words like and and that don't add meaning, stopwords are eliminated from the tokenized text in this step. As a result, the data is less cluttered. Stemming is then used to reduce words to their most basic form. In order to use them for feature extraction in the following step, the processed tokens are then linked back into text.


Stopword Removal

```
# Remove stopwords
stop_words = set(stopwords.words('english'))
df1['tokens_no_stop'] = df1['tokens'].apply(lambda tokens: [t for t in tokens if t not in stop_words])

# Show example
print("Before stopword removal:", len(df1['tokens'].iloc[0]), "tokens")
print("After stopword removal:", len(df1['tokens_no_stop'].iloc[0]), "tokens")
print("Tokens:", df1['tokens_no_stop'].iloc[0][:20])
```

```
Before stopword removal: 128 tokens
After stopword removal: 74 tokens
Tokens: ['registered', 'website', ',', 'tried', 'order', 'laptop', ',', 'entered', 'details', ',', 'instead', 'charging', 'sending', 'product', ',', 'froze', 'account', ',', 'demandi
```

Stemming

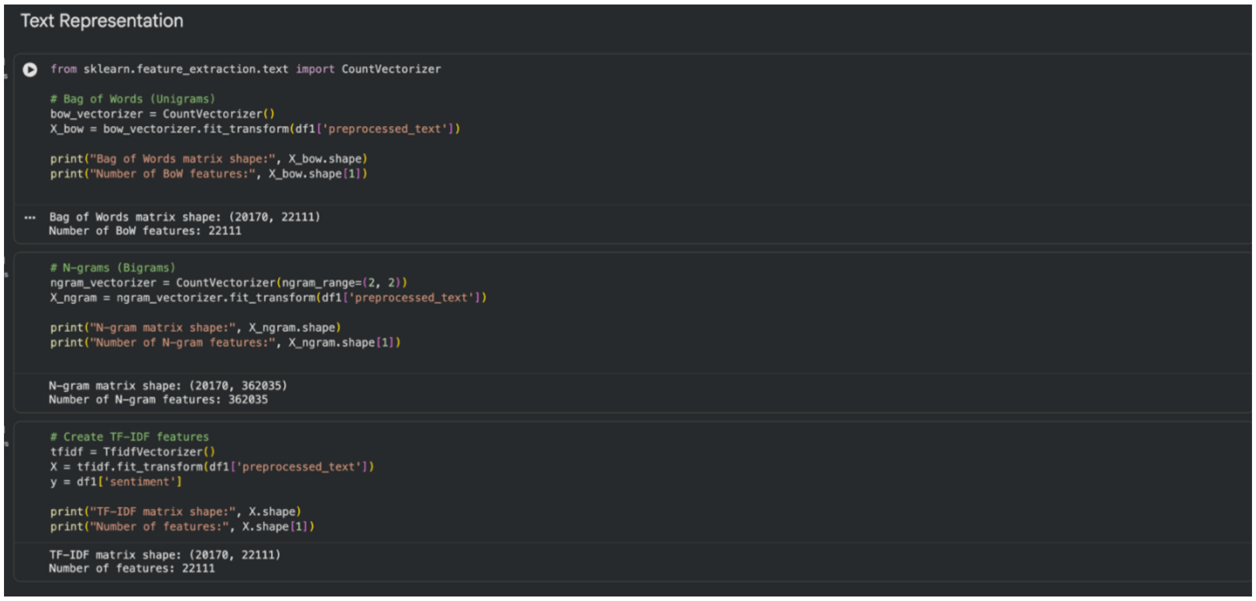
```
# Apply stemming
stemmer = PorterStemmer()
df1['tokens_stemmed'] = df1['tokens_no_stop'].apply(lambda tokens: [stemmer.stem(t) for t in tokens])

# Join tokens back to text for vectorization
df1['preprocessed_text'] = df1['tokens_stemmed'].apply(lambda tokens: ' '.join(tokens))

# Show example
print("Stemmed tokens:", df1['tokens_stemmed'].iloc[0][:20])
print("Preprocessed text:", df1['preprocessed_text'].iloc[0][:100])
```

```
Stemmed tokens: ['regist', 'websit', ',', 'tri', 'order', 'laptop', ',', 'enter', 'detail', ',', 'instead', 'charg', 'send', 'product', ',', 'froze', 'account', ',', 'demand', 'vario
Preprocessed text: regist websit , tri order laptop , enter detail , instead charg send product , froze account , deman
```

Figure 4 - Applying stemming and stopword removal to an example customer review

Text Representation & Classification

During this step, the cleaned reviews will be converted into numbers, which can be interpreted by machine learning models with the help of different text representation methods. The first one is the Bag of Words (BoW) approach. It gives a plain description of the text and calculates how many times a certain word can be found in a review. N-grams (bigrams) are then used to capture pairs of words and not single words. This helps the model to understand simple word order and short phrases. Finally, the most important feature method of training the models is TF-IDF. Words that occur frequently in one of the reviews and rarely in the rest are given more weight by TF-IDF. The data is then split into training and testing set in an 80:20 ratio in order to evaluate the performance of the model.



Text Representation

```
from sklearn.feature_extraction.text import CountVectorizer

# Bag of Words (Unigrams)
bow_vectorizer = CountVectorizer()
X_bow = bow_vectorizer.fit_transform(df1['preprocessed_text'])

print("Bag of Words matrix shape:", X_bow.shape)
print("Number of BoW features:", X_bow.shape[1])
```

```
Bag of Words matrix shape: (20170, 22111)
Number of BoW features: 22111
```

```
# N-grams (Bigrams)
ngram_vectorizer = CountVectorizer(ngram_range=(2, 2))
X_ngram = ngram_vectorizer.fit_transform(df1['preprocessed_text'])

print("N-gram matrix shape:", X_ngram.shape)
print("Number of N-gram features:", X_ngram.shape[1])
```

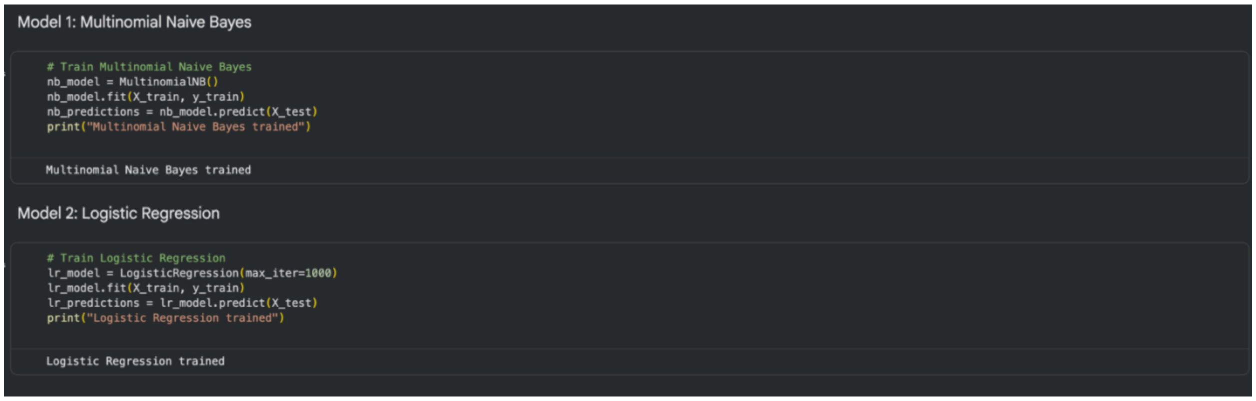
```
N-gram matrix shape: (20170, 362035)
Number of N-gram features: 362035
```

```
# Create TF-IDF features
tfidf = TfidfVectorizer()
X = tfidf.fit_transform(df1['preprocessed_text'])
y = df1['sentiment']

print("TF-IDF matrix shape:", X.shape)
print("Number of features:", X.shape[1])
```

```
TF-IDF matrix shape: (20170, 22111)
Number of features: 22111
```

Figure 5 - N-grams, TF-IDF feature representation, and bag of words with train-test data splitting

Two machine learning models are trained for sentiment classification in this step. First, the training data and the TF-IDF features are used to train a Multinomial Naive Bayes model. On the test set, the model then generates predictions. The same training data is then used to train a Logistic Regression model. This enables a straightforward comparison of two popular text data classification models. In the following stage, both models are trained independently and ready for assessment.



Model 1: Multinomial Naive Bayes

```
# Train Multinomial Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_predictions = nb_model.predict(X_test)
print("Multinomial Naive Bayes trained")
```

```
Multinomial Naive Bayes trained
```

Model 2: Logistic Regression

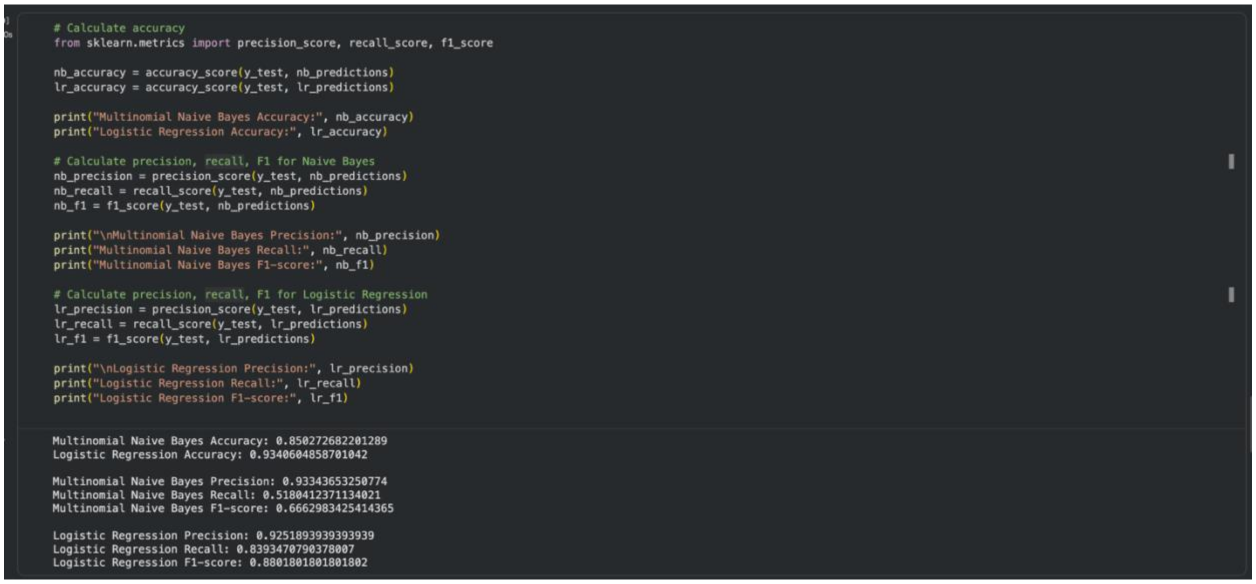
```
# Train Logistic Regression
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
print("Logistic Regression trained")
```

```
Logistic Regression trained
```

Figure 6 - Multinomial Naive Bayes and Logistic Regression model training for sentiment analysis

Evaluation & Reflection

As seen in Figures 7, 8, and 9, accuracy ratings and confusion matrices were used to assess the model's performance. The accuracy results for both models are shown in Figure 7, with Logistic Regression outperforming Multinomial Naive Bayes. The confusion matrices for Naive Bayes and Logistic Regression are displayed in Figures 8 and 9, showing the number of reviews that were correctly and incorrectly classified.



```python
# Calculate accuracy
from sklearn.metrics import precision_score, recall_score, f1_score

nb_accuracy = accuracy_score(y_test, nb_predictions)
lr_accuracy = accuracy_score(y_test, lr_predictions)

print("Multinomial Naive Bayes Accuracy:", nb_accuracy)
print("Logistic Regression Accuracy:", lr_accuracy)

# Calculate precision, recall, F1 for Naive Bayes
nb_precision = precision_score(y_test, nb_predictions)
nb_recall = recall_score(y_test, nb_predictions)
nb_f1 = f1_score(y_test, nb_predictions)

print("\nMultinomial Naive Bayes Precision:", nb_precision)
print("Multinomial Naive Bayes Recall:", nb_recall)
print("Multinomial Naive Bayes F1-score:", nb_f1)

# Calculate precision, recall, F1 for Logistic Regression
lr_precision = precision_score(y_test, lr_predictions)
lr_recall = recall_score(y_test, lr_predictions)
lr_f1 = f1_score(y_test, lr_predictions)

print("\nLogistic Regression Precision:", lr_precision)
print("Logistic Regression Recall:", lr_recall)
print("Logistic Regression F1-score:", lr_f1)
```

```
Multinomial Naive Bayes Accuracy: 0.850272682201289
Logistic Regression Accuracy: 0.9340604858701042

Multinomial Naive Bayes Precision: 0.93343653250774
Multinomial Naive Bayes Recall: 0.5180412371134021
Multinomial Naive Bayes F1-score: 0.6662983425414365

Logistic Regression Precision: 0.9251893939393939
Logistic Regression Recall: 0.8393470790378007
Logistic Regression F1-score: 0.8801801801801802
```

Figure 7 - F1-score, recall, accuracy, and precision for logistic regression and multinomial naive bayes models

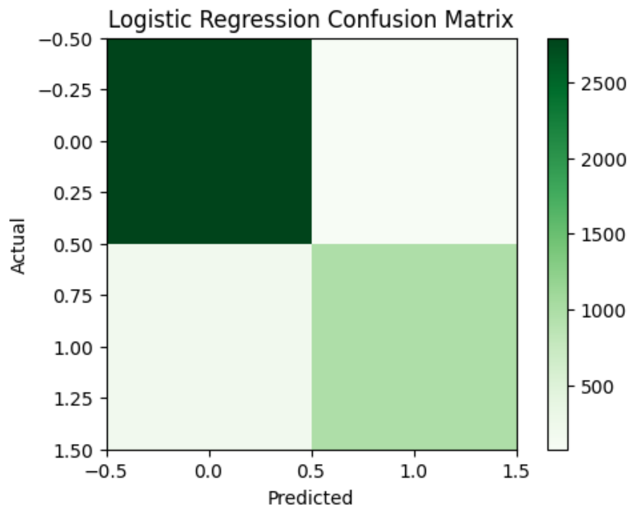


Figure 8 - Confusion matrix for the sentiment classification model using logistic regression

According to the results, Logistic Regression did better overall and had fewer misclassifications. I gained a better understanding of the significance of TF-IDF feature representation and text preparation in sentiment analysis thanks to this project. Further techniques to address class imbalance and enhance model performance could be investigated in future research.

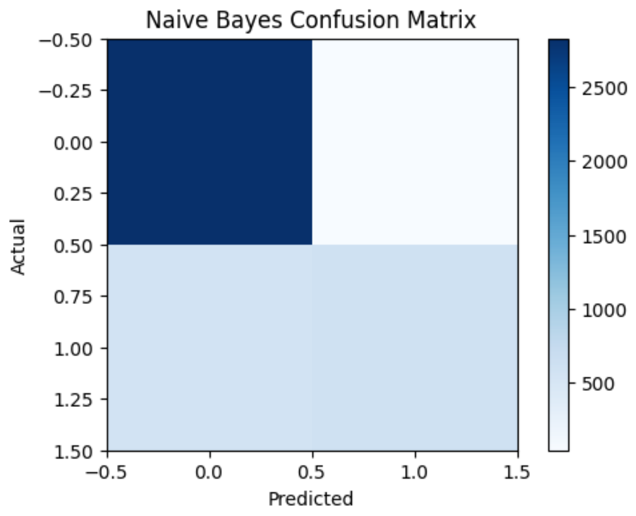


Figure 9 - Confusion matrix for the sentiment classification model using Naive Bayes

This project strengthened the need to match the preprocessing methods with the learning assumptions of the selected model. The comparative findings indicated that more complex feature representations do not imply more complex models to be used to attain high performance. Another learning outcome of this exercise was the practical challenges encountered when handling real-world textual data, such as noise, imbalance, and ambiguity in sentiment expression. These lessons can be useful in the application of NLP methods in real-world e-commerce settings where interpretability and efficiency may hold equal importance to accuracy.

Conclusion

This project shows how sentiment analysis in e-commerce product reviews may be done using NLP techniques. Customer sentiment was successfully determined using machine learning classification, feature extraction, and text preprocessing. The project gave me hands-on experience with NLP ideas I have studied in class.

References

The Jupyter Notebook and dataset, together with the entire project implementation, are accessible on GitHub. The sentiment analysis notebook, the Amazon reviews dataset, and a README file outlining the project's structure and technique are all included in the repository.

GitHub Repository:

https://github.com/hatemmsc-ops/Hatemlsa_202508993_IT9002_NLP_Project

- Bird, S., Klein, E., & Loper, E. (2020). *Natural language processing with Python* (2nd ed.). O'Reilly Media.
- Jiang, Y., Li, S., & Wang, R. (2021). Deep learning for text classification. *Information Processing & Management, 58*(2).
- Minaee, S., et al. (2021). Deep learning–based text classification: A comprehensive review. *ACM Computing Surveys, 54*(3).
- Zhang, L., Wang, S., & Liu, B. (2020). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10*(2).
- Kaggle. (2023). *Amazon product reviews dataset*. https://www.kaggle.com
- OpenAI. (2024). *ChatGPT* [Large language model]. https://www.openai.com

Only language improvement and writing clarity were supported by generative artificial intelligence (AI) methods. The student produced all of the analysis, code implementation, findings, and interpretations.