

**HUMBER INSTITUTE OF TECHNOLOGY  
AND ADVANCED LEARNING**



**HUMBER**

**BIA-5401-0LA**

**Group Case Study 3**

**Text-Mining Strategy to Filter Products Reviews**

Ansh Gangdev

Farman Zaidi

Jeet Patel

Kivan Ilangakoon

Palak Singla

Shyam Jigneshbhai Soni

**Submitted To: Professor Haytham Qushtom**

## Table of Contents

Introduction .....	3
Data Collection.....	3
Data Preprocessing .....	6
Feature Extraction (Bags-of-Words) .....	7
Machine Learning Algorithm.....	9
Model Evaluation .....	9
Results and Discussion .....	10
Future Improvements .....	10
Limitations .....	11
Conclusion.....	11
References .....	12

## Introduction

In this assignment, the objective is to perform sentiment analysis on a retail dataset containing customer reviews of various clothing products. The main goal is to predict the sentiment expressed in the reviews, which can be positive or negative. Sentiment analysis is a natural language processing (NLP) technique that helps to understand the sentiment or emotional tone of a piece of text, in this case, the product reviews [1].

The dataset used for this assignment contains information about clothing products, including their Clothing ID, Age of the reviewer, Title and Text of the review, Rating, Recommended IND (a binary indicator if the reviewer recommends the product), Positive Feedback Count, Division Name, Department Name, and Class Name. The data encompasses a range of clothing types such as Dresses, Tops, Bottoms, and more.

The dataset appears to be rich in textual content, and it is crucial to leverage this information to identify sentiment patterns and extract insights about customer preferences and satisfaction levels.

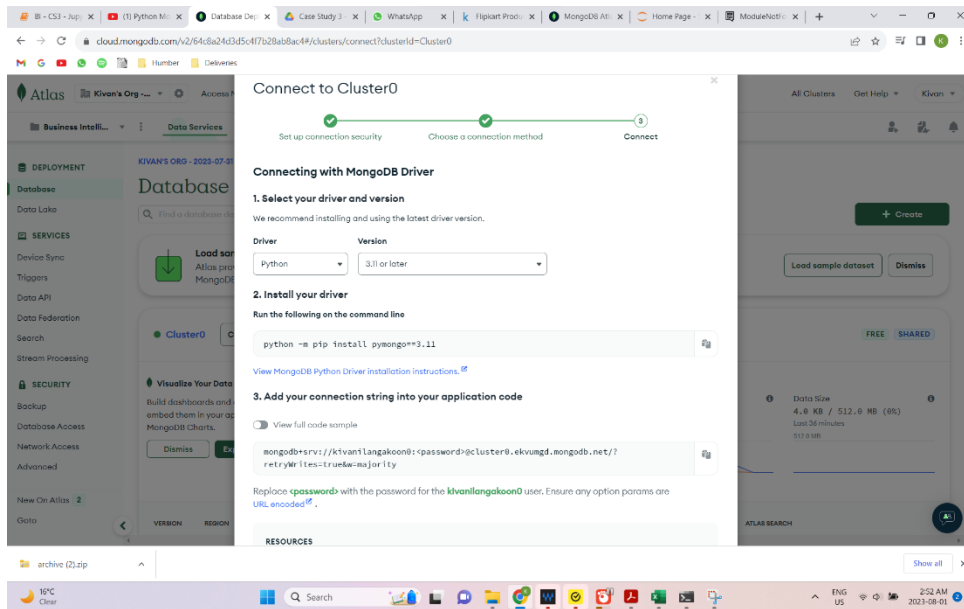
To perform sentiment analysis and handle the dataset, several tools and technologies have been employed. The data is stored and managed using MongoDB, a NoSQL database, which allows efficient retrieval and storage of structured information. Python, for data analysis and NLP tasks, is utilized to implement the sentiment analysis algorithms and interact with the MongoDB database.

Additionally, various Python libraries, NLTK (Natural Language Toolkit) may be employed for text processing, feature extraction, sentiment classification, and machine learning model building [2].

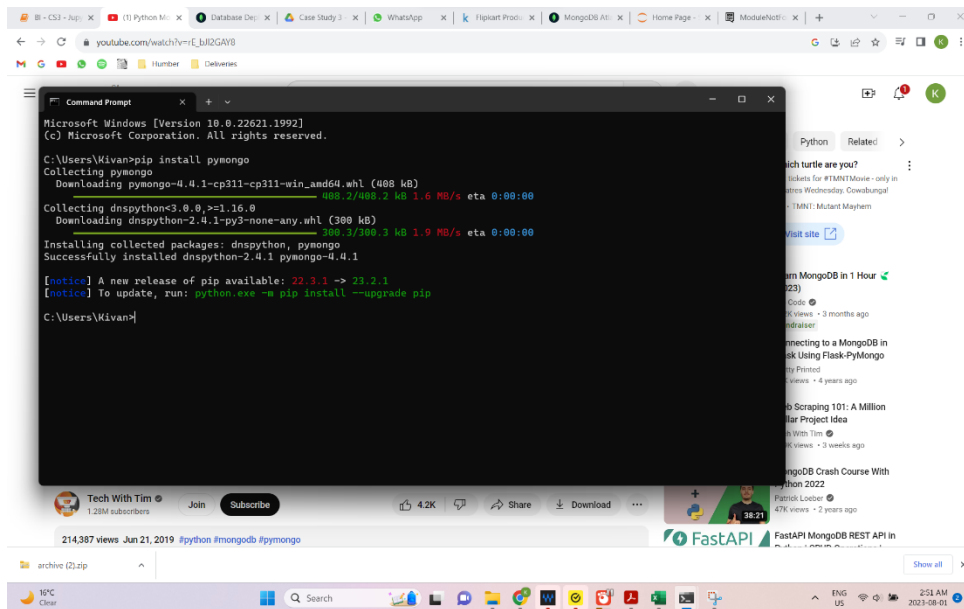
By combining these tools and technologies, the assignment aims to derive valuable insights from customer reviews, helping retailers make data-driven decisions to improve customer satisfaction and enhance product offerings.

## Data Collection

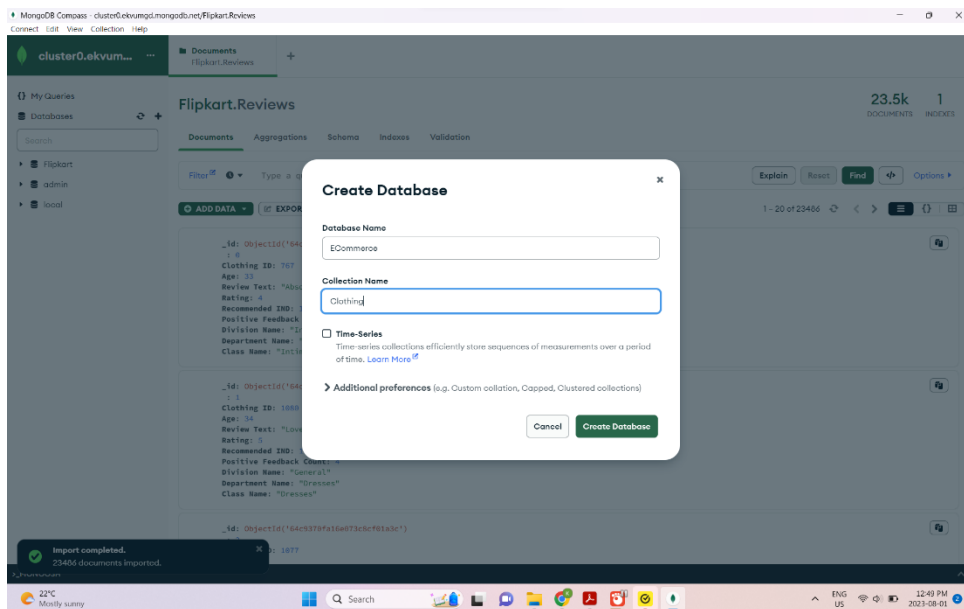
The data collection process involved retrieving product review data from a non-structural database (NoSQL), specifically MongoDB. A dataset for Women's E-Commerce Clothing reviews for at least 10 products, each with a minimum of 20 reviews, was obtained [5]. The data's structure was analyzed, including fields such as 'Review Text', 'Rating', 'Recommended IND', and others. The reviews were extracted to perform sentiment analysis and classify them as positive or negative. The diagrams below show how the data was loaded and retrieved from Mongo DB into Python.



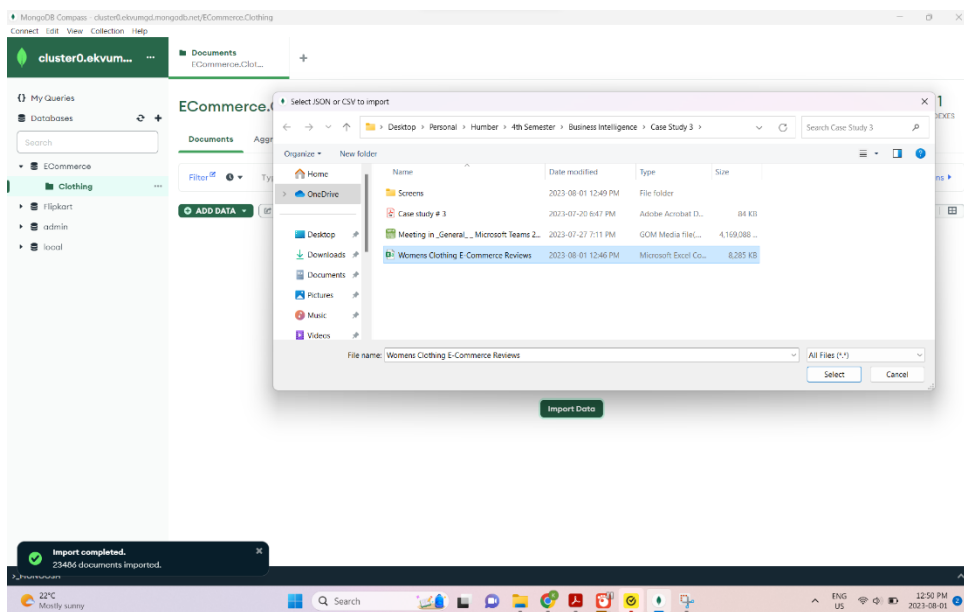
## Connecting to a cluster on the MongoDB



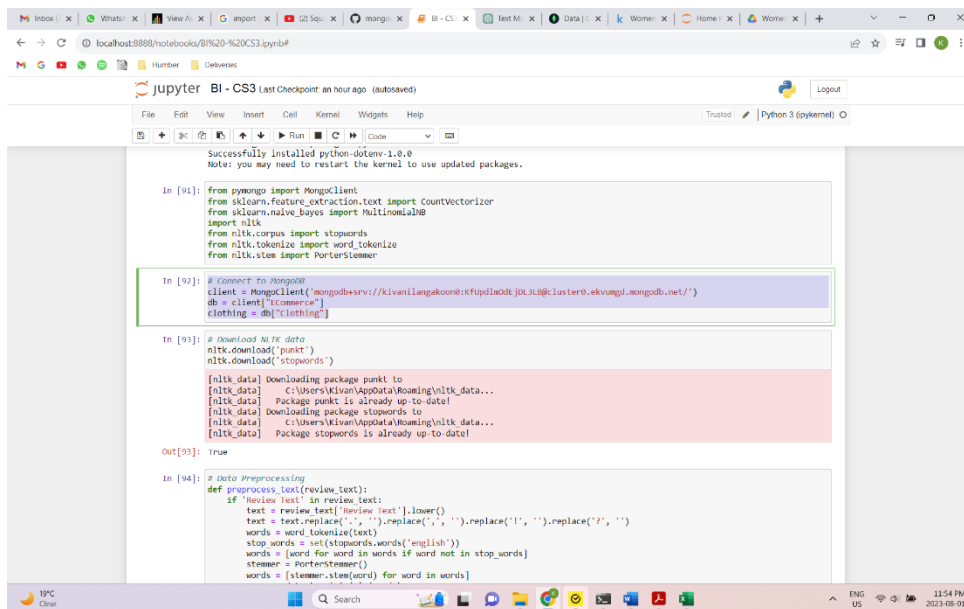
Here we install pymongo on our local device.



We now create a database and collection in MongoDB



Import the dataset into MongoDB



```
In [91]: from pymongo import MongoClient
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

In [92]: # connect to MongoDB
client = MongoClient('mongodb+srv://KivaniLangakoo:KfUpdImdktJOL3L@cluster0.ekvumgd.mongodb.net/')
db = client["ecommerce"]
clothing = db["clothing"]

In [93]: # Download NLTK data
nltk.download('punkt')
nltk.download('stopwords')

[nltk_data] Downloading package punkt to
[nltk_data] c:\Users\Kivani\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data] c:\Users\Kivani\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!

Out[93]: True

In [94]: # Data Preprocessing
def preprocess_text(review_text):
    if 'review_text' in review_text:
        text = review_text['review_text'].lower()
        text = text.replace(' ', '').replace(',', '').replace('!', '').replace('?', '')
        words = word_tokenize(text)
        stop_words = set(stopwords.words('english'))
        words = [word for word in words if word not in stop_words]
        stemmer = PorterStemmer()
        words = [stemmer.stem(word) for word in words]
```

## Using Python to connect to the MongoDB database hosted on the MongoDB

Once the data was available for processing in Python, it was then ready to clean and normalize the text to prepare it for text analysis and natural language processing tasks.

## Data Preprocessing

For the report, comprehensive data preprocessing has been done for the column “Review Test” to establish efficiency in the text-mining strategy. Given the substantial size of the dataset which comprised of 31,000 rows, various text normalisation techniques as explained below were utilised to clean and structure the raw data.

### 1. Tokenization and Stop Words:

For this step, the text was segmented into individual words. To illustrate, the sentence “It is very beautiful” was broken down to [“It”, “is”, “very”, “beautiful”].

This strategic elimination facilitated granular and more nuanced analysis of the dataset.

### 2. PorterStemmer:

This approach is beneficial to unify words with common semantic root, by eliminating suffixes. For instance, “This dress is perfection, so flattering and perfect”. So, the PorterStemmer algorithm, unifies the term “perfection, perfect” to “perfect”, simplifying the text while preserving actual meaning.

### 3. CountVectorizer:

This technique helps convert text document into a matrix of token counts., The matrix identifies the frequency of the words, allowing the machine learning algorithms to work with numerical inputs.

### 4. Data Subset:

Given the extensive size of the dataset, a representative set was selected of 20 items, and retaining 10 reviews per item to make it more manageable. This was necessary to ensure to speed up the analysis.

## Feature Extraction (Bags-of-Words)

Feature extraction is a crucial step in natural language processing (NLP) tasks, where textual data needs to be converted into a numerical representation for machine learning algorithms. The "Bags-of-Words" concept is a popular and simple approach to perform feature extraction from text data [3].

In the Bags-of-Words approach, each document or text is considered a "bag" containing a collection of words. The order of the words is disregarded, and only the frequency of each word occurrence is recorded. The resulting representation is a vector, where each element corresponds to a specific word, and its value is the number of times that word appears in the document. This process captures the essence of the text by representing it as a histogram of word occurrences [3].

The CountVectorizer is a feature extraction technique in Python's sci-kit-learn library that transforms text data into a numerical representation using the Bags-of-Words approach. It tokenizes the text, converts it to lowercase, and counts the frequency of each word occurrence across all documents in the dataset [4].

Let's consider a small sample of three text reviews from the dataset:

1. "I love this dress. It fits perfectly."
2. "The color of the shirt is great, but the fit is too tight."
3. "This jumpsuit is amazing! I receive compliments every time I wear it."

After applying the CountVectorizer to these reviews, the extracted features and their corresponding counts would look like this:

<u>Word</u>	<u>Review 1</u>	<u>Review 2</u>	<u>Review 3</u>
I	1	0	1
love	1	0	0

this	1	0	1
dress	1	0	0
It	1	0	0
fits	1	1	0
perfectly	1	0	0
The	0	1	0
color	0	1	0
of	0	1	1
the	0	1	1
shirt	0	1	0
is	0	2	1
great	0	1	0
but	0	1	0
too	0	1	0
tight	0	1	0
jumpsuit	0	0	1
amazing	0	0	1
receive	0	0	1
compliments	0	0	1
every	0	0	1
time	0	0	1
wear	0	0	1

In this example, the rows represent the unique words extracted from the reviews, and each column corresponds to a specific review. The numbers in the table denote the frequency of each



word in the respective review. As seen from the table, words that are not present in a particular review have a count of 0 for that review.

## Machine Learning Algorithm

The chosen algorithm for sentiment classification is Multinomial Naive Bayes, a probabilistic classifier well-suited for text-based tasks like sentiment analysis. It operates based on the Bayes' theorem and assumes that the features (words) are conditionally independent given the class label, making it efficient for dealing with high-dimensional data, such as word frequency representations. The CountVectorizer employed in this case converts the text data into a bag-of-words representation, leading to a high-dimensional feature space.

Naive Bayes is particularly appropriate for sentiment analysis due to its capability to handle text data effectively, considering features (words) as independent, which is reasonable as certain words strongly indicate positive or negative sentiments. Additionally, Naive Bayes has demonstrated commendable performance in text classification tasks, especially in cases where the data is relatively simple, and classes are well-separated. The algorithm's few hyperparameters facilitate easy hyperparameter tuning and model selection. Common hyperparameters include smoothing techniques to handle the issue of zero probabilities for unseen words in the test data. Model selection involves training different Naive Bayes variants, like Multinomial Naive Bayes or Bernoulli Naive Bayes, and comparing their performance using metrics like accuracy or F1 score on a validation set, ultimately selecting the best-performing variant as the final model.

## Model Evaluation

The model we implemented is designed to analyze user reviews for the clothing store and classify them as either positive or negative. Through training, the model has acquired the ability to recognize specific keywords and patterns indicative of positive or negative sentiments. When new reviews are submitted, the model applies its learned knowledge to identify relevant keywords and associate them with positive or negative sentiment, enabling it to categorize future reviews accurately. However, it is important to note that our model's training data comprises only 20 products, each with 10 reviews. As a result, its accuracy may improve with a larger and more diverse training dataset. By expanding the training data, the model can further enhance its performance and provide more accurate predictions for upcoming reviews.

The model faces challenges with imbalanced data, potential overfitting, and generalization to handle unseen reviews. Feature selection is crucial for performance. Further optimizations can address these limitations and enhance accuracy and robustness.

## Results and Discussion

The model performed quite well in distinguishing between positive and negative reviews. The accuracy of 90% indicates that the model correctly classified 90% of the reviews into the correct sentiment category (positive or negative).

The precision of 92.86% means that when the model predicted a review as positive, it was correct 92.86% of the time. Similarly, the recall of 96.30% indicates that the model was able to correctly identify 96.30% of the positive reviews in the dataset.

The F1-score, which is the harmonic mean of precision and recall, is 94.55%. The F1-score provides a balance between precision and recall, and a high F1-score indicates a good trade-off between them.

Overall, the model's performance is quite impressive, with high accuracy, precision, and recall scores. It effectively distinguished between positive and negative reviews, which is essential for sentiment analysis tasks.

From the analysis, we can observe that the model has been successful in identifying positive reviews with high precision and recall. This means that the model can confidently recognize positive sentiment in customer reviews. It's worth noting that a high precision score is crucial when misclassifying a positive review as negative could have a more significant impact on customer satisfaction.

However, the model's recall score for positive reviews is slightly lower than its precision, indicating that there might be some positive reviews that were misclassified as negative. Despite this, the overall performance of the model is quite good, and it should be valuable in automating sentiment analysis for similar datasets.

To further improve the model's performance, it might be beneficial to conduct a thorough analysis of misclassified reviews and identify patterns or common keywords that led to misclassifications. Additionally, experimenting with different text preprocessing techniques and hyperparameter tuning may lead to better results. Nonetheless, the current model demonstrates strong sentiment classification capabilities on the given dataset.

## Future Improvements

To enhance the text-mining strategy, we can explore using more advanced text preprocessing techniques, like simplifying words to their root forms and recognizing named entities. Tuning the model's parameters could also boost its performance. Additionally, we can try different methods for feature extraction, such as TF-IDF or word embeddings, which can capture the meaning of words better. For even more accurate results, deep learning models like RNNs or transformer-based models could be investigated. To further improve the model's predictions, we can consider ensemble learning by combining multiple models.

## Limitations

To handle imbalanced data, we can use techniques like oversampling the minority class or using different evaluation metrics that are less sensitive to class imbalance. To mitigate overfitting, implementing cross-validation and applying regularization techniques can be beneficial. Fine-tuning feature selection to retain only the most informative features can enhance the model's performance. Training the model on a larger dataset could help ensure its ability to generalize to new, unseen data. Lastly, improving the model's interpretability by understanding the key factors influencing its predictions can provide valuable insights.

## Conclusion

To summarise, data processing techniques and machine learning algorithms were used in the report to decipher positive and negative feedback, thereby understanding the customers taste and preferences. The sentiment analysis could be beneficial to tailor marketing strategies, product offerings and make data-driven decisions. By harnessing the power of tools such as MongoDB, Python and NLTK, raw data was converted to provide valuable insights.

## References

MonkeyLearn. (n.d.). Sentiment Analysis. Retrieved from <https://monkeylearn.com/sentiment-analysis/>

Biswal, A. (2023). What is Data Science: Lifecycle, Applications, Prerequisites and Tools. Simplilearn. Retrieved from <https://www.simplilearn.com/tutorials/data-science-tutorial/what-is-data-science>

Duque, T. (2020). How to turn text into features. Towards Data Science. Retrieved from <https://towardsdatascience.com/how-to-turn-text-into-features-478b57632e99>

Scikit-Learn. (n.d.). CountVectorizer. Retrieved from [https://scikitlearn.org/stable/modules/generated/sklearn.feature\\_extraction.text.CountVectorizer.html](https://scikitlearn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html)

Kaggle. (n.d.). Women's E-Commerce Clothing Reviews. Retrieved from <https://www.kaggle.com/datasets/nicapotato/womens-ecommerce-clothing-reviews>

DataCamp. (n.d.). Naive Bayes Classification Tutorial using Scikit-learn. Retrieved from <https://www.datacamp.com/tutorial/naive-bayes-scikit-learn>

MongoDB. (n.d.). MongoDB Node.js Driver 3.0 - Tutorials - Connect. Retrieved from <https://mongodb.github.io/node-mongodb-native/3.0/tutorials/connect/>

Tech With Tim. (n.d.). Python MongoDB Tutorial using PyMongo. YouTube. Retrieved from [https://www.youtube.com/watch?v=rE\\_bJl2GAY8&t=929s](https://www.youtube.com/watch?v=rE_bJl2GAY8&t=929s)