INTERNATIONAL UNIVERSITY

VIETNAM NATIONAL UNIVERSITY, HCM CITY

-----*-----**

School of Computer Science & Engineering

FINAL REPORT

Topic: Developing a Personalized Product Recommendation System with PySpark

Submitted by.

No.	Student Name	Student ID	Contribution(%)
1	Le Huynh Nha Nguyen	ITDSIU21058	50%
2	Phan Bao Tran	ITDSIU21125	50%

Course name: Big Data Technology

Professors: Dr. Ho Long Van

ACKNOWLEDGMENTS

We would like to express our deepest gratitude to Professor Ho Long Van, who directly guided us throughout the process of completing this project. Through the Big Data course taught by him, we have gained significant insights into the principles of Big Data and the related algorithms, as well as learned how to apply this knowledge to data analysis, exploration, and problem-solving to understand data and make data-driven decisions.

In addition to the specialized knowledge imparted during the lectures, Professor Ho Long Van also shared his invaluable practical experiences while researching this topic, as well as the benefits of applying this research to real-world business scenarios.

Given our limited knowledge and experience in researching this topic, we sincerely hope to receive feedback and constructive suggestions from Professor Ho Long Van to further improve our work.

HCMC, December 20, 2024

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 2: LITERATURE REVIEW

CHAPTER 3: DATA COLLECTION

CHAPTER 4: EXPERIMENTATION AND MODEL EVALUATION

CHAPTER 1: INTRODUCTION

Chapter 1 will provide an overview of the research topic, the reasons for selecting it, the objectives to be achieved, the scope of the study, as well as an understanding of the data and the system's functionality. Additionally, this chapter will present the resources used during the project execution.

1.1 Background

Recommendation systems have become an indispensable component of e-commerce platforms, online services, and mobile applications, particularly in the context of increasingly rich and diverse user data. These systems not only provide product recommendations but also leverage complex algorithms to personalize the user experience, thereby enhancing customer satisfaction and improving conversion rates. Major companies such as Amazon, Netflix, and Spotify have demonstrated the power of personalized recommendation systems in establishing deep connections with users, while simultaneously driving sales revenue and customer retention.

While traditional recommendation methods such as Collaborative Filtering and Content-based Filtering are still widely used, newer technologies like machine learning and big data are gradually replacing older approaches to deliver more accurate and effective results. The processing of vast amounts of data from user behaviors, product reviews, and transaction records has become a significant challenge for companies. As a result, the use of powerful data processing tools like Apache Spark and the PySpark library has emerged as an effective solution for handling and analyzing large-scale data. Therefore, the research and development of a personalized product recommendation system using PySpark not only meets market demand but also provides tangible value to businesses, contributing to the advancement of cutting-edge technologies in big data analytics and machine learning.

1.2 Rationale For The Topic

In the context of the rapidly growing e-commerce landscape, enhancing the customer experience has become a competitive advantage and a critical factor for businesses to stand out in a fiercely competitive market. The topic "Building a Personalized Product Recommendation System with PySpark" was chosen for profound and practical reasons, reflecting the synergy between real-world demands and the advancement of modern technology.

First and foremost, customer reviews serve as an exceptionally rich and valuable data source, providing an authentic reflection of consumer sentiments and needs. These reviews not only offer insights into product quality but also reveal customers' attitudes toward services and brands. Analyzing these reviews enables businesses to gain a deeper understanding of customer preferences and desires, as well as to identify emerging trends. This allows companies to adjust and optimize their product portfolios while enhancing service quality. The use of PySpark, a powerful and efficient tool for processing large-scale data, enables businesses to rapidly exploit and analyze massive volumes of data from millions of reviews, extracting valuable information regarding customer behavior without encountering issues related to processing speed or scalability.

Secondly, a personalized recommendation system not only contributes to enhancing customer satisfaction but also has a direct impact on the business's revenue. When customers receive personalized product suggestions that align with their preferences and needs, their likelihood of making a purchase increases significantly. According to numerous studies, personalizing the shopping experience can boost conversion rates and sales. These recommendations not only save time for customers but also introduce them to new products they may not have previously considered. Furthermore, PySpark allows the integration of machine learning algorithms, enabling the system to continuously improve and adapt based on customers' behaviors and shopping habits, thus increasing the accuracy of recommendations and optimizing business strategies in real-time.

Finally, developing a personalized recommendation system not only provides clear benefits to the business but also creates tangible value for the customer. In a world where product choices are abundant and diverse, offering products tailored to individual preferences and needs helps customers save time and effort in their search for suitable items. These recommendations can even lead to unexpected and delightful experiences, making customers feel cared for and valued. This not only provides convenience to customers but also strengthens their loyalty to the brand, fostering long-term and sustainable transactions. In conclusion, this topic holds not only academic value but also high practical relevance, contributing to the sustainable development strategy of businesses. Developing such a system will help companies optimize business operations, enhance marketing and sales efficiency, and create a smarter and more convenient shopping experience for customers, ultimately building a lasting and trustworthy relationship between the business and consumers.

1.3 Deliverables and Objectives

The project aims not only to develop an effective recommendation system but also to contribute to improving customer experience and enhancing business efficiency for enterprises in the e-commerce sector. Therefore, we have established the following objectives:

Analyze Review Data: Research and analyze customer reviews on e-commerce platforms to gain deeper insights into consumer perceptions, needs, and behaviors. This includes identifying the positive and negative sentiments within the reviews.

Optimize Data Processing: Utilize PySpark to efficiently process and analyze large volumes of data. This includes optimizing the processes of data collection, storage, and analysis based on customer reviews.

Build a Recommendation Model: Develop a personalized product recommendation model using machine learning algorithms integrated with PySpark. This model will leverage information from customer reviews to provide the most suitable product recommendations for each individual customer.

Enhance Customer Experience: Improve user experience on e-commerce platforms by providing tailored product suggestions based on customer feedback across any platform. This effort aims to save customers time in searching for products while increasing satisfaction and loyalty toward the brand.

Develop Professional Skills: Enhance the individual skills of team members in areas such as data analysis, programming with PySpark, and applying machine learning algorithms to practical problems through the implementation of this project.

1.4 Scope

The project utilizes a dataset titled *"Amazon Reviews on Women Dresses,"* containing over 23,000 entries of product reviews from the Amazon e-commerce platform. This dataset includes various details such as customer age, review content, product ratings, an indicator showing whether the customer recommends the product to others, and other relevant information. The data provides a comprehensive view of customer feedback, enabling a deeper analysis of consumer behavior and preferences, which is crucial for building an effective personalized product recommendation system.

1.5 Resource Utilization

Languages and tools used: in this project, we utilized PySpark with Python as the programming language to handle large-scale data processing in a distributed environment.

PySpark enabled us to efficiently perform tasks such as data cleaning, transformation, and analysis. Additionally, we opted for Google Colab instead of setting up a local environment on personal computers. This choice offered significant advantages, allowing us to work online and easily share notebooks for more effective collaboration. Moreover, working on Google Colab facilitated the handling of large datasets, as the team could directly access and store data on Google Drive.

Dataset: Women Dresses Review data

CHAPTER 2: LITERATURE REVIEW

Chapter 2 will focus on surveying related studies, and presenting the methods and models that have been applied. It will introduce and propose an analytical model suitable for the research objectives and business practices. At the same time, this chapter will also establish the theoretical foundation for the project.

2.1 Survey of the Research Status

Recommendation systems have been developed to assist users in making intelligent decisions by offering personalized suggestions for products and services. A significant advancement in this field is the integration of sentiment analysis, which helps extract user preferences from textual reviews, thereby enhancing the accuracy of recommendations. Traditional recommendation systems typically rely on Collaborative Filtering (CF), Content-Based Filtering (CBF), or hybrid approaches that combine multiple methods. CF uses past behaviors of similar users, while CBF focuses on recommending products that the user has interacted with previously. However, incorporating sentiment analysis provides a more dynamic and refined approach by considering explicit user feedback through their opinions and reviews.

Asani et al. in their study “Restaurant recommender system based on sentiment analysis” (2021) developed a context-based restaurant recommendation system that extracts users' food preferences from their reviews using sentiment analysis. This system employs a clustering method for food-related nouns based on semantic similarity, allowing for more accurate restaurant suggestions, with an accuracy rate of 92.8% based on data from TripAdvisor [1]. In the paper "An Approach to Integrating Sentiment Analysis into Recommender Systems", Cach N. Dang and colleagues used Singular Value Decomposition (SVD) to reduce data dimensionality in recommendation systems, particularly to condense and optimize the user-

product rating matrix. SVD helps improve system performance when combined with other sentiment analysis models such as BERT, CNN, and LSTM, thus enhancing the accuracy of recommendations [5].

In the field of text clustering - a key technique for organizing and filtering data in recommendation systems - Huang (2008) tested five similarity measurement methods, including Euclidean distance, cosine similarity, Jaccard coefficient, Pearson correlation coefficient, and average Kullback-Leibler (KL) divergence. The results indicated that cosine similarity, Jaccard coefficient, and Pearson coefficient were more effective than Euclidean distance in creating meaningful clusters, especially when the clusters were balanced in structure [2]. Research by Komal Maher and colleagues in "Effectiveness of Different Similarity Measures for Text Classification and Clustering" also reported 79% accuracy for cosine similarity when applied to k-NN, Naïve Bayes, and k-means algorithms, due to its ability to maintain stability across documents of different lengths [6].

In conclusion, modern recommendation systems have advanced further by integrating sentiment analysis, providing deeper personalized suggestions by leveraging user emotions and feedback. Notably, cosine similarity is an optimal choice, as it not only demonstrates high efficiency in text classification and clustering but also maintains stability across documents of varying lengths. This significantly improves the accuracy and quality of recommendations, offering a better user experience in finding suitable products and services.

2.2 Proposed Model

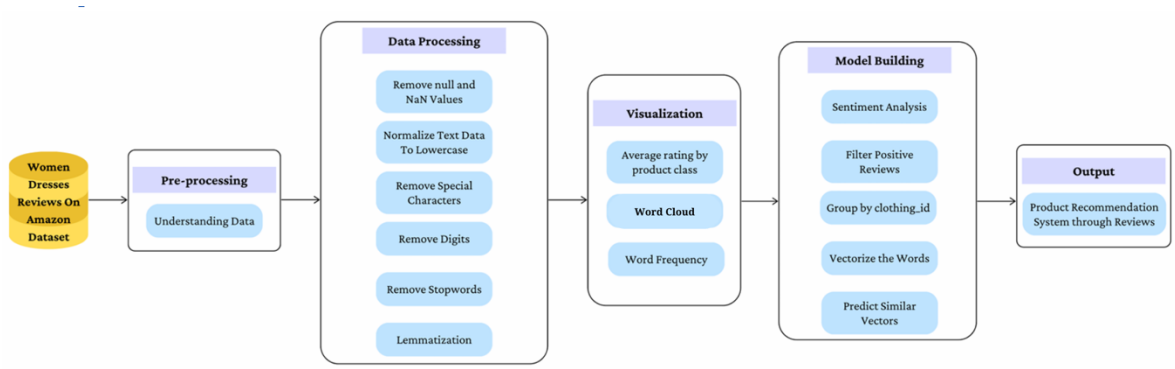

Figure 2. 1: Recommendation model

The process of analyzing fashion product reviews on the Amazon platform is carried out in five main stages. The pre-processing stage is the first and most crucial step in the analysis pipeline, as it ensures the quality of the input data. Initially, it is essential to understand the meaning of each column in the dataset.

Next, in the data processing stage, the first steps involve removing null and NaN values to ensure the data is complete and valid. Then, the text data is normalized to lowercase to eliminate unnecessary case sensitivity. Special characters, punctuation, and digits are removed to clean the data and reduce noise. Stopwords, such as "the," "is," and "and," are eliminated because they do not carry meaningful information. Finally, lemmatization is applied to standardize words to their root form, reducing variability and improving accuracy in semantic analysis.

After the data is processed, visualization steps help uncover trends and key patterns in the data. Visualizing average ratings by product class helps identify which product categories receive the highest ratings. A Word Cloud displays the most common keywords in customer reviews, revealing prominent themes or features that users care about. Lastly, word frequency analysis highlights recurring attributes mentioned in reviews, such as quality, design, or product features, which can inform and optimize product recommendation systems.

After completing the visualization stage, the model-building phase begins by using Vader to perform sentiment analysis on the reviews that have been filtered to remove noise. Only reviews with positive sentiment are retained. Reviews for the same product (based on the clothing_id) are then aggregated to avoid duplicates. Next, the text data is transformed into numerical vectors using the Word2Vec method, which facilitates the prediction of similar reviews or products. Finally, the similarity between vectors is calculated using cosine similarity, enabling accurate measurement of the relationship between reviews and products.
The final output is a recommendation system capable of suggesting products based on customer reviews with similar sentiments, allowing users to easily access products that align with their personal preferences on the Amazon platform.

2.3. Theoretical Basis

2.3.1 Sentiment Analysis

Sentiment analysis is the process of identifying and extracting the underlying emotions and opinions expressed in text. It is also known by various terms such as subjectivity analysis, opinion mining, and appraisal extraction. [7]

By utilizing sentiment analysis, organizations can gain valuable insights into customers' real-time sentiments and experiences, as well as the public perception of their brand. This approach enables companies to better align their products and services with customer needs.

Depending on the organization's goals, available resources, and the scope of the analysis, different types of sentiment analysis can be applied. Some common approaches include graded sentiment analysis, emotion detection, aspect-based sentiment analysis, and multilingual sentiment analysis.

Sentiment analysis employs machine learning algorithms to identify the emotional tone conveyed in the text, determining whether it is positive, negative, or neutral. The process of sentiment analysis typically follows these steps:

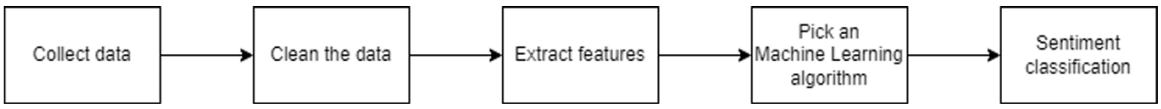

Figure 2. 2: Sentiment Analysis Process

2.3.2 NLTK and Vader

NLTK (Natural Language Toolkit) is an open-source software library developed to support tasks in natural language processing (NLP) using the Python programming language. First introduced in 2001, NLTK quickly became one of the most popular tools in the NLP field. This library provides a wide range of tools and resources for tasks such as tokenization (splitting text into smaller parts), stemming (finding the root form of words), tagging (assigning grammatical labels), parsing (syntax analysis), and semantic reasoning. Additionally, NLTK includes numerous language corpora and lexicons, making it easier for researchers and developers to integrate and apply them to their NLP projects.

VADER (Valence Aware Dictionary and Sentiment Reasoner) is a module within the NLTK library, specifically designed for sentiment analysis of English text. Unlike traditional sentiment analysis tools, VADER is particularly well-suited to handle expressions and informal language often found in short texts such as product reviews and social media posts. VADER not only provides scores for positive, negative, and neutral sentiments but also measures the intensity of the sentiment and determines its polarity. With a sentiment lexicon optimized for short text contexts, VADER has become a valuable tool for researchers and developers to analyze and better understand emotions in text.

2.3.3 Word2vec

Word2Vec is a machine learning technique developed by Google that is used to convert words into vectors in a high-dimensional space. This model captures the semantic relationships between words, allowing similar words to be represented closer together in the vector space.

How it works: Word2Vec uses two main methods for training the model:

- Continuous Bag of Words (CBOW): Predicts the target word based on the surrounding context words.
- Skip-Gram: Predicts the surrounding context words based on the target word.

Applications: Word2Vec is commonly used in tasks such as finding synonyms, text classification, and improving machine learning models by providing vector representations of words.

2.3.4 Cosine Similarity

Cosine similarity is a measure used to assess the similarity between two vectors. It calculates the angle between the two vectors in space and indicates the degree of similarity between them, with values ranging from -1 to 1.

Formula: Cosine similarity between two vectors A and B is given by:

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|}$$

Where:

- $A \cdot B$ is the dot product of A and B,
- $\|A\|$ and $\|B\|$ are the magnitudes (lengths) of vectors A and B.

The meaning of cosine similarity values:

- 1: The vectors are identical (100% similarity).
- 0: The vectors are orthogonal (no similarity).
- (-1): The vectors are diametrically opposed (completely dissimilar).

CHAPTER 3: DATA COLLECTION AND PREPROCESSING

Chapter 3 presents the workflow for working with data, including the steps from data collection to preprocessing and data visualization. The operations in this chapter play a crucial role in preparing the data for in-depth analysis in the subsequent chapters.

3.1 Data Pre-processing

3.1.1 Data Collection and Description

s_no	age	division_name	department_name	class_name	clothing_id	title	review_text	alike_feedback_count	rating	recommend_index
0	40	General	Bottoms	Jeans	1028	Amazing fit and wash	Like other reviewers i was hesitant to spend this much on	0	5	1
1	62	General Petite	Tops	Blouses	850	Lovely and unique!	As is true of a bunch of the fall clothing photos, the colors	12	5	1
2	47	General Petite	Bottoms	Skirts	993	Meh	I so wanted this skirt to work, love the design! but, it's way	3	1	0
3	45	General Petite	Bottoms	Pants	1068	Wow	Love love this! i was hesitant to buy this at first - the revie	0	5	1
4	37	Initmates	Intimate	Swim	24	Great for bigger busts	I absolutely love the retro look of this swimsuit. i first saw i	0	5	1
5	43	General	Tops	Sweaters	933	Love the pattern and c	I love this sweater but i'm on the fence about keeping it or	0	4	1
6	83	General	Tops	Sweaters	937	Beautiful and unique.	Love this sweater! soft and cozy and the ruffles are not overwhelming....just it will look pretty with a turquoise necklace for a pop of col classic, comfy and classy! an anhtro a ++++++++ runs very true to size, ordered my usual size an it fits perf	4	5	1
7	34	General	Tops	Knits	868	Unique and wonderful	The sweater is comfortable and a good weight. the zipper	2	5	1
8	49	General Petite	Tops	Fine gauge	900	Great look all in one	I love everything about this sweater. it is very well made, t	4	5	1
9	49	General	Tops	Knits	873	Beauty meets comfort	I love this top, the details at the neck and shoulders are lo	0	5	1
10	32	General	Tops	Knits	872	Great fit	I have been searching around for new and unique clothes	0	5	1
11	41	General Petite	Tops	Knits	873	Adore! but the white *i	I own this in the dark blue and adore it. i read reviews late	0	4	1
12	53	General	Bottoms	Skirts	1008	Not red, not orange...	Yes, as much as i'd have liked this to be a true red, it's kin	0	5	1
13	23	General	Bottoms	Skirts	1020	Cute, but not flattering	I really loved this skirt on the model, but it just wasn't flatte sadly, i had to return both. perhaps this would look better	0	2	0

Figure 3. 1: Women Dresses Reviews Dataset

"The 'Women Dresses Reviews' dataset originates from the Amazon e-commerce platform, where products are bought and sold electronically through online services. This dataset was obtained from Kaggle, a platform that hosts datasets for data science and machine learning projects. It contains 23,500 rows and 11 attributes, providing valuable insights into customer reviews and product details."

Link: Women Dresses Review data

Field Name	Data Type	Description	Field Value
S.no	Integer	Serial number	0 - 23500
Age	Integer	Reflects the age of the customer	18 - 99
Division name	String	The division of the product purchased , indicating the market segment.	- General: Regular products with no specific characteristics in size or type. - General Petite: Products designed specifically for petite

			customers with smaller body frames. - Intimates: Products related to lingerie, including items like bras, panties, and sleepwear.
Department name	String	Specifies product types	- Tops: Products for the upper body, such as shirts, t-shirts, or lightweight jackets. - Dresses: Various dresses, including formal gowns, office dresses, and casual dresses. - Bottoms: Products for the lower body, like pants, skirts, or shorts. - Intimate: Lingerie products, including bras, panties, and sleepwear. - Jackets: Outerwear, typically used for weather protection or fashion styling.
Class name	String	The product group or category.	- Dresses: This category includes various types of dresses suitable for different occasions, ranging from casual to special events. - Knits: This category encompasses knitted items, typically sweaters, accessories made from wool or other knit fabrics. - Blouses: This category includes blouses, which are

			often collared and sleeved tops, suitable for both office and casual settings. - Sweaters: This category comprises sweaters, common worn in cold weather. - Pants: This category includes various types of pants, such as long pants, shorts, and other styles.
Clothing id	Integer	ID of the product.	x
Title	String	A concise summary by the customer, expressing their overall view of the product.	x
Review text	String	Detailed feedback about product features like material, design, or fit.	x
Alike feedback count	Integer	The number of other customers who agreed with this feedback, showing shared experiences.	0 - 122
Rating	Integer	The star rating given by the customer, reflecting overall satisfaction.	1 - 5
Recommend Index	Boolean	Indicator of whether the customer recommends the product to others.	0 : No 1 : Yes

3.2 Data Preprocessing

This section outlines the data preprocessing steps undertaken to ensure the dataset is clean, consistent, and suitable for analysis. Key preprocessing tasks, including removal of null values, normalization, lemmatization, tokenization, removal of special characters, and elimination of stopwords, are applied primarily to the review text and title columns. These two columns are the primary focus of the analysis, as they contain the most relevant information

3.2.1 Removing null or NaN Values

Raw data often contains null or missing values, which can compromise data integrity and reduce the accuracy and reliability of subsequent analyses. Ensuring high-quality data requires identifying and handling null values as a critical and essential step in the data preprocessing workflow. Initially, rows containing null values are identified through comprehensive data checks. These rows are then removed or appropriately handled to ensure the dataset remains consistent and free from critical information gaps. This process not only enhances the completeness of the dataset but also lays a solid foundation for more accurate analysis and model development. A clean and comprehensive dataset ensures both integrity and improves the efficiency of algorithms in delivering reliable results.

```
from pyspark.sql.functions import col, isnan
print("Number of NULL 'age':", df.filter(col("age").isNull() | isnan(col("age"))).count())
print("Number of NULL 'division_name':", df.filter(col("division_name").isNull() | isnan(col("division_name"))).count())
print("Number of NULL 'department_name':", df.filter(col("department_name").isNull() | isnan(col("department_name"))).count())
print("Number of NULL 'class_name':", df.filter(col("class_name").isNull() | isnan(col("class_name"))).count())
print("Number of NULL 'review_text':", df.filter(col("review_text").isNull() | isnan(col("review_text"))).count())
print("Number of NULL 'title':", df.filter(col("title").isNull() | isnan(col("title"))).count())
```

```
Number of NULL 'age': 0
Number of NULL 'division_name': 14
Number of NULL 'department_name': 14
Number of NULL 'class_name': 14
Number of NULL 'review_text': 845
Number of NULL 'title': 3810
```

```
nan_values = df.filter(col("division_name").isNull() | isnan(col("division_name")))
nan_values.show()
```

```
+-----+---+-------------+---------------+----------+-----------+--------------------+--------------------+--------------------+------+---------------+
| s.no|age|division_name|department_name|class_name|clothing_id|               title|         review_text|alike_feedback_count|rating|recommend_index |
+-----+---+-------------+---------------+----------+-----------+--------------------+--------------------+--------------------+------+---------------+
|  593| 34|         NULL|           NULL|      NULL|        184|Nubby footless ti...|These are amazing...|                   5|     5|              1|
| 2275| 36|         NULL|           NULL|      NULL|        152|       Warm and cozy|Just what i was l...|                   0|     5|              1|
| 4349| 43|         NULL|           NULL|      NULL|        665|        So worth it!|Got these on sale...|                   0|     5|              1|
| 4745| 50|         NULL|           NULL|      NULL|        772|   Comfy sweatshirt!|This sweatshirt i...|                   0|     5|              1|
| 7589| 39|         NULL|           NULL|      NULL|        152|     "long and warm"|These leg warmers...|                   0|     5|              1|
| 8163| 25|         NULL|           NULL|      NULL|         72|My favorite socks!!!|I never write rev...|                   0|     5|              1|
| 8333| 36|         NULL|           NULL|      NULL|        136|         Super socks|I love these litt...|                   0|     5|              1|
|14464| 54|         NULL|           NULL|      NULL|        184|       New workhorse|These tights are ...|                   0|     5|              1|
|14670| 49|         NULL|           NULL|      NULL|        492|     Wardrobe staple|Love this hoodie....|                   0|     5|              1|
|15513| 37|         NULL|           NULL|      NULL|        152|               Love!|I am loving these...|                   0|     5|              1|
|20789| 47|         NULL|           NULL|      NULL|        136|Charcoal, pale gr...|These socks are s...|                   1|     5|              1|
|22325| 48|         NULL|           NULL|      NULL|        492|                NULL|                NULL|                   0|     5|              1|
|22679| 33|         NULL|           NULL|      NULL|        136|     Cute itsy socks|Love polkadots, l...|                   0|     5|              1|
|23181| 23|         NULL|           NULL|      NULL|        492|           So soft!|I just love this ...|                   1|     5|              1|
+-----+---+-------------+---------------+----------+-----------+--------------------+--------------------+--------------------+------+---------------+
```

Figure 3. 2: Check for Null Values

3.2.2 Normalize Text to Lowercase

Text data often contains uppercase characters, which can lead to inconsistencies and complicate the analysis process, especially in automated language processing systems. To address this issue, converting all text to lowercase is a crucial standardization step. This step not only minimizes unnecessary differences caused by variations in capitalization but also ensures uniformity in data structure, thereby enhancing the efficiency of processing algorithms. Moreover, standardizing text to lowercase plays a key role in optimizing semantic analysis, classification, and information extraction models, ensuring accuracy and effectiveness when handling large or complex datasets. This approach is particularly beneficial when analyzing diverse text data that may vary in format or style.

```python
from pyspark.sql import Row

# Convert DataFrame to RDD
rdd = df_cleaned.rdd

def process_row(row):
    try:
        return Row(
            age=row.age,
            division_name=row.division_name,
            department_name=row.department_name,
            class_name=row.class_name,
            clothing_id=row.clothing_id,
            title=row.title.lower() if row.title else None,
            review_text=row.review_text.lower() if row.review_text else None,
            alike_feedback_count=row.alike_feedback_count,
            rating=row.rating,
            recommend_index=row['recommend_index']
        )
    except AttributeError as e:
        print(f"Lỗi xử lý dòng: {row}, lỗi: {e}")
        return None

rdd = rdd.map(process_row).filter(lambda x: x is not None)
result = rdd.take(5)

print("Data after conversion to lowercase in title and review_text:")
for row in result:
    print(row)
```

Figure 3. 3: Normalize Text to Lowercase

3.2.3 Removing Non-Alphabetic Characters

Text data often contains special characters, numbers, or other unnecessary elements that can introduce noise and reduce the effectiveness of language analysis models. These characters typically lack significant semantic value and may complicate the analysis process, particularly in natural language processing tasks. Removing all unnecessary characters is a crucial standardization step that helps retain meaningful words while eliminating distracting elements. This process not only enhances the consistency and cleanliness of the data but also optimizes the performance of classification and semantic analysis algorithms, ensuring higher accuracy

and efficiency during data processing. Furthermore, this standardization facilitates the application of machine learning models to analysis, ensuring that the algorithms focus solely on information of true value.

```
import re
def replace_special_characters(text):
    return re.sub(r'[^\w\s]', '', text)

rdd = rdd.map(lambda row: Row(
    age=row.age,
    division_name=row.division_name,
    department_name=row.department_name,
    class_name=row.class_name,
    clothing_id=row.clothing_id,
    title=replace_special_characters(row.title),
    review_text=replace_special_characters(row.review_text),
    alike_feedback_count=row.alike_feedback_count,
    rating=row.rating,
    recommend_index=row['recommend_index']
))

result = rdd.take(5)

print("Data after replacing special characters in title and review_text:")
for row in result:
    print(row)
```

Figure 3. 4: Removing Non-Alphabetic Characters

3.2.4 Removing Stop Words

In semantic analysis, stopwords such as "and," "of," "is," often lack significant informational value and do not contribute meaningfully to understanding the core content of the text. The presence of these words can reduce the efficiency and accuracy of analysis models, especially in tasks like classification or information extraction. Therefore, removing stopwords is a critical step in data preprocessing. This process cleans the text data, reduces the input data size, and focuses on words with higher analytical value, thereby enhancing the performance and precision of natural language processing algorithms.

```
[ ] import nltk
    nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

Figure 3. 5: Download NLTK library

```python
from pyspark.sql import SparkSession
from nltk.corpus import stopwords

spark = SparkSession.builder.appName("Remove Stopwords").getOrCreate()

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

def remove_stopwords(text):
    words = text.split()
    filtered_words = [word for word in words if word.lower() not in sw]
    return ' '.join(filtered_words)

rdd = rdd.map(lambda row: Row(
    age=row.age,
    division_name=row.division_name,
    department_name=row.department_name,
    class_name=row.class_name,
    clothing_id=row.clothing_id,
    title=remove_stopwords(row.title),
    review_text=remove_stopwords(row.review_text),
    alike_feedback_count=row.alike_feedback_count,
    rating=row.rating,
    recommend_index=row.recommend_index))

result = rdd.take(5)
print("Data after removing stopword in title and review_text:")
for row in result:
    print(row)
```

Figure 3. 6: Removing Stopwords

3.2.5 Lemmatization

Raw text data often contains multiple variations of the same word (e.g., "go," "going," "went"), creating inconsistencies and potential noise in the analysis. Normalizing words to their base form, known as lemmatization, reduces this variability by converting words to their root form. This process not only reduces vocabulary size but also significantly enhances the accuracy and efficiency of natural language analysis models.

Implementing lemmatization requires the use of the NLTK library, a powerful tool for natural language processing. Initially, the NLTK library must be installed. Then, the necessary resources for the lemmatization process, including *wordnet* and *omw-1.4*, are downloaded to ensure accurate conversion of words to their base forms. This step is critical for standardizing and optimizing the data before proceeding with advanced analysis.

```python
!pip install nltk
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.6)
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

```python
from nltk.stem import WordNetLemmatizer
spark = SparkSession.builder.appName("Lemmatization of Review Text").getOrCreate()

lemmatizer = WordNetLemmatizer()

def lemmatize_review(review):
    return " ".join([lemmatizer.lemmatize(word) for word in review.split()])

rdd = rdd.map(lambda row: Row(
    age=row.age,
    division_name=row.division_name,
    department_name=row.department_name,
    class_name=row.class_name,
    clothing_id=row.clothing_id,
    title=lemmatize_review(row.title),
    review_text=lemmatize_review(row.review_text),
    alike_feedback_count=row.alike_feedback_count,
    rating=row.rating,
    recommend_index=row.recommend_index))

result = rdd.take(5)
print("Data after lemmatizating in title and review_text:")
for row in result:
    print(row)
```

Figure 3. 7: Lemmatization

3.3 Data Visualization

3.3.1 Average Rating in Product Class

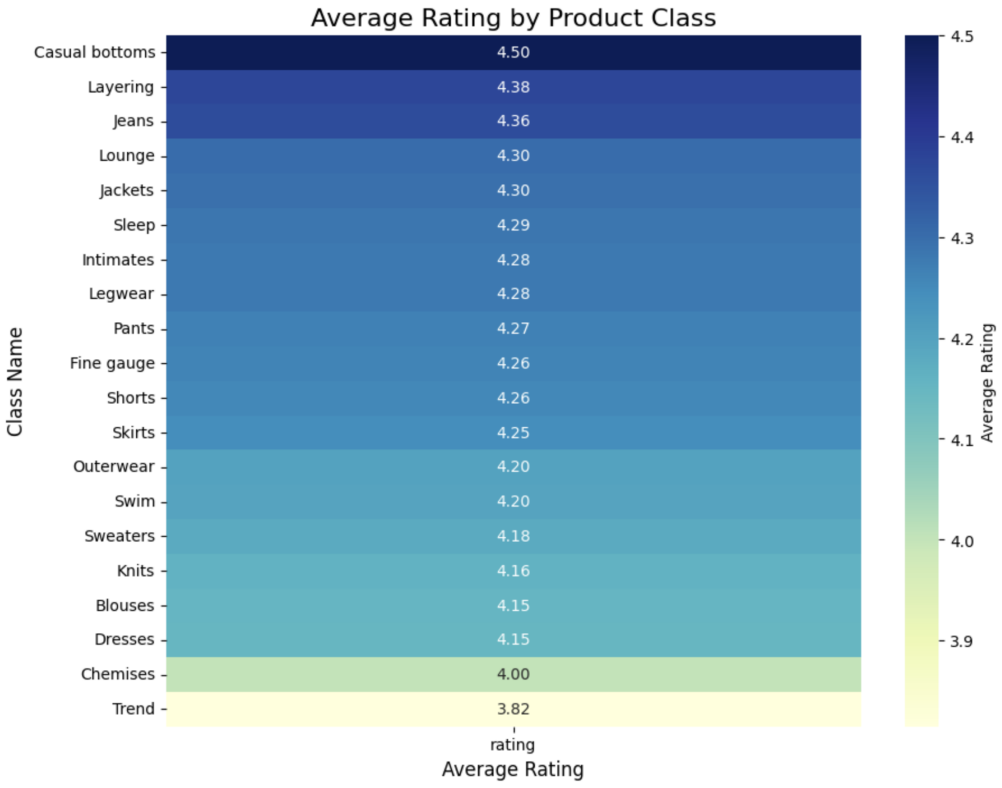

Figure 3. 8: Average rating in Product Class

The Average Rating by Product Class shows that Casual Bottoms lead with the highest average rating of 4.50, reflecting strong customer satisfaction. Categories like Layering and Jeans follow closely with ratings of 4.38 and 4.36, indicating their consistent quality and popularity. On the other hand, Chemises and Trend have the lowest ratings at 4.00 and 3.82, suggesting areas for improvement in design or quality. Most categories, including Knits, Blouses, and Dresses, fall between 4.15 and 4.30, reflecting generally positive feedback but room for enhancement. Notably, Swim and Sweaters show moderate ratings around 4.20, indicating average performance. Overall, while most product classes achieve ratings above 4.0, targeted improvements in lower-rated categories can further boost customer satisfaction.

3.3.2 WordCloud

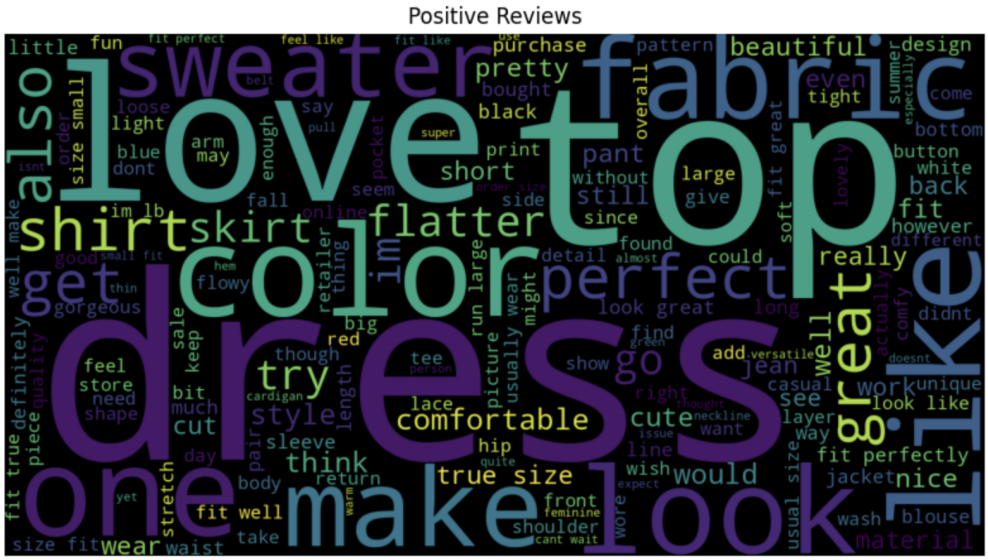

Figure 3. 9: WordCloud

The Word Cloud illustrates the most frequently mentioned words in customer reviews for fashion products, with terms like "size," "fit," "dress," and "top" being the most prominent. This indicates that customers primarily focus on size, fit, and dress-related products. Additionally, words such as "love" and "like" appear frequently, reflecting high levels of satisfaction and positive feedback from customers. Words related to material and color, such as "fabric," "color," and "comfortable," also stand out, emphasizing the importance of these aspects in customers' evaluations of fashion products.

3.3.3 Word Frequency

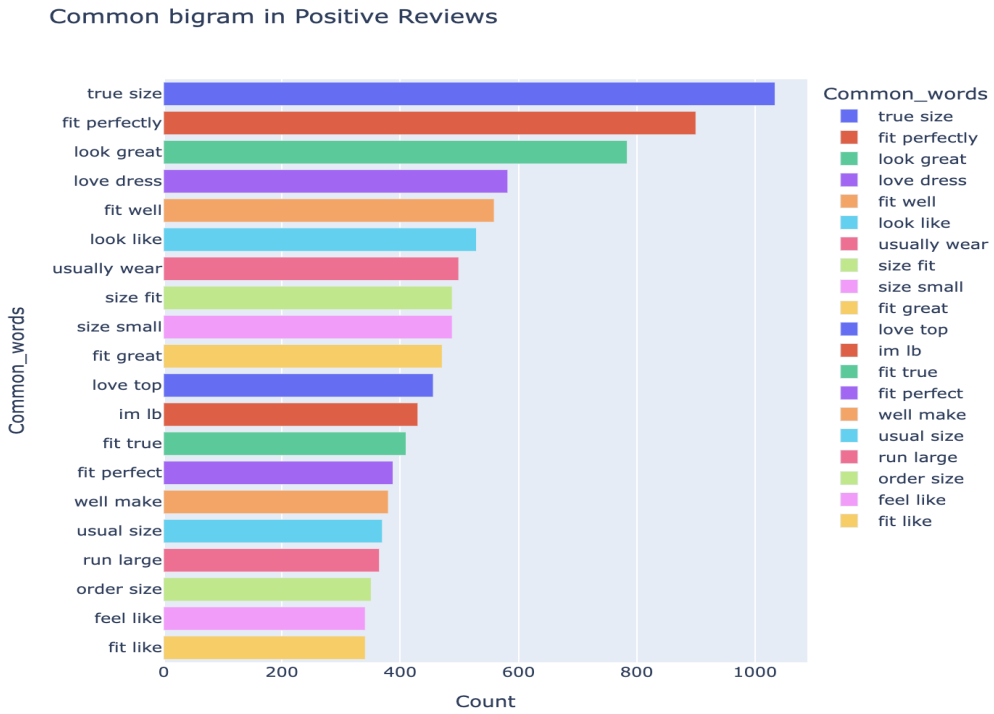

Figure 3. 10: Word Frequency

The chart "Common bigram in Positive Reviews" highlights key phrases frequently mentioned in favorable customer feedback. The bigram "true size" dominates with the highest count, indicating that accurate sizing significantly impacts positive reviews. Following closely, phrases such as "fit perfectly," "look great," and "fit well" suggest that customers value well-fitting and visually appealing products. Words like "size fit" and "size small" further emphasize the importance of size accuracy in driving satisfaction. Additionally, expressions like "love dress" and "love top" reflect strong customer appreciation for specific clothing items. Phrases like "run large" and "order size" indicate that size consistency remains a focal point. Overall, the insights underline that fit, size accuracy, and visual appeal are critical drivers of customer satisfaction, guiding product quality improvement efforts.

CHAPTER 4: EXPERIMENTATION AND MODEL EVALUATION

Chapter 4 focuses on the experimentation and evaluation of the proposed model. This chapter will carry out sentiment analysis, experiment with the algorithm, and ultimately evaluate the experimental results.

4.1. Sentiment Analysis

Next, filter the rows where the recommend_index is not equal to 0 and the rating is not 1 or 2, meaning remove the rows where recommend_index is 0 or rating is 1 or 2. Finally, df.show() will display the result of the DataFrame after it has been filtered.

```
df = df_loaded.filter((df_loaded["recommend_index"] != 0) & (df_loaded["rating"] != 1) & (df_loaded["rating"] != 2))
df.show()
```

```
+---+--------------+---------------+----------+-----------+--------------------+--------------------+-------------------+------+---------------+
|age| division_name|department_name|class_name|clothing_id|               title|         review_text|alike_feedback_count|rating|recommend_index|
+---+--------------+---------------+----------+-----------+--------------------+--------------------+-------------------+------+---------------+
| 40|       General|        Bottoms|     Jeans|       1028|amazing fit and wash|like reviewer hes...|                  0|     5|              1|
| 62|General Petite|           Tops|   Blouses|        850|   lovely and unique|true bunch fall c...|                 12|     5|              1|
| 45|General Petite|        Bottoms|     Pants|       1068|                 wow|love love hesitan...|                  0|     5|              1|
| 37|     Initmates|       Intimate|      Swim|         24|great for bigger ...|absolutely love r...|                  0|     5|              1|
| 43|       General|           Tops|  Sweaters|        933|love the pattern ...|love sweater im f...|                  0|     4|              1|
| 83|       General|           Tops|  Sweaters|        937|beautiful and unique|love sweater soft...|                  4|     5|              1|
| 34|       General|           Tops|     Knits|        868|unique and wonderful|sweater comfortab...|                  2|     5|              1|
| 49|General Petite|           Tops|Fine gauge|        900|great look all in...|love everything s...|                  4|     5|              1|
| 49|       General|           Tops|     Knits|        873|beauty meets comfort|love top detail n...|                  0|     5|              1|
| 32|       General|           Tops|     Knits|        872|           great fit|searching around ...|                  0|     5|              1|
| 41|General Petite|           Tops|     Knits|        873|adore but the whi...|dark blue adore r...|                  0|     4|              1|
| 53|       General|        Bottoms|    Skirts|       1008|  not red not orange|yes much id liked...|                  0|     5|              1|
| 50|General Petite|        Dresses|   Dresses|       1078| great with leggings|dress short wear ...|                 13|     4|              1|
| 50|General Petite|        Bottoms|     Pants|       1060| perfect white pants|love pant purchas...|                  0|     5|              1|
| 40|     Initmates|       Intimate|      Swim|        286|              wowser|made dawn suit bo...|                  1|     5|              1|
| 38|       General|        Bottoms|     Pants|       1053|pretty but a bit ...|cant decide pant ...|                 12|     4|              1|
| 34|       General|        Bottoms|    Shorts|        177| great summer staple|fit quite well ra...|                  2|     5|              1|
| 36|       General|        Jackets|   Jackets|        967|       great jacket|one cutest jacket...|                  5|     5|              1|
| 72|       General|        Dresses|   Dresses|       1094|perfect for irish...|bought dress gree...|                  0|     5|              1|
| 42|       General|           Tops|   Blouses|        829|even more vibrant...|color blouse beau...|                  0|     5|              1|
+---+--------------+---------------+----------+-----------+--------------------+--------------------+-------------------+------+---------------+
only showing top 20 rows
```

Figure 4. 1: Remove unused rating values

Sentiment analysis is a crucial step in evaluating customer feedback. After collecting and preprocessing data from product reviews, the next step is to classify the sentiment of these reviews to better understand customer satisfaction. The team used Sentiment Analysis to analyze emotions, particularly through the VADER (Valence Aware Dictionary and Sentiment Reasoner) model. VADER is a model especially well-suited for analyzing short text segments like product reviews because it takes into account both the vocabulary and context to calculate sentiment scores.

Implementation Process

After selecting the relevant columns from the DataFrame, including the review content (review_text), review title (title), and metrics such as the rating, the team proceeded with sentiment analysis through the following steps:

1. **Initialize the VADER Model**: We used the VADER lexicon from the NLTK library to calculate the sentiment polarity scores for the reviews. The sentiment score is computed on a scale from -1 (very negative) to 1 (very positive).

```
!pip install nltk
import nltk

nltk.download('vader_lexicon')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.6)
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
True
```

```
from pyspark.sql.functions import udf, col
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from pyspark.sql.types import StringType, FloatType
```

```
spark = SparkSession.builder.appName("SelectColumns").getOrCreate()
new_data = df.select("department_name","class_name","clothing_id","title", "review_text", "rating","recommend_index")
new_data.show()
```

Figure 4. 2: Initialize VADER model

2. **Create UDF (User Defined Functions)**: Two UDFs were defined:

- The first function, get_polarity_score(), calculates the overall sentiment score for each review.
- The second function, get_sentiment_label(), assigns a sentiment label to the reviews based on the overall sentiment score:
    + Positive if the sentiment score > 0.05.
    + Negative if the sentiment score < -0.05.
    + Neutral if the sentiment score is between -0.05 and 0.05.

```
def get_polarity_score(review):
    return float(sia.polarity_scores(review)['compound'])

def get_sentiment_label(review):
    score = sia.polarity_scores(review)['compound']
    if score > 0.05:
        return "positive"
    elif score < -0.05:
        return "negative"
    else:
        return "neutral"

polarity_score_udf = udf(get_polarity_score, FloatType())
sentiment_label_udf = udf(get_sentiment_label, StringType())
```

Figure 4. 3: Create UDF (User Defined Functions)

3. **Apply Sentiment Analysis**: After registering these UDFs in PySpark, we applied them to the review_text column to create two new columns:

- polarity_score: The sentiment score for each review.
- sentiment_label: The corresponding sentiment label (Positive, Negative, Neutral)

```
polarity_score_udf = udf(get_polarity_score, FloatType())
sentiment_label_udf = udf(get_sentiment_label, StringType())

new_data = new_data.withColumn("polarity_score", polarity_score_udf(col("review_text")))
new_data = new_data.withColumn("Sentiment_Label", sentiment_label_udf(col("review_text")))
```

Figure 4. 4: Apply Sentiment Analysis

4.2 Algorithm Experimentation

In the input data processing pipeline, the team carried out steps to aggregate and optimize information from product reviews. The main goal was to minimize duplication of product IDs (clothing_id) and enhance the ability to analyze the data.

By grouping reviews based on the clothing ID, the team eliminated unnecessary duplicate records and retained the most important information. Attributes such as department name, product class name, and product title were assigned the first available value, while the review content (review_text) was concatenated into a single text string. This approach helps create an overview of customer sentiments toward the product.

In addition, the team also calculated average metrics for review factors such as rating, recommendation index, and sentiment score. Based on these metrics, the team automatically identified recommended products based on the average recommendation score. This process not only optimizes the data but also enhances the ability to analyze and make decisions in the subsequent steps.

clothing_id	department_name	class_name	title	combined_review_text
100	Intimate	Intimates	sizing is off	good design comfortable fit however sizin
1000	Bottoms	Skirts	awesome skirt	fit like model even though im really happ
1004	Bottoms	Skirts	great skirt and yetno pockets	bought faded pink rose color inexplicably
1006	Bottoms	Skirts	beautiful but not exactly as pictured	beautiful well made skirt flow curve beau
1008	Bottoms	Skirts	love this skirt	skirt fit amazing run bottom went little
101	Intimate	Intimates	so cute	expensive panty worth money unique flatte
1010	Bottoms	Skirts	petites please	absolutely adore skirt sadly lot skirt im
1012	Bottoms	Skirts	adorable	pretty pocket also nice quality price int
1013	Bottoms	Skirts	great work skirt	skirt nice medium weight lined normal siz
1016	Bottoms	Skirts	beautiful skirt	love skirt lovely soft gauze layer lining
1017	Bottoms	Skirts	nice basic	nice basic pencil skirt closet think fit
1021	Bottoms	Skirts	love this skirtget it	beautiful skirt great piece collect erin
1022	Bottoms	Jeans	a long last perfect leggings	cant say enough pant often find legging r
1027	Bottoms	Jeans	love these	th pair currentelliot jean ive bought im
103	Intimate	Layering	shimmer in silvergrey size down	ribbed camis stretch size prettydelicate
1030	Bottoms	Jeans	bought 2 pairs	cord went sale bought nd pair great work
1038	Bottoms	Jeans	floral fun	absolutely love jean rise mid perfect low
1039	Bottoms	Jeans	great summer jeans	jean fit true size look amazing wear athl
1040	Bottoms	Jeans	great fall jeans	fit incredible print subtle make little b
1041	Bottoms	Pants	sophisticated feminine overalls	yeah know sound ridiculous really nice pi

Figure 4. 5: Data for model building

First, to build the model, the team used the Word2Vec technique to transform both old and new reviews into vectors.

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import Word2Vec
from pyspark.sql import functions as F

spark = SparkSession.builder.appName("ReviewSimilarity").getOrCreate()

new_review_text = "This is so comfortable and beautiful"
new_review_df = spark.createDataFrame([(new_review_text,)], ["combined_review_text"])
tokenized_df = aggregated_comments.select("class_name","combined_review_text",F.split(F.col("combined_review_text"), " ").alias("words"))

tokenized_new_review = new_review_df.select(F.split(F.col("combined_review_text"), " ").alias("words"))

word2Vec = Word2Vec(vectorSize=100, minCount=0, inputCol="words", outputCol="result")
model = word2Vec.fit(tokenized_df)

result_df = model.transform(tokenized_df)

new_result_df = model.transform(tokenized_new_review)
new_vector = new_result_df.collect()[0].result
```

Figure 4. 6: Word2Vec model

After completing the transformation process, the next step in the workflow is to calculate the similarity between the new review and the old reviews to determine similarity. To accomplish this, the team defined a cosine similarity function, which helps evaluate the degree of similarity between two vectors. This function calculates a similarity value ranging from -1 to 1, where 1 represents perfect similarity and 0 indicates no similarity.

```
def cosine_similarity(v1, v2):
    norm_v1 = float(v1.norm(2))
    norm_v2 = float(v2.norm(2))
    if norm_v1 == 0 or norm_v2 == 0:
        return 0.0
    return float(v1.dot(v2)) / (norm_v1 * norm_v2)

similarities = []
for row in result_df.collect():
    similarity = cosine_similarity(new_vector, row.result)
    similarities.append((row.words, similarity, row.class_name, row.combined_review_text))
```

Figure 4. 7: Function to calculate similarity

After completing the calculations, the team converted the list of similarity results into a DataFrame, which included columns containing the words from the review, the similarity values, product class names, and the review content. From this, the team filtered out the top 5 reviews with the highest similarity to the new review (ensuring no product duplication), helping to identify the products most likely to be a good fit.

```
similarities_df = spark.createDataFrame(similarities, ["words", "similarity", "class_name", "combined_review_text"])
similarities_df = similarities_df.select("class_name", "similarity") \
    .orderBy("similarity", ascending=False)
```

```
similarities_df.show()
```

```
+----------+------------------+
|class_name|        similarity|
+----------+------------------+
|    Lounge|0.8240463035813141|
|      Swim| 0.818630007178348|
|    Lounge|0.8118630790012138|
|    Lounge|0.8013311285498402|
| Intimates|0.7771936148549765|
|    Shorts|0.7768788095691238|
|    Shorts|0.7732644691660442|
|  Layering|0.7681678930943401|
|      Swim|0.7664632296083066|
|    Lounge|0.7630835723484342|
|    Lounge|0.7588442006808428|
```

Figure 4. 8: Filtering products based on similarity and sorting them in descending order

```
top_5_similarities = similarities_df.limit(5)

for row in top_5_similarities.collect():
    print(f"Recommended Products: '{row.class_name}'")
```

Figure 4. 9: Filter top 5 recommended products based on similarity

4.3 Experimental Results and Evaluation

4.3.1 Results

During the experimentation process, the model successfully recommended five products based on the similarity between the new review and the previously collected reviews. Specifically, the recommendation results are as follows:

```
Recommended Products: 'Lounge'
Recommended Products: 'Swim'
Recommended Products: 'Lounge'
Recommended Products: 'Lounge'
Recommended Products: 'Intimates'
```

4.3.2 Model Evaluation

The model uses the Word2Vec technique to convert reviews into vectors, allowing it to calculate the similarity between new and old reviews using the cosine similarity method. This enables the model to identify products that align with the content of user reviews, enhancing the personalized shopping experience.

- **Accuracy:** The model successfully recommends relevant products, including 'Lounge,' 'Swim,' 'Lounge,', 'Lounge,' and 'Intimates.' These products provide diverse options and are consistent with the content of the reviews. The repetition of. 'Lounge' also indicates the model's ability to accurately select products that match the user's needs and preferences, ensuring a comprehensive and precise recommendation
- **Consistency:** The model demonstrates consistency in providing relevant product recommendations, ensuring users receive a diverse and valuable list of suggestions. The presence of products from various categories, such as swimwear ('Swim'), sleepwear ('Lounge'), and casual wear ('Lounge'), shows the model's ability to understand the semantic meaning of reviews and make appropriate suggestions based on them.
- **Potential for Improvement:** Although the results are promising, there is still potential for further optimization to enhance the user experience. One improvement could be to expand the number of recommended products, providing users with more options. Additionally, the model could be upgraded to handle more complex or specific review cases, ensuring greater accuracy and flexibility when applied in real-world environments.

This model shows great potential in improving the customer shopping experience by analyzing the semantics of reviews to suggest suitable products. The results not only provide accurate and diverse suggestions but also demonstrate high consistency. Continued improvement and optimization of the model could lead to even richer recommendations, thus enhancing its value and better meeting customer needs.

CHAPTER 5: CONCLUSION

Chapter 5 summarizes the results achieved after the research and experimentation process. It outlines the outcomes obtained and the practical value of the proposed model. The limitations section updates on the remaining challenges and presents directions for future development.

5.1. Results

The project "Building a Personalized Product Recommendation System with PySpark" focuses on using a product prediction model based on customer feedback, particularly through user comments. Some key findings of the research include:

- Comprehensive analysis of customer feedback data: The study performed an in-depth analysis of customer comments, extracting key features such as sentiment, preferences, and shopping behavior. These features highlight the relationships between products that customers are interested in, making it easier for the system to recommend similar products based on the user's latent needs.
- Application of Natural Language Processing (NLP) techniques and prediction models: The research team applied NLP techniques, specifically using Vader and Word2Vec, to convert the content of comments into vectors, enabling accurate similarity measurement between reviews. This allows the system to recommend products that fit the context of each individual review, optimally catering to the distinct needs and preferences of each customer.
- Effectiveness of the recommendation system: The research results show that the model achieves high accuracy in suggesting similar products, with recommendations closely matching customer preferences. This not only enhances the shopping experience but also supports e-commerce businesses in increasing revenue by boosting conversion rates.

Overall, this study demonstrates the feasibility and effectiveness of a personalized product recommendation model based on customer feedback analysis. The insights from this research provide businesses with strategic directions for personalizing shopping experiences, improving marketing strategies, and enhancing customer retention. This is a highly valuable study with practical implications, helping businesses optimize services and better meet customer needs in the e-commerce sector.

5.2. Limitations

This study has some limitations that should be addressed in future research to improve the model's accuracy and generalizability.

Firstly, the model has not been tested on large datasets, which could affect its reliability and ability to generalize to real-world situations. The lack of diverse data makes it harder for the model to reflect actual consumer behavior.

Secondly, the study doesn't provide clear methods or metrics to measure the accuracy of the NLP models used. Without these, it's difficult to evaluate the model's performance or compare it with other models.

Finally, the study primarily focuses on analyzing customer reviews and does not consider other attributes that could influence customer shopping behavior, such as product ratings, satisfaction levels, or recommendations. The failure to incorporate these factors may reduce the model's predictive accuracy and feasibility in suggesting personalized products.

5.3. Work Future

In the future, this study could focus on several key areas to enhance the effectiveness and accuracy of the product recommendation system. First, expanding the scale and diversifying the data will be a top priority, enabling the model to generalize better and accurately reflect real-world consumer trends. The study could also apply modern evaluation methods such as cross-validation and A/B testing to assess the accuracy of the natural language processing (NLP) models, using metrics like F1-score, Precision, Recall, and Area Under Curve (AUC) to measure performance in classification and prediction.

Additionally, incorporating other attributes such as user information, shopping context, and product characteristics would improve the quality of recommendations. Integrating these factors would allow the model to provide more personalized suggestions, better meeting the individual needs and preferences of each customer.

The study could also explore the application of deep learning techniques and advanced machine learning models to optimize the learning process and enhance semantic analysis, leading to a better shopping experience for consumers.

At the same time, the study could look into the development of Big Data applications by storing and loading models for online execution, through building web or mobile applications. This implementation would not only extend the reach to consumers but also integrate product analysis and recommendation functions seamlessly, providing a better shopping experience in the e-commerce sector.

REFERENCES

[1] Ziani, A., Azizi, N., Schwab, D., Aldwairi, M., Chekkai, N., Zenakhra, D., & Cheriguene, S. (2017). *Recommender system through sentiment analysis*. 2nd International Conference on Automatic Control, Telecommunications and Signals, Annaba, Algeria.

[2] Vijaymeena, M. K., & Kavitha, K. (2016). *A survey on similarity measures in text mining*. Machine Learning and Applications: An International Journal, 3(2), 19-28.

[3] Sahu, S., Kumar, R., MohdShafi, P., Shafi, J., Kim, S., & Ijaz, M. F. (2022). A hybrid recommendation system of upcoming movies using sentiment analysis of YouTube trailer reviews. *Mathematics*, 10(9), 1568.

[4] Breitfuss, A., Errou, K., Kurteva, A., & Fensel, A. (2021). Representing emotions with knowledge graphs for movie recommendations. *Future Generation Computer Systems*, 125, 715 725.

[5] Dang, C. N., Moreno-García, M. N., & Prieta, F. D. L. (2021). An approach to integrating sentiment analysis into recommender systems. *Sensors*, 21(16), 5666.

[6] Maher, K., & Joshi, M. S. (2016). Effectiveness of different similarity measures for text classification and clustering. *International Journal of Computer Science and Information Technologies*, 7(4), 1715-1720.

[7] Pang, B., & Lee, L. (2008). *Opinion mining and sentiment analysis*. Foundations and Trends in Information Retrieval*, 2*(1-2), 1-135.