

Master of Science (MS) in Data Science  
Module: ITC6010A1 – Natural Language Processing

Instructor: Lazaros Polymenakos

Students’ names:

Alkiviadis Kariotis 241735

Celsa Almeida 267628

Georgios Aliferis 272183

Konstantinos Margaritis 122451

Term: Spring Term 2023  
Type: Group Term Project

Submission Date: Friday, July

Words:

Table of Contents

[Introduction 3](#_Toc140006775)

[Main Body 4](#_Toc140006776)

[Data Loading & Preprocessing 4](#_Toc140006777)

[Data Loading 4](#_Toc140006778)

[Data Preprocessing 4](#_Toc140006779)

[Exploratory Data Analysis 5](#_Toc140006780)

[Sentiment Analysis 8](#_Toc140006781)

[Topic Extraction 9](#_Toc140006782)

[Topic Extraction with LDA 9](#_Toc140006783)

[Topic Extraction with NMF 10](#_Toc140006784)

[Text Summarization 11](#_Toc140006785)

[Sentiment Analysis | Logistic Regression 11](#_Toc140006786)

[IMBALANCE FITTING 13](#_Toc140006787)

[FLAN MODEL 15](#_Toc140006788)

[DataFrameChatBot 17](#_Toc140006789)

[Conclusion 18](#_Toc140006790)

[References 19](#_Toc140006791)

# ­­Introduction

In this project we aimed to do an in-depth analysis of the dataset of Women’s Clothing E-Commerce Reviews from Kaggle. Throughout our whole project we covered various subjects, such as EDA analysis, topic extraction, summarization, sentiment analysis and tried to create simple versions of chatbots. We had considered also doing a tending analysis, but this proved out to be impossible since our dataset did not contain any column for the time tracking. So, through our python code snippet we tried to cover the entire data science pipeline, starting from loading and cleaning our dataset, making an extensive EDA (exploratory data analysis) so we can identify and understand the dataset and what it talks about, then we proceeded with extracting the topic of the reviews, making a summarization of them, performing a sentiment analysis, trying to evaluate the sentiment analysis through a Logistic Regression classifier, applying a question answering model and more specifically experimenting with the FLAN model from google and finally creating chat bot for data frame interactions. Through the following extensive report, we will unravel and try to explain as much as possible all the processes that we have encapsulated in our code snippets, trying to illuminate the underpinning reasoning behind each method.

# Main Body

## Data Loading & Preprocessing

### Data Loading

Every data-driven task starts by acquiring the data, Firstly, we started by importing the required Python libraries, and continued to fill the necessary one’s throughout the code completion, such as pandas, NumPy, seaborn, nltk, sklearn, and many more others. Then we proceeded with downloading the Vader lexicon, which has to do with sentiment analysis, and it was used to assign sentiment scores to the reviews that we had (Hutto & Gilbert, 2014). For loading the dataset that we have obtained through Kaggle, we used the read\_csv function from a python library called pandas, which is well known for its ease to handle structured data, cleaning it, handling missing values and converting the desired data types according to your preferences.

### Data Preprocessing

So, the data preprocess stage started by inspecting the first few rows trying to identify missing values. Then we proceeded with removing the rows of the dataset that contained missing review text, so we could be certain about the completeness of the data. Special attention was given to the column named ‘Review Text’, as it plays a vital role for our later analysis steps that we performed. So, as aforementioned, by using the dropna function of pandas, we completely cleared the dataset based on the particular column, and in that way, we ensured the availability of textual data that we were going to be needed for the effective sentiment analysis and topic extraction.

Then the data as we have them in their raw form, they are not ready for an analysis. So, we went on with several steps of organizing and cleaning our data so we can use them effectively and efficiently through our next NLP tasks. We used NLTK's RegexpTokenizer to achieve text tokenization, meaning to split the text into tokens, meaning individual words or punctuations. Then in order to ensure uniformity, we converted all our data into lower case. Then we proceeded with removing punctuations, which do not play a vital role in the sentiment or topic analysis. Furthermore, we removed all the common ‘stop words’ that are not usual playing a significant meaning (such as “the”, “and”, “a”, etc.). Finally, we used lemmatization, which is a technique towards reducing the words down to their base root form (like for example “playing” becomes “play”). This aids us not only at grouping similar words together but also by reducing the dimensionality of the data.

## Exploratory Data Analysis

The exploratory data analysis process or EDA is the investigation of a given dataset in order to understand its trends, its characteristics and maybe some form of patterns (Amaratunga et al., 2009). Through this stage we are able to identify any possible anomalies in the dataset, get hidden insights and set the foundations for model building and subsequent analysis, helping us that way to refine our general approach. After identifying columns and get a description of them, we proceeded with plotting various things with Matplotlib and Seaborn python libraries so we could get a holistic approach.

**Age Distribution of Reviewers**

We began our EDA by investigating age distribution that the reviewers of our dataset have. The histogram function of Matplotlib was very suitable for this, because it provided immediate insights into the most actively writing reviews age group. For a business, gaining this kind of understanding could be vital, because the demographics and the overall understanding of how the different age groups interact with their e-commerce platform are some of the basic criteria used to navigate their marketing strategies and product offerings.

A graph of age distribution

Description automatically generated

**Most Reviewed Clothing Items**

Then we proceeded with analyzing the most reviewed items. Through this we reveal which are the most popular products in our dataset that the customers seem to interact the most with. The popularity of those items could be due to various factors, which are not contained in our dataset, but again this is quite important in the business context for the marketing and inventory strategic efforts.

A bar graph with different colored squares

Description automatically generated

**Reviews by Clothing Type and Department**

Next, we moved forward by analyzing through various visualizations the distribution of the reviews per clothing item and also the number of reviews by department and clothing type. This analysis aims to understand and inform us in a more detailed manner the customer engagement behavior.

A graph showing the number of reviews

Description automatically generated

A graph of a number of reviews

Description automatically generated

**Best-rated Products**

In our python code snippet, we also examine the products that are best rated but at the same time that have more than ten reviews. Through this step we aim to understand which products are more successful based on customer rating.

A bar chart of different colors

Description automatically generated

**Word and Punctuation Count in Reviews**

Last into the EDA section, we investigated the average worg and punctuation count in the reviews, sorted descending by their rating. This could help us get an isnight on whether the reviewers who are submitting a high or low rating, tend to writ longer and more complex reviews. Through this procedure, we aim towards understanding whether a customer who might be leaving a longer and more detailed review makes it more possible to be a negative one, in which they will express their complains.

A graph of a bar graph

Description automatically generated

Despite the fact of understanding and giving us a better overview of the dataset, this analysis could be also very helpful for businesses, adjusting their product strategy accordingly based on the insights that they have on customer ratings and reviews.

## Sentiment Analysis

As for the sentiment analysis, we applied the vader lexicon to derive the sentiments hidden behind its review. The vader sentiment analyzer, which is part of the NLTK library, is a rule based tool used for extracting sentiment from texts containing mixed ones’ or oral language. Also in the cases of sarcasms, slang expressions, emojis etc.,which are often spotted in texts like reviews or social media, vader performs pretty well. The way that analyzer works is by assigning a ‘compound”score between -1 and +1, where the - indicates the negative sentiment, close to 0 indicates neutral sentiment, and + indicates positive sentiment (Medhat et al., 2014). So, it categorizes them into the 3 aforementioned categories, assigning them a new rating based on their sentiment score. Furthermore we went on printing the highest positive and negative ones. Thinking out of the box, a sentiment analysis could be proved very useful for business cases, to understand their reputation and adjust their sales strategy accordingly. This analysis opens the way for a quantitative way to measure and to compare the sentiments derived from the reviews of the customers throughout the dataset.

A screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

## Topic Extraction

Topic extraction is an essential text analysis technique that is used to group or cluster text data based on their themes or “topics”. For identifying the main themes of the reviews and extracting hidden “abstract” topics from text data, we proceeded with applying topic modeling techniques. In our script, we used two different methods or modeling algorithms, the Latent Dirichlet Allocation (LDA) and the Non-Negative Matrix Factorization (NMF) (Karl et al., 2015). The difference between those two is that why work on different mathematical principles but have the same goal, which is to uncover the hidden thematic structure in the column corpus. From both of the two models used, we printed the top 10 topics that we extracted in order to help us identify common themes or the topics that the customers are discussing in their reviews. This whole process provides a general understanding of the primary themes that customers tend to discuss in their reviews, allowing businesses that way to not only address common issues but also to be able to capitalize on widespread praises.

### Topic Extraction with LDA

The LDA model assumes that in each review in our dataset, there is a mixture of a certain number of topics and that each word in that specific review is attributable to one of the general topics of the column (Jelodar et al., 2018). On the other hand, NMF model performs dimensionality reduction and works better when the matrix to factorize (tf-idf matrix, in our specific case) has no negative elements. NMF has an advantage over probability-based techniques because it can be used together with the 'tf-idf' model, which often results in a better performance. Finally, another difference of LDA is that it is a probabilistic method, which means that the output may differ throughout different executions of the code, in comparison with NMF where it is deterministic, meaning that it will provide the same output for the same input independently of how many times we executed our code snippet.

A screen shot of a computer

Description automatically generated

### Topic Extraction with NMF

So, as aforementioned we proceeded with using also Non-negative Matrix Factorization (NMF) for topic extraction. NMF is a dimensionality-reduction and unsupervised learning technique. The general idea is to factorize the given matrix into two non-negative matrices, capturing patterns in the original one. In order to create a review-term matrix, we use the TF-IDF vectorizer from the library of Scikit-learn (Nugumanova et al., 2022). That way we converted the text data into a matrix of TF-IDF features, assigning importance to words that are frequent in the review column. This way we managed to encapsulate the importance and relevance of words in a particular review.

A screen shot of a computer

Description automatically generated

## Text Summarization

Text Summarization is an NLP technique that it can shorten long pieces of information and at the same time maintain its original context, meaning and information. The need for that arises because of the abundance of data. So, for example if a review is lengthy, reading throughout all of them might be impractical and not so time and cost effective for a business, so that is where a summarization tool becomes handy, by condensing these reviews and offers a brief gist of the customer’s opinion, enabling business to quickly understand the main points of each one.

In order to generate a summarized version of the review texts, we used the BART transformer-based model (Bidirectional and Auto-Regressive Transformers), obtained from the Hugging Face's transformers library which provides numerous pre-trained models used for NLP tasks and it is developed by Facebook (Chen & Song, 2021). This model is a denoising autoencoder, meaning that it is trained to reconstruct original text after “noise” integration. It is also fine-tuned for downstream and generation tasks, such like text summarization, where you can prompt it a sample text and then the summarized version is displayed.

A screenshot of a computer screen

Description automatically generated

## Sentiment Analysis | Logistic Regression

In the final step of this part of our analysis, we created a script that builds a machine learning model in order to predict the sentiment of a review. The logic behind this is to create an automated system that will be able to categorize new or unseen reviews, based on their sentiment. For that reason, we used a combination of TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer and a classifier and as a text feature extraction method, more specifically Logistic Regression. The TF-IDF vectorizer, as aforementioned, converts the text into a matrix of TF-IDF features, and then assigns ‘importance’ to words that are most frequent in a specific text piece but not across the entire data, which in our case is a column. Also, for hyper parameter tunning of the TfidfVectorizer we used the GridSearchCV technique, as to identify the best values. Through this process we achieve weightage, which can often provide a better performance in tasks like text classification (Ramadhan et al., 2017). On the other hand, in our machine learning pipeline, the Logistic Regression model is simple, but yet an effective algorithm, for the purpose of binary and multi-class classification problems. The choice of going with Logistic regression was done due to the fact that it can handle sparse data like the ones derived from TF-IDF. It derived scores from TextBlob as the targeted variables, and as predictors the features from TF-IDF. It is useful when it comes to predictions of the probability of an instance belonging to the default class, which can be then converted into binary or multi-class classifications, for two classes or more. So, by applying thresholds to the probabilities that were predicted (for example those with probability greater than 0.5 should be considered as positive), we did obtain the predicted sentiments. In order to evaluate our model’s performance, we proceeded with some standard performance measurements like a plot of the confusion matrix, a classification report, accuracy score, precision, recall and F1 score. That way we were able to identify the True Positive, the False Positive, the True Negative and the False negative values for all the three sentiment categories that we had (negative, neutral, and positive). So, by looking at its performance metrics we can predict the sentiment of reviews and noticed that it didn’t work so well as we wanted to, so we proceeded with the next step.

A screenshot of a computer screen

Description automatically generated

A diagram of a comparison of a number of blue squares

Description automatically generated

## IMBALANCE FITTING

So, as we observed from the Linear Regression results above, we can see that there were significantly more instances of positive and neutral reviews compared to the negative ones, in our dataset. So, our model tended to be biased toward predicting the two classes with the more instances of appearance. To wrap things up, in this specific code block we performed both pre-processing and modeling in order to take reviews and handle class imbalance issue (having to do with negative one’s). We have done so, by using a combination of the three techniques: TF-IDF for text pre-processing, random oversampling as to handle class imbalance, and logistic regression for binary classification modeling (Barr et al., 2022). Through the various strategies that we could implement to handle this matter, we proceeded with the one called up-sampling minority class. More specifically what this technique does is that it adds copies of instances of the minority class in the training dataset, increasing that way the instances of that specific class (Barr et al., 2022). Then we used a confusion matrix again, with measures of exactness and completeness like Precision and Recall but also AUC-ROC score to see how much capable our model is to make the distinguish between the two classes, to show basically a table with the correct and the incorrect predictions that our model have performed.

Using Penalized Models: Use models that allow for the weighting of classes like Support Vector Machines or Logistic Regression.

More specifically we will proceed by listing all the steps that we did alongside with what the code does and name the techniques that we used:

* We started by importing all the necessary libraries for this part.
  + TfidfVectorizer, obtained from sklearn.feature\_extraction.text. This as aforementioned is a numerical statistical tool used in reflect the importance that a word has to a document in a collection.
  + RandomOverSampler, from imblearn.over\_sampling. This technique was used in order to handle the imbalanced in our ecommerce dataset. We worked with up sampling method in order to enhance the strength of our minority class.
  + Counter, from collections. A dictionary subclass used for counting hash able objects. In other words, it is a collection where elements are stored as the keys of the dictionary and their counts are stored as the values of that dictionary.
* Then we proceeded with initializing the TfidfVectorizer. We initialized it with a maximum of 5000 features (or unique words).
* Then we fitted and transformed the TfidfVectorizer on our training ecommerce dataset. The specific function used learns the global term weights of the idf vector through the fit step, and then it proceeds with applying that into the transform step. In the last step it also applies scaling of sublinear tf type, to be more specific it goes forward and replace tf with 1 + log(tf).
* Then using the same vectorizer we transformed our test data into TF-IDF features. But here, we did not use the fit\_transform function because we choosed to use the idf that was learned from the training data set.
* Here is where the RandomOverSampler makes its appearance. With upscaling or up sampling our negative reviews minority class, we tried to eliminate the bias of our model as much as possible and that way not to harm the accuracy of the model’s prediction about our minority class. This sampler achieves that by making the representation of the minority class higher in our dataset.
* Then we proceeded with fitting again a Logistic Regression Model on the sampled data. We chose as solver parameter the liblinear one, because when it comes to binary classification and especially for small datasets, it is a considered a good one. Then for the class\_weight parameter we used the balanced one, which was because we wanted to automatically adjust our weights, but in inverse of the proportion of our class frequencies.
* Then for the prediction and the evaluation part we used the now fitted logistic regression model. By using accuracy\_score() function we printed the respected accuracy of the predictions of our model.
* Further on we print the classification report in which we displayed the following:
  + Recall score, which is the ability of the classifier to locate all the samples that are positive.
  + The precision, which is the ability of our classifier not to label a negative sample as positive.
  + The F1 score for our model, which is the harmonic mean of recall but also precision.
* Finally, we print out the Confusion Matrix, which is a table used for performance description reasons of a classification models on a specific dataset, in which we know it’s true values. We plotted the table through a function that we have named as plot\_confusion\_matrix and it revealed in which aspect was our classification model confused.

A screenshot of a black screen

Description automatically generated

A diagram of a confusion matrix

Description automatically generated

## 

## FLAN MODEL

Then in our notebook we included a question-answering model, that uses the transformer model named FLAN T5 base. The model either takes a code snippet and a question as input or you can directly accept questions and then moves on into generating an answer (Varshney et al., 2023). Not only for the purpose of research, but also for practical implementation, this could be very handy in various applications where is required to understand and asnwer questions based on text data. After experimenting and trying the small version also, we ended up going with the base model and not the medium one, because of the large storage that the model required and the heavy computational power that it asked for. In that specific code block, we implemented the aforementioned model in order to answer questions based rather on a given text or without using an input from us, and then through the transformer’s models in the Hugging Face library come up with an answer. According to Varshney et al. (2023) here is a more detailed step by step breakdown in what we exactly did:

* Imported all the libraries that we needed for the FLAN model to run. That libraries were the AutoModelForSeq2SeqLM and the AutoTokenizer from the transformer’s library.
* Then we instantiated the model with the following command: AutoModelForSeq2SeqLM.from\_pretrained("google/flan-t5-base")
* Respectively the tokenizer with the following command: AutoTokenizer.from\_pretrained("google/flan-t5-base")
* So, the "google/flan-t5-base" is loaded from the Hugging Face’s model repository, and we proceeded with the choice of the base one due to storage and computational issues. The bigger models that the repository had required a lot of disk space and more computational power that we could afford in order to see results in “real-time”.
* Then we moved forward defining a function with the name answer\_question, which is responsible for taking reading a text snippet that we provide according to our liking and also a question as input. So, based on the text we provided the model with the goal is to answer the question. The flow of the whole process that is taking place is the following:
  + The whole input of the model is firstly created, through the combination of our text input and the question that we have set for the model.
  + Then through instantiated tokenizer, our whole input package is tokenized. Then this package is truncated to the specific length of 512 tokens, which is the limit that the FLAT base model gave us, for smaller models the tokens are fewer and for larger respectively bigger.
  + Then all that results that are now tokenized are fed into the model. Then for the generation the model uses beam search, and more specifically with a width of 5 for decoding. This is a heuristic searching algorithm that is used for exploration of the most promising nodes of the input package. At each step it tries to reduce the risk of not getting the optimal path, by keeping track of the best most promising ones.
  + Furthermore, by the use of the tokenizer the model decodes back into text its output, which is a sequence of tokens ids.
  + Finally, the answer that has been decoded in printed in a conversational manner/format.
* To sum up, the last step that we did is to use the answer\_question function to generate/print the answer to the question that we have fed the model with.

This is an example on our input text and asking the model to answer on that:

A computer screen with red text

Description automatically generated

This is an example of just questioning without our input, and just asking of the FLAN to answer:

A screenshot of a computer program

Description automatically generated

## DataFrameChatBot

In the last part of our code script, we defined a Python class, named **DataFrameChatBot**, that acts as a chatbot interface in order to achieve some interaction with our panda’s data frame, Women’s Clothing E-commerce. This could be proved very useful for data analysis purposes where a user can get information about the data and its structure in a conversational manner and the same time offering the ease that in the case that the imported dataset changes then minor changes are needed for the code to be executed correctly on the new data. So, here is a summarization on how we separated that part of our code snippet into 3 main parts:

1. **Initialization (init):** It first reads and takes a pandas data frame as input, and then saves it as a class attribute. Then it loads the **en\_core\_web\_sm** model from SpaCy, and then proceeds with assigning it to the class attribute named **nlp**. This is the model that it is going to be used for natural language processing tasks.
2. **Information Retrieval Methods:** In this class we defined several methods, in order to retrieve information from the our imported data frame, like the shape of the data frame, the columns that it consists of, the type of specific columns, the number of null values contained in a column, the mean or the sum or the max or the min calculated on specific columns, the number of the unique values that a column has, and many more. We also implemented methods in order to check whether a column does exist in our dataset, if a column has null values in it, and finally to obtain the most frequent category of a column.
3. **Question Processing (process\_question):** This is our core or main function if you like of our manual chatbot. It receives as an input a question and then uses the SpaCy model, in order to parse the question. Then depending on the words and phrases that the user concludes in the question, it calls the relevant method of combination of multiple one’s, in order to retrieve the required information from the data frame.

Finally, using the process\_question function that we have defined, we process the question, retrieve the answer from the chatbot and finally print it. This is kind of unique and extraordinary way for a user to get and understand the information that is needed in a unique way though interacting or speaking if we can say, directly with the data frame.

A screenshot of a computer screen

Description automatically generated

# Conclusion

In conclusion, the code provides a holistic approach on a complete data science pipeline, as to understand specifically the customer reviews in the dataset that we are investigating. From the little baby steps of data cleaning and EDA analysis, we systematically reached the stage of applying and integrating into our code snippet machine learning models and other deep learning techniques, towards achieving the sentiment analysis, text summarization and topic extraction. Also, another thing worth mentioning will be not only the use of the FLAN model, and demonstrating how transformer models can harness not only on our own data but also in general questions that we asked, but also the intriguing deployment of a data frame chat bot.

Through our analysis on the dataset, we revealed valuable insights, offering a methodology that could be easily applied to other similar tasks, which may involve text data analysis. On the other hand, we understand that we tailored the whole process to fit the specific need of our project, meaning that for a real-life example more advanced and complex techniques could definitely be used, but those will require more computational resources.

Finally, this whole pipeline gives a starting point of how a business can take advantage of customer reviews to extract meaning and actionable insights. They can better understand their customers, satisfy their need, and inevitably drive growth, by interpreting key sentiments and summarizing key topics. The insights generated by this report and code snippet, can be used as a template compass for a more informed strategic decision making, opening the path towards business evolution and success. By applying the project in a bigger time scale, we could have enhanced our exploration into more sophisticated analytical techniques.

# References

Amaratunga, D., Cabrera, J., Fernholz, L. T., & Morgenthaler, S. (2009). Exploratory data analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, *1*(1), 33–44. <https://doi.org/10.1002/wics.2>

Barr, J. T., Sobel, M., & Thatcher, T. (2022). *Upsampling, a comparative study with new ideas*. <https://doi.org/10.1109/icsc52841.2022.00059>

Chen, Y., & Song, Q. (2021). *News Text Summarization Method based on BART-TextRank Model*. <https://doi.org/10.1109/iaeac50856.2021.9390683>

Hutto, C. J., & Gilbert, E. (2014). Vader: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. *Proceedings of the International AAAI Conference on Web and Social Media*, *8*(1), 216–225. <https://doi.org/10.1609/icwsm.v8i1.14550>

Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2018). Latent Dirichlet allocation (LDA) and topic modeling: models, applications, a survey. *Multimedia Tools and Applications*, *78*(11), 15169–15211. <https://doi.org/10.1007/s11042-018-6894-4>

Karl, A. T., Wisnowski, J. W., & Rushing, W. H. (2015). A practical guide to text mining with topic extraction. *Wiley Interdisciplinary Reviews: Computational Statistics*, *7*(5), 326–340. <https://doi.org/10.1002/wics.1361>

Medhat, W., Hassan, A. E., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. <https://doi.org/10.1016/j.asej.2014.04.011>

Nugumanova, A., Akhmed-Zaki, D., Mansurova, M., Baiburin, Y., & Maulit, A. (2022). NMF-based approach to automatic term extraction. *Expert Systems With Applications*, *199*, 117179. <https://doi.org/10.1016/j.eswa.2022.117179>

Ramadhan, W. P., Novianty, S. T. M. T. A., & Setianingsih, S. T. M. T. C. (2017). *Sentiment analysis using multinomial logistic regression*. <https://doi.org/10.1109/iccerec.2017.8226700>

Varshney, N., Parmar, M., Patel, N., Handa, D., Sarkar, S., Luo, M., & Baral, C. (2023). Can NLP Models Correctly Reason Over Contexts that Break the Common  Assumptions? *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2305.12096>