****

**Women’s E-Commerce Clothing Review Analysis**

Course: ALY 6040 Data Mining Applications

By,

Shruti Avinash Pawar

Shashan Mukesh Pande

**Instructor: Amin Karimpour**

16th May 2020

Author Note

This Assignment is created under the guidance of the above-mentioned professor and is demonstrating natural language processing and sentiment analysis on the data set provided.

**Introduction**

NLP is conventionally used for the analysis of emotions and sentiments. It involves analyzing the review text to understand what is expressed by the person writing it. The sentiment in the text is calculated using polarity which tags the comment as positive or negative (Sarkar, 2018). With growing competition in the clothing section of e-commerce, it is very essential to take note of what the customers think of the product. Also, the company should be aware of the improvement area in terms of how the product satisfaction can be improved for the customer. Thus, we aim to do an analysis of reviews of clothes purchased by women and build a model to predict if the product will be recommended or not based on the reviews received.

**Methodology**

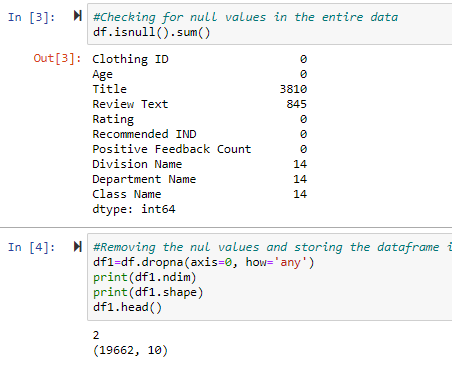
The data that will be used for review analysis is web-scrapped from an e-commerce website and was available on Kaggle. The data consists of 23486 rows and 10 variable columns such as age, rating, recommended IND, review text, etc. The tool used to process and analyze this acquired data is ‘python’ and the packages used are NumPy, Pandas, Scikit learn, Matplotlib, Seaborn, Textblob, and NLTK.

**Analysis**

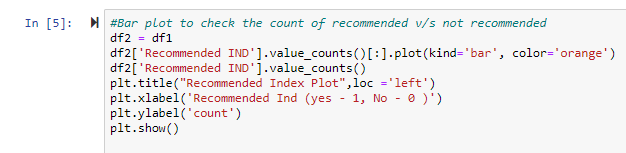
The first step of the review analysis process was data cleaning and EDA. The data was extracted from a csv file to a pandas data frame and below was the output in python:



The head() function in python gives a gist of the data as seen in the snip. We used ‘isnull function’ and found that there are 3810 null values in Title and 845 null values in review text. These null values will add no value to the analysis; thus, we have excluded those from the data frame. This leaves 19662 rows available in the data frame for analysis. Below is the snip of the code and output:



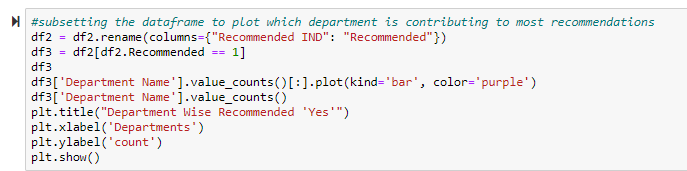
Further, to understand the data variables, we performed Exploratory Data Analysis and Visualization of the data. A bar plot was created using matplotlib and below was the code and output:

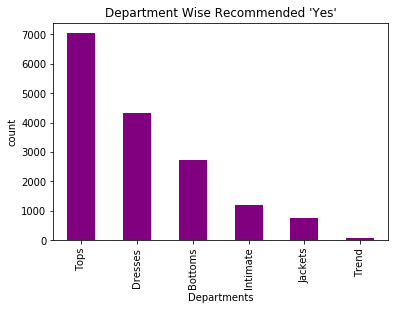




**Figure 1:** Recommended Yes/No plot with bars representing count of reviewers

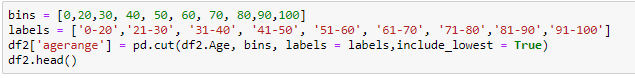
Figure 1 shows that there are more than 15000 reviewers have recommended the product and about 4000 have not recommended the product. Amongst these 150000 reviewers, we checked department wise split using the below code:



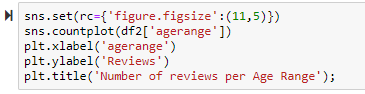


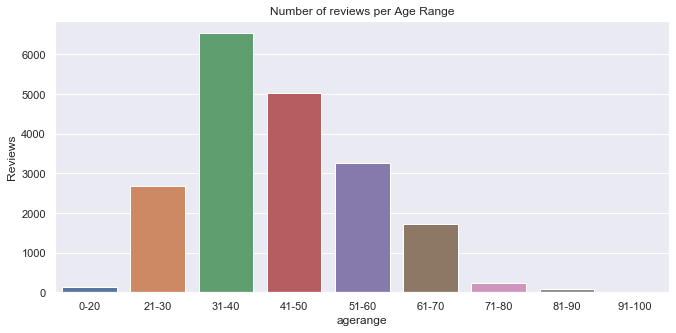
**Figure 2:** Department wise Recommended Yes with bars representing count of reviewers

In figure 2 we can see that the most recommended product department is Tops whereas there are low recommendations for Jackets and trends clothing. These departments are the focus area for the company to enhance the quality of the product. Further, we used the below code to classify the age into classes called agerange:



We then used seaborn library to plot reviews per age range

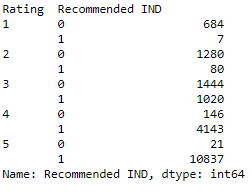




**Figure 3:** Number of reviews per age class with bars representing count of reviewers

In figure 3 it is seen that majority of the reviews are given by women in the age class of 31 to 40 followed by 41 to 50 and 51 to 60. It is very surprising that the age class 21 to 30 which is supposedly considered to be active on social media, have given lesser reviews as compared to the other age classes. We now took a rating wise count of recommendations and found the below output:

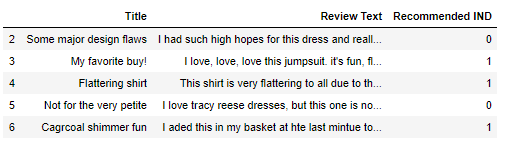




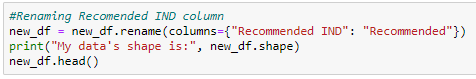
A high number of reviews with high product ratings is justified however 7 people who have rated the product 1 and 80 who have rated 2 have recommended the product.

Our target variables for NLP feature extraction are Title, Review text, and recommended IND. We, therefore, made a new data frame with these variables to start the feature extraction process.





In order to get all the text in one variable column, we combined the Title and Review text column and named the new variable as review.

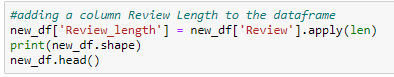


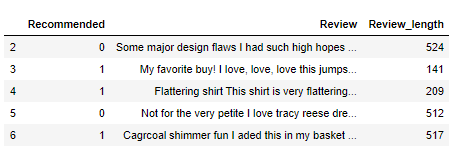
It is important to rename the ‘Recommendation IND’ to ‘Recommended’ as a column label; in order to simplify the nomenclature.

A screenshot of a cell phone

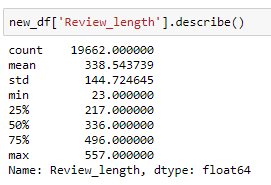
Description automatically generated

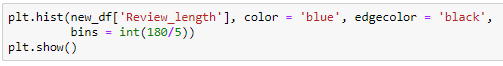
Moreover, now the string is defined, we would like to assign some numerical data in order to tangibly process data strings into a model in order to define the dataset at large.



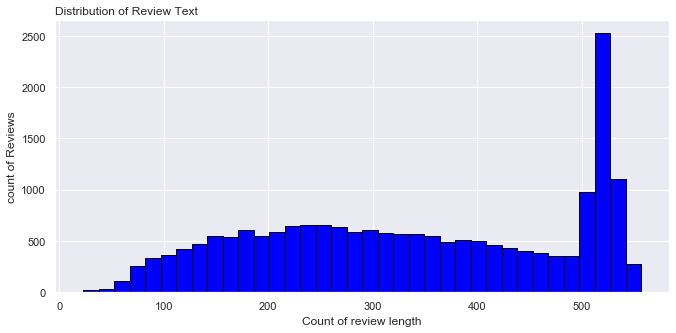


The very first step to assign numerical value to a categorical set is via the length function, as defined in the code above. Here using the apply(len) function we are counting characters in the Review String.





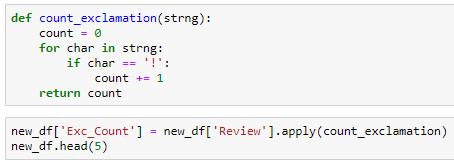
Here, we are doing descriptive analysis of count of ‘Review length’. We see that the mean character count is 338 words. This states that there are a lot of customers writing reviews on this E-commerce portal. At the same time, it shows the sincerity of how these recommendations are leading to consumer’s buying behavior. Now that, we have figured out the data is important and leading to revenue generation, we will try to segregate the data and assign bash values by doing further processes on the text.

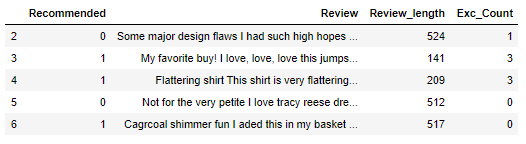


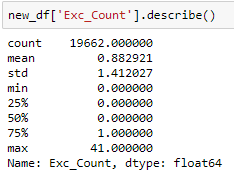
**Figure 4 :** Distribution of Review Length with bars representing count of reviewer

Figure 4 shows how the review length is distributed all across and there are more than 2500 customers who have review with a character length of 500 words and beyond. Hence, we are now assuming that people are either very happy with the products and that’s why they are recommending buying; or they are extremely disappointed and hence they are describing in length as to why not to buy the product.

Exclamations are important as they provide emphasis on a statement, this demarcates as to if the person is happy with the product or sad with the product.







We have defined a function and assigned ‘for’ and ‘nested if’ loop in count function order to find the number of exclamations each review string contains. Also, we have described these counts and have found that one comment has 41 exclamations in total. Below is the distribution of the count of exclamation in a review string.



****

**Figure 5 :** Plot for count of Exclamations with bars representing count of reviewer

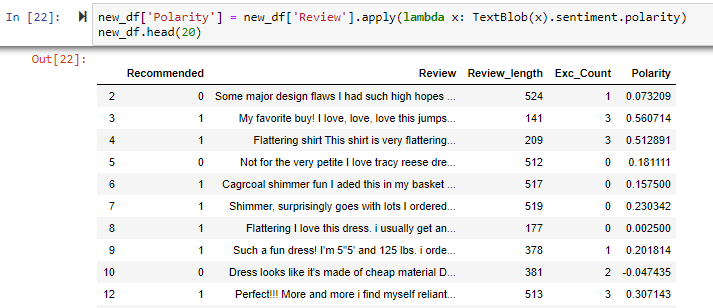
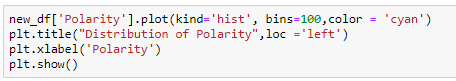
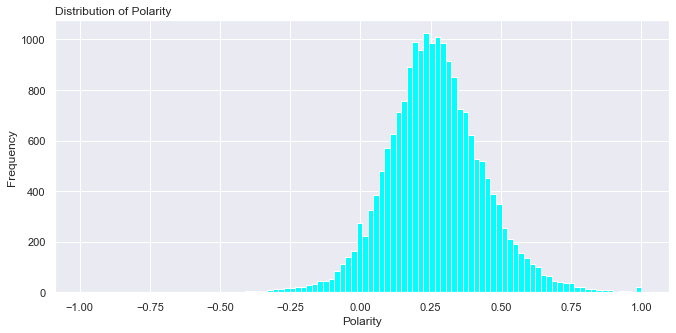


Figure 5 shows that maximum comments don’t have any exclamation marks, hence it is difficult for us to identify the emphasis given in a particular statement. Next, let’s start with polarity. Polarity defines the negative and positive word string with the negative trending from -1 to 0 and the positive trending from 0 to 1. With the help of a Text Blob function which is part of the ntlk package we are able to assign polarity value to the entire string set.

With figure 6, we are able to identify that the maximum number of reviews have a mean polarity of 0.25 which is tending towards the positive side. Therefore, we can say that the maximum number of reviews are positive.





**Figure 6 :** Distribution of polarity

Moreover, we see that there are a lot of punctuations in the string set, hence its removal will help us clean the data a bit better. Hence, we have defined string.punctuation to define what do we mean by punctuations and then we remove the punctuations by defining a function.

A screenshot of a social media post

Description automatically generated

We see that when we apply the above function, the string has no punctuations. This is like the statements now do not make any sense and they are just mere words in a string set.

A screenshot of a cell phone

Description automatically generatedNext, we see that the above code takes a lot of time in processing, this is because the adj collector function that we have created used the word\_tokenize function, which basically extracts tokens from these string of characters and then used tuples to pick out verbs and adjectives, and for a data set. This review set now has the verbs and adjectives that we require to streamline.

Now, we use have a string full of adjective and verbs, we see that there are some words like I, you, again, etc. in it. We need to get rid of these. For this, we use stop word-extraction again by using the word\_tokenize variable.

A screenshot of a social media post

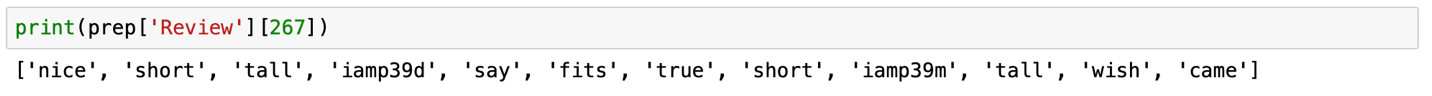
Description automatically generated

We see that there are multiple stop words that require removal. But we need to be cautious as these stop words might remove some words that we might require. Hence, we create a list of these variable as clothes\_list and we make sure that these words are not removed along with stop words. After applying the function, we see that new string doesn’t have any stop words.

**A screenshot of a social media post

Description automatically generated**

Let’s test, suppose if we want to see for the review 267, are we getting the desired list.

****

Now, that we have made sure that string of character set is fine, we can proceed further with the analysis. In the reviews, people generally put numerical values of size, removing that is essential.

**A screenshot of a cell phone

Description automatically generated**

With the help of rem\_nos, we are able to remove the numbers included in these reviews. With this, we move further to stemming, which is also called lemmatization.

**A screenshot of a cell phone

Description automatically generated**

Every word is either a lemma or a combination of a lemma and a suffix or a prefix. These lemma needs to be identified as these can remove redundancy of word meaning in a sentence. Hence Stemming is removing a stem and leaving behind the root word, which is also called the lemma. Like for example, fluttering, flutters, fluttered have the same root as flutter. Hence, flutter becomes the stem and we keep fluttered as flutter. With the help of Porter Stemmer, we are able to stem the words in the reviews and then take out all the stem words and just keep the roots in the data set. Therefore, we favorite become favorit.

A screenshot of a cell phone

Description automatically generated

A very important part is to identify which are positive words and negative words. These can be identified by a word cloud. Also, a word cloud is important, as we can also identify the frequency of a word by identify the size of the word in the display. We will be identifying the two categories of recommendation separately. 1, being the positive and the 0, being the negative.

A screenshot of a cell phone

Description automatically generated

Below are the positives and negative charts:

**A close up of a newspaper

Description automatically generated**

**Figure 7: Word Cloud denoting positive words**

**Figure**

**A screenshot of a cell phone

Description automatically generated**

**A close up of a newspaper

Description automatically generated**

**Figure 8: Word Cloud denoting negative words**

We see that there is ‘love’ both in positive and negative words, and the frequency is high in both the variation. Hence, now we get a sense that the model might be not that accurate as the love seems to be a positive word and not a negative word.

**A screenshot of a social media post

Description automatically generated**

Turing the entire process to a numeric front, we start vectorizing the words. Bag of words is a phrase that denotes a bunch of words in a set. Here, we will refer the set as a review. With the help of bow transformer, we will be creating a corpus and then we are tokenizing the text and building a vocabulary of words in total. Hence, we see if we lock the 6th review, we get the details of a corpus.

**A screenshot of a social media post

Description automatically generated**

We are creating an example of the locked review and vectorizing the string set based on bash numbers, the right-side column denotes the frequency in the string set.

Furthermore, we are applying the transformer in the review’s column. and after updating the values, we are using sparse matrix to store 237857 stored elements.

A screenshot of a social media post

Description automatically generated

TF-IDF, Term Frequency Inverse Document Frequency, simply states that a particular word occurring in a statement water review upon the number of words in a string set multiplied by that particular word occurring in the entire document set that we have; basically, the collection of reviews. This Transformer actually converts a vectorized value of the corpus into a bash number giving it a unique value. The statistical explanation is explained as follows:

**A picture containing knife

Description automatically generated**

This bash numbers are unique to every word in the review and can be looked up using a bow transformer vocabulary. Once the bash number is identified using the TF-IDF method, we assign the transformer to the entire ‘review’ labeled column.

**A screenshot of a social media post

Description automatically generated**

Once, this is done, we will be merging the sparse matrix with other variables to form a matrix containing all bash values of these vectorizing corpus.

**A screenshot of a cell phone

Description automatically generated**

Finally, we have all the categorical values of the Review column in numerical form.

Now, we will be merging these values into the original data set. A screenshot of a social media post

Description automatically generated

Now, as we have merged these data sets together. Now we split the data into two sets. One containing the ‘recommendations’ column and the other containing the entire data set.

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a cell phone

Description automatically generated**

We now have the entire data set apart from the recommended column. After describing this, we get the mean and maximum value of the data set, which can be comparable to the previous value.

**A screenshot of a cell phone

Description automatically generated**

We are now dividing the data set into two columns the train and the test data set. We see that values in the output above, and finds that the train data set of the Y(recommendation part) have more positive recommendations, same as that with the test data of Y. Below is the code used to fit a logistic regression model:

A screenshot of a social media post

Description automatically generated

Confusion matrix of the logistic regression model output is as follows:

|  |  |  |
| --- | --- | --- |
|  | Actual Yes | Actual No |
| Predicted yes | True Positive | False Positive |
|  | 201 | 695 |
| Predicted No | False Negative | True Negative |
|  | 124 | 3919 |

**Table 1:** Logistic regression model accuracy summary

|  |  |  |
| --- | --- | --- |
| **Head** | **Formula** | **Percentage** |
| Model Accuracy | TP+TN /TP+TN+FP+FN | 83.41% |
| Recall | TP/TP+FN | 61.84% |
| Precision | TP/TP+FP | 22.43% |
| F1 Score | 2 \* (P \* R / P+R) | 32.91% |

As per the accuracy calculations in table 1, we can see that the model accuracy is 83.41% with a precision of 22.43% and recall of 61.84%. For the model to be balanced between precision and recall, F1 score should approach 100%, however, the F1 score for our logistic regression model is low. ("Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures - Exsilio Blog", 2016)

ROC curve is used to find a good threshold to distinguish the decision as a yes or no. AUC i.e Area Under the ROC curve is a measure of how well the model can distinguish between the yes or no decision. The higher the AUC the better is the model (Schoonjans, 2020).

**A close up of a map

Description automatically generated**

**Figure 9 :** ROC curve and AUC for logistic regression Model

In figure 9, we can see that the AUC for the logistic regression model as per the micro average ROC curve is 0.91.

Further, we fit a random forest using the below code to check if it is a better model to derive product recommendation.**A screenshot of a cell phone

Description automatically generated**

Confusion matrix of the Random Forest model Output is as follows:

|  |  |  |
| --- | --- | --- |
|  | Actual Yes | Actual No |
| Predicted yes | True Positive | False Positive |
|  | 587 | 309 |
| Predicted No | False Negative | True Negative |
|  | 917 | 3126 |

**Table 2:** Random Forest Model accuracy summary

|  |  |  |
| --- | --- | --- |
| **Head** | **Formula** | **Percentage** |
| Model Accuracy | TP+TN /TP+TN+FP+FN | 75.17% |
| Recall | TP/TP+FN | 39.02% |
| Precision | TP/TP+FP | 65.51% |
| F1 Score | 2 \* (P \* R / P+R) | 48.90% |

If we compare accuracy for both the models from table 1 and 2, logistic regression model has a higher accuracy whereas random forest has a higher precision and F1 score. Thus, our deciding factor for a better model will be the AUC.**A close up of a map

Description automatically generated**

**Figure 10 :** ROC curve and AUC for Random Forest Model

In figure 10, we can see that the AUC for the Random Forest model as per the micro average ROC curve is 0.81. Logistic regression with AUC of 0.91 is a better model as compared to the Random forest. Now that we have identified that the logistic regression model is a better fit, we can increase the accuracy by adding/dropping significant features.

**Conclusion**

To recapitulate, below are a few recommendations based on the analysis for the company:

* Department Intimates, Jackets and Trends are the focus departments with the least recommendations and the company can check for the key negative words in the review column of the dataframe to check why the product is perceived as low quality and what are the areas of improvement.
* 21 to 50 is the age class that provides most reviews, thus the company can perks to this age class in the form of review reward points in order to acquire better data, which will be the input in the implemented logistic model in form of a TD – IDF sparse matrix.
* With the output of the logistic regression model, the company can anticipate the demand for a product if it is predicted as recommended, which can be used for inventory management decisions.

**Citation**

Sarkar, D. (2018). A Practitioner's Guide to Natural Language Processing (Part I) — Processing & Understanding Text. Retrieved 15 May 2020, from <https://towardsdatascience.com/a-practitioners-guide-to-natural-language-processing-part-i-processing-understanding-text-9f4abfd13e72>

Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures - Exsilio Blog. (2016). Retrieved 16 May 2020, from <https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/>

Schoonjans, F. (2020). ROC curve analysis with MedCalc. Retrieved 16 May 2020, from <https://www.medcalc.org/manual/roc-curves.php>

**Link to the dataset:** <https://www.kaggle.com/nicapotato/womens-ecommerce-clothing-reviews>