Text

Description automatically generated

**CSE380 - NATURAL LANGUAGE PROCESSING**

**CA4 Report**

**Title- Sentiment Analysis on Women Clothing Review**

**Team Details**

|  |  |  |
| --- | --- | --- |
| **Sno** | **Unique ID** | **Name** |
| 1 | E0119067 | Vidarshana Govilesh |
| 2 | E0119056 | Bhavadharani.E |
| 3 | E0119046 | Gokul.P |

**Marks Split-up**

|  |  |  |
| --- | --- | --- |
| **Split up** | **Max Marks** | **Marks Scored** |
| **Subject Knowledge and documentation** | 5 |  |
| **Individual Contribution** | 5 |  |
| **Dataset pre-processing** | 3 |  |
| **Result** | 2 |  |
| **Total** | **15** |  |

**PROBLEM DESCRIPTION**

As online marketplaces have been popular during the past decades, the online sellers and merchants ask their purchasers to share their opinions about the products they have bought. As a result, millions of reviews are being generated daily which makes it difficult for a potential consumer to make a good decision on whether to buy the product. Analysing this enormous amount of opinions is also hard and time consuming for product manufacturers. This considers the problem of classifying reviews by their overall semantic (positive or negative) and decide if a customer would recommend the product or not.

**INTRODUCTION ABOUT THE PACKAGES/ ML MODELS/ OTHER TOOLS YOU HAVE USED**

First we try to understand the data and its distribution with basic EDA with the help of Pandas and Matplotlib libraries.

# NLTK

NLTK stands for Natural Language Toolkit. This toolkit is one of the most powerful NLP libraries which contains packages to make machines understand human language and reply to it with an appropriate response. Tokenization, Stemming, Lemmatization, Punctuation, Character count.

Using nltk we can determine the mood of text, NLTK’s N-grams, and word2vec.

**TEXTBLOB**

 Text Blob, It uses NLTK (Natural Language Toolkit) and the input contains a single sentence, The output of Text Blob is polarity and subjectivity. Polarity score lies between (-1 to 1) where -1 identifies the most negative words such as ‘disgusting’, ‘awful’, ‘pathetic’, and 1 identifies the most positive words like ‘excellent’, ‘best’. Subjectivity score lies between (0 and 1), It shows the amount of personal opinion, If a sentence has high subjectivity i.e. close to 1, It resembles that the text contains more personal opinion than factual information.

**N-GRAM**

An N-gram means a sequence of N words. Natural Language Processing, or NLP for short, n-grams are used for a variety of things. Some examples include auto completion of sentences (such as the one we see in Gmail these days), auto spell check (yes, we can do that as well), and to a certain extent, we can check for grammar in a given sentence. N-grams are useful for turning written language into data and breaking down larger portions of search data into more meaningful segments that help identify the root cause behind trends.

**TENSORFLOW**

TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations.

**SKLEARN**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon **NumPy, SciPy** and **Matplotlib**.

**Re**

A Regular expression is a sequence of characters that define a search pattern mainly for use in pattern matching with strings or string matching , i.e. find and replace like operations. Regular expressions are a generalized way to match patterns with sequence of characters.

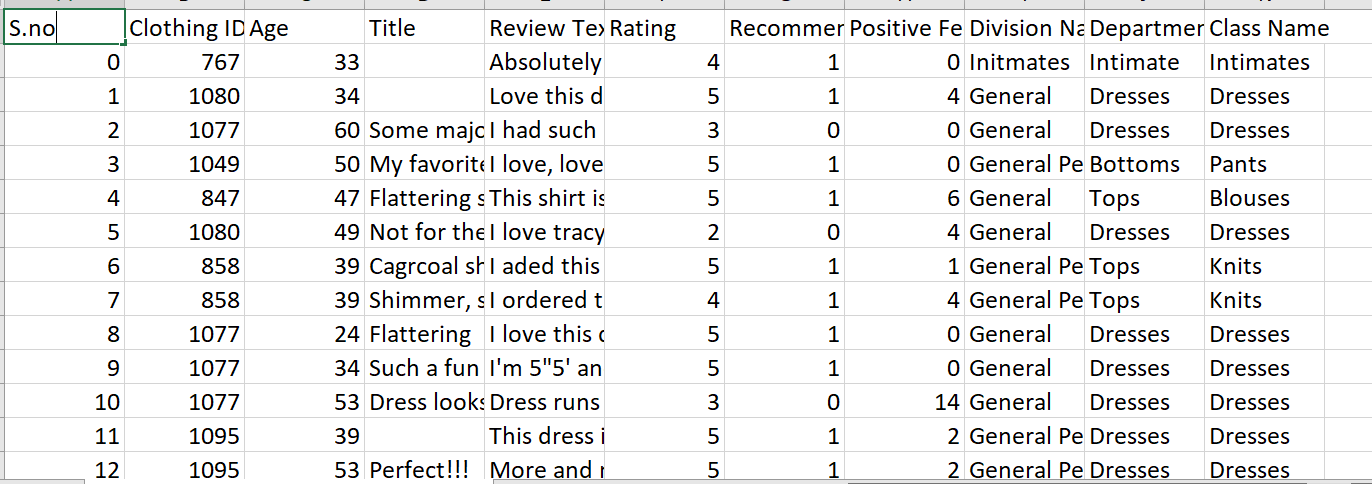
**DESCRIPTION ABOUT THE DATASET**

# Features present :

* Clothing ID: Integer Categorical variable that refers to the specific piece being reviewed.
* Age: Positive Integer variable of the reviewers age.
* Title: String variable for the title of the review.
* Review Text: String variable for the review body.
* Rating: Positive Ordinal Integer variable for the product score granted by the customer from 1 Worst, to 5 Best.
* Positive Feedback Count: Positive Integer documenting the number of other customers who found this review positive.
* Division Name: Categorical name of the product high level division.
* Department Name: Categorical name of the product department name.
* Class Name: Categorical name of the product class name.

Target :

* Recommended IND: Binary variable stating where the customer recommends the product where 1 is recommended, 0 is not recommended.



**CODE**

from tqdm import tqdm

for i **in** tqdm(range(len(X))):

review = re.sub('[^a-zA-z]',' ',X['Review Text'][i])

review = review.lower()

review = review.split()

ps = PorterStemmer()

review =[ps.stem(i) for i **in** review if **not** i **in** set(stopwords.words('english'))]

review =' '.join(review)

corpus.append(review)

corpus[0:5]

import tensorflow as tf

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

tokenizer = Tokenizer(num\_words = 3000)

tokenizer.fit\_on\_texts(corpus)

sequences = tokenizer.texts\_to\_sequences(corpus)

padded = pad\_sequences(sequences, padding='post')

word\_index = tokenizer.word\_index

count = 0

for i,j **in** word\_index.items():

if count == 11:

break

print(i,j)

count = count+1

embedding\_dim = 64

model = tf.keras.Sequential([

tf.keras.layers.Embedding(3000, embedding\_dim),

tf.keras.layers.GlobalAveragePooling1D(),

tf.keras.layers.Dense(6, activation='relu'),

tf.keras.layers.Dense(1, activation='sigmoid')

])

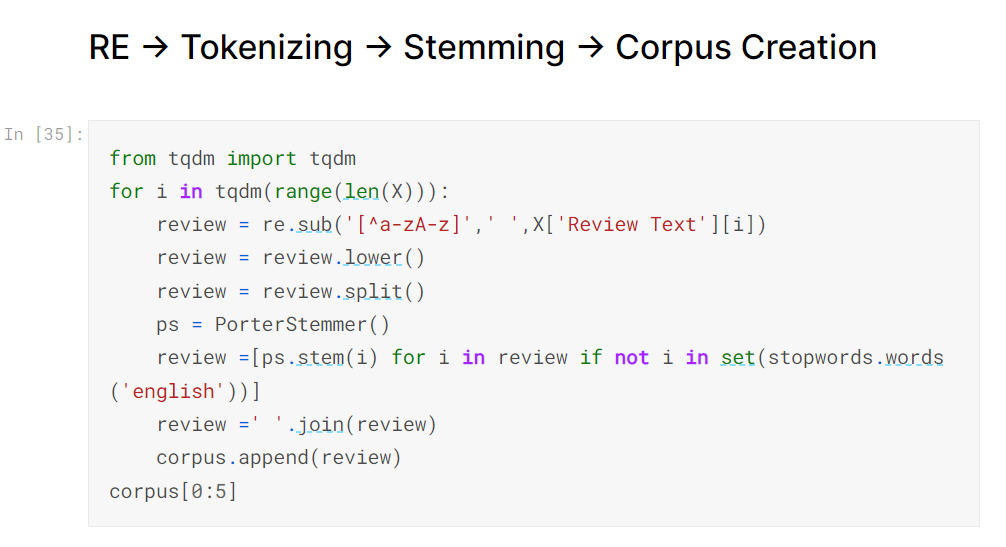
model.summary()

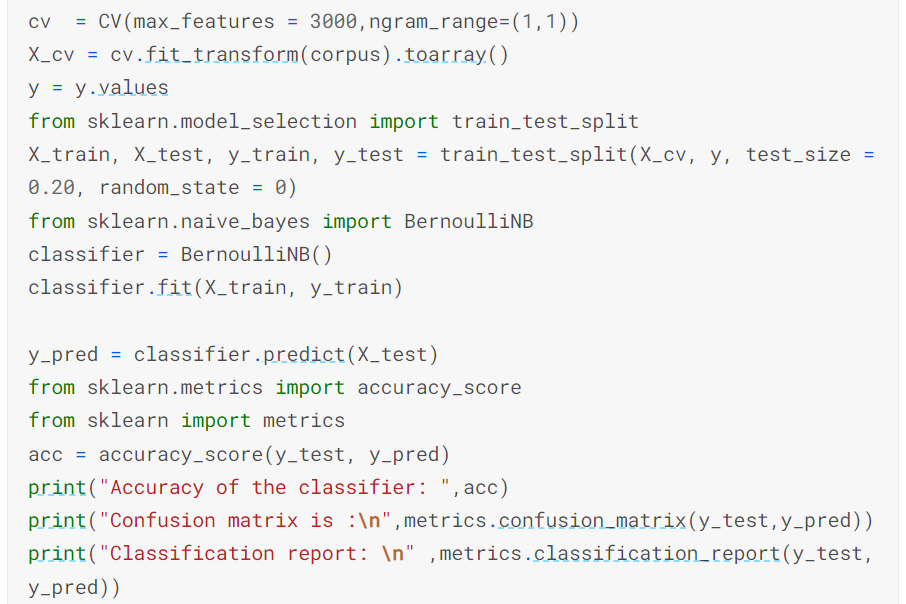
num\_epochs = 10

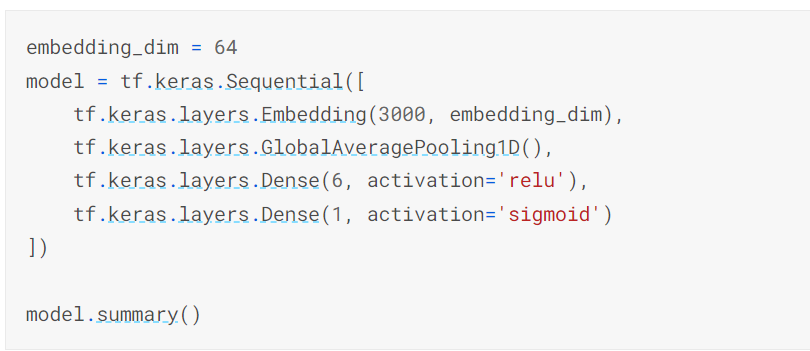
model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

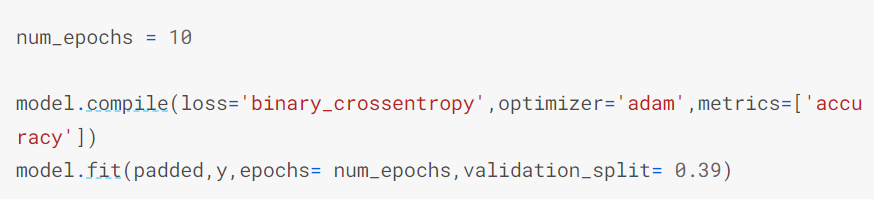
model.fit(padded,y,epochs= num\_epochs,validation\_split= 0.39)

**SCREENSHOTS**









**RESULTS**

