Megha Banerjee (SUID: 301610297)

Reetodeep Hazra (SUID: 675984954)

IST 736: Text Mining Project Code

E-commerce Product Description Classification

```
import time, psutil, os

# Data manipulation
import numpy as np
import pandas as pd

# Plotting and visualization
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import seaborn as sns
sns.set_theme()
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)

# NLP
import string, re, nltk
from string import punctuation
from nltk.tokenize import word_tokenize, RegexpTokenizer
from nltk.corpus import stopwords
!pip install num2words
from num2words import num2words
!pip install pyspellchecker
from spellchecker import SpellChecker
from nltk.stem.porter import PorterStemmer
import spacy
from nltk.stem import WordNetLemmatizer

# TF-IDF
from sklearn.feature_extraction.text import TfidfVectorizer

# Scipy
import scipy
from scipy import sparse
from scipy.sparse import csr_matrix

# Train-test split and cross validation
from sklearn.model_selection import train_test_split, ParameterGrid

# Classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import RidgeClassifier
from xgboost import XGBClassifier
from sklearn.ensemble import AdaBoostClassifier

# Model evaluation
from sklearn import metrics
from sklearn.metrics import accuracy_score

# Others
import json
import gensim
from sklearn.decomposition import TruncatedSVD
```

```
from sklearn.decomposition import TruncatedSVD
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting num2words
  Downloading num2words-0.5.12-py3-none-any.whl (125 kB)
     |████████████████████████████████| 125 kB 26.6 MB/s
Collecting docopt>=0.6.2
  Downloading docopt-0.6.2.tar.gz (25 kB)
Building wheels for collected packages: docopt
  Building wheel for docopt (setup.py) ... done
  Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl size=13723 sha256=98f67e96cd77eeba46f883c9d
  Stored in directory: /root/.cache/pip/wheels/56/ea/58/ead137b087d9e326852a851351d1debf4ada529b6ac0ec4e8c
Successfully built docopt
Installing collected packages: docopt, num2words
Successfully installed docopt-0.6.2 num2words-0.5.12
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyspellchecker
  Downloading pyspellchecker-0.7.0-py3-none-any.whl (2.5 MB)
     |████████████████████████████████| 2.5 MB 19.1 MB/s
Installing collected packages: pyspellchecker
Successfully installed pyspellchecker-0.7.0
```

```
# Loading and customizing the data
data = pd.read_csv('/content/ecommerceDataset.csv',
    names = ['label', 'description']
)
data = data[['description', 'label']]

print(pd.Series({"Memory usage": "{:.2f} MB".format(data.memory_usage().sum()/(1024*1024)),
                 "Dataset shape": "{}".format(data.shape)}).to_string())
data
```

```
Memory usage        0.77 MB
Dataset shape    (50425, 2)
```

	description	label
0	Paper Plane Design Framed Wall Hanging Motivat...	Household
1	SAF 'Floral' Framed Painting (Wood, 30 inch x ...	Household
2	SAF 'UV Textured Modern Art Print Framed' Pain...	Household
3	SAF Flower Print Framed Painting (Synthetic, 1...	Household
4	Incredible Gifts India Wooden Happy Birthday U...	Household
...	...	...
50420	Strontium MicroSD Class 10 8GB Memory Card (Bl...	Electronics
50421	CrossBeats Wave Waterproof Bluetooth Wireless ...	Electronics
50422	Karbonn Titanium Wind W4 (White) Karbonn Titan...	Electronics
50423	Samsung Guru FM Plus (SM-B110E/D, Black) Colou...	Electronics
50424	Micromax Canvas Win W121 (White)	Electronics

50425 rows × 2 columns

```
data.skew()
```

```
<ipython-input-3-b3b431164adb>:1: FutureWarning:

Dropping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is deprecated; in a future version

Series([], dtype: float64)
```

```
# Example description
data['description'].iloc[5]
```

```
'Pitaara Box Romantic Venice Canvas Painting 6mm Thick Mdf Frame 21.1 X 14Inch Enclosure Material:MDF Mount Frame
\xa0|\xa0                          Size:21.1inch x 14inch (53.6cms x 35.6cms)   SIZE: 21.1 inch x 14.0 inch (53.6
of your room walls with this breathtaking digital printed artwork. Our high-end printing technology captures every
n enhanced matte painting canvas, ensuring rich and lively colours. This wall art panel is mounted on MDF and read
or home d©cor artwork gifts for Living, Dining Room, Outdoor, Gallery, Hotels, Restaurants, Office, Reception, Kit
```

```
# Missing values and duplicate observations
print(pd.Series({"Number of observations with missing values": len(data) - len(data.dropna()),
                 "Number of duplicate observations": data.duplicated().sum()}).to_string())
```

```
Number of observations with missing values        1
Number of duplicate observations              22622
```

```
data.dropna(inplace = True) # Dropping observations with missing values
data.drop_duplicates(inplace = True) # Dropping duplicate observations
data.reset_index(drop = True, inplace = True) # Resetting index
```

The labels are manually encoded with the following scheme:

Electronics: 0

Household: 1

Books: 2

Clothing & Accessories: 3

```
# Manual encoding of labels
label_dict = {'Electronics': 0, 'Household': 1, 'Books': 2, 'Clothing & Accessories': 3}
data.replace({'label': label_dict}, inplace = True)

print(pd.Series({"Memory usage": "{:.2f} MB".format(data.memory_usage().sum()/(1024*1024)),
                 "Dataset shape": "{}".format(data.shape)}).to_string())
data
```

```
Memory usage        0.42 MB
Dataset shape    (27802, 2)
```

	description	label
0	Paper Plane Design Framed Wall Hanging Motivat...	1
1	SAF 'Floral' Framed Painting (Wood, 30 inch x ...	1
2	SAF 'UV Textured Modern Art Print Framed' Pain...	1
3	SAF Flower Print Framed Painting (Synthetic, 1...	1
4	Incredible Gifts India Wooden Happy Birthday U...	1
...	...	...
27797	Micromax Bharat 5 Plus Zero impact on visual d...	0
27798	Microsoft Lumia 550 8GB 4G Black Microsoft lum...	0
27799	Microsoft Lumia 535 (Black, 8GB) Colour:Black ...	0
27800	Karbonn Titanium Wind W4 (White) Karbonn Titan...	0
27801	Nokia Lumia 530 (Dual SIM, Grey) Colour:Grey ...	0

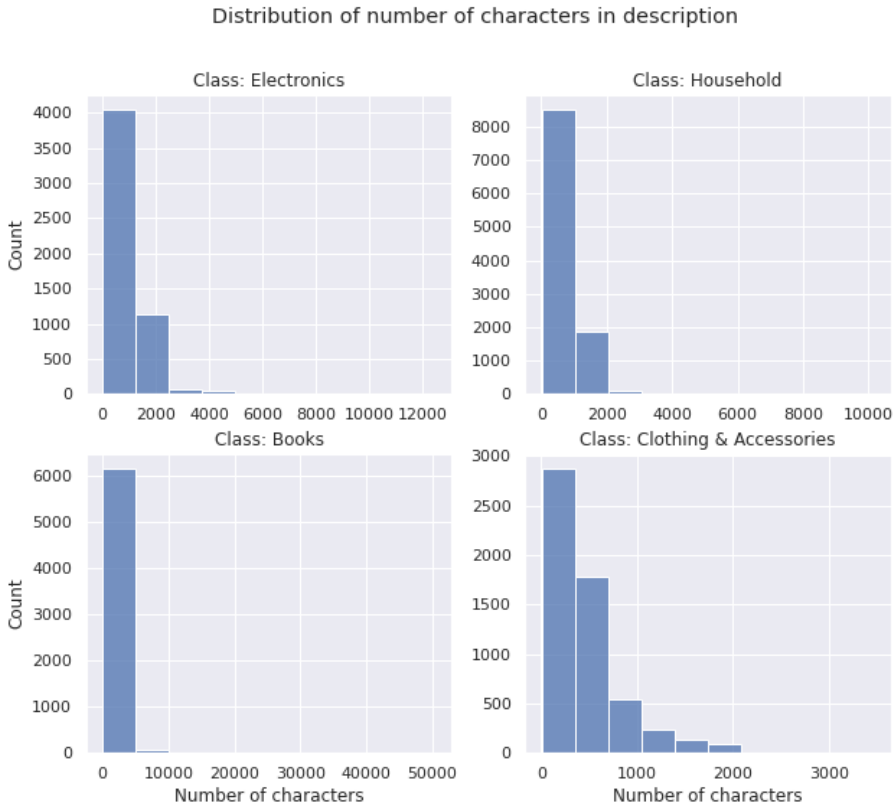
27802 rows × 2 columns

```
# Splitting the dataset by label
data_e = data[data['label'] == 0] # Electronics
data_h = data[data['label'] == 1] # Household
data_b = data[data['label'] == 2] # Books
data_c = data[data['label'] == 3] # Clothing & Accessories
```

```
# Visualization of class frequencies
values = np.array([len(data_e), len(data_h), len(data_b), len(data_c)])
labels = ['Electronics', 'Household', 'Books', 'Clothing & Accessories']
fig = go.Figure(data = [go.Pie(values = values, labels = labels, hole = 0.5, textinfo = 'percent', title = " ")])
```

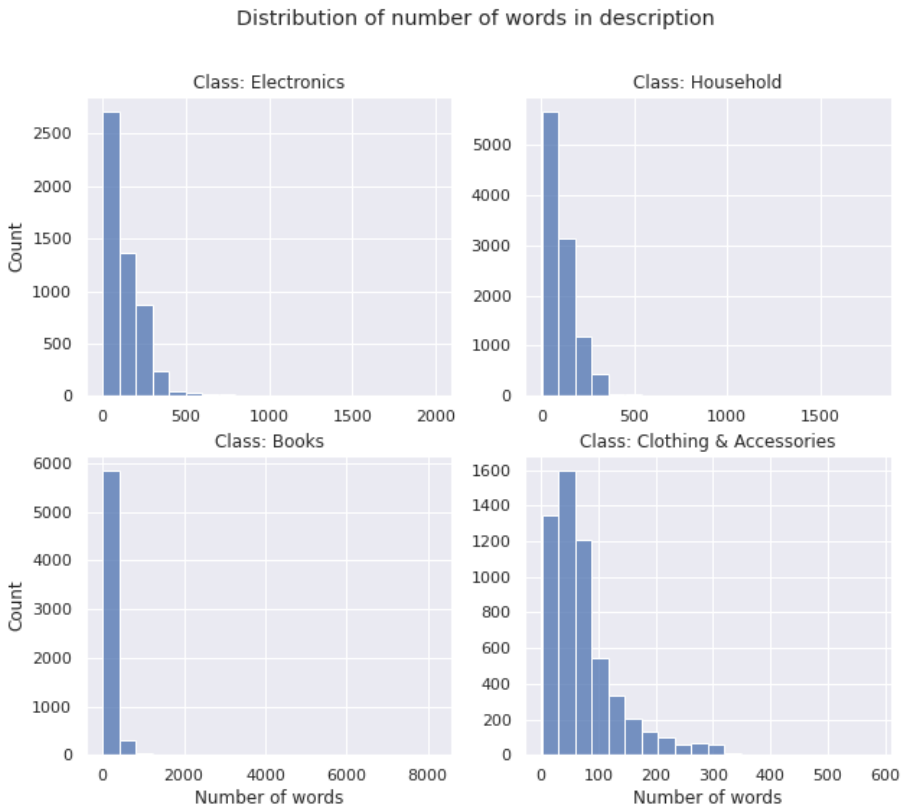
```
text_title = "Comparison of class frequencies"
fig.update_layout(height = 50, width = 80, showlegend = True, title = dict(text = text_title, x = 0.5, y = 0.95))
fig.show()
```

```
# Distribution of number of characters in description
data_e_char = data_e['description'].str.len()
data_h_char = data_h['description'].str.len()
data_b_char = data_b['description'].str.len()
data_c_char = data_c['description'].str.len()

fig, ax = plt.subplots(2, 2, figsize = (10, 8.4), sharey = False)
sns.histplot(x = data_e_char, bins = 10, ax = ax[0, 0]).set_title('Class: Electronics')
sns.histplot(x = data_h_char, bins = 10, ax = ax[0, 1]).set_title('Class: Household')
sns.histplot(x = data_b_char, bins = 10, ax = ax[1, 0]).set_title('Class: Books')
sns.histplot(x = data_c_char, bins = 10, ax = ax[1, 1]).set_title('Class: Clothing & Accessories')

fig.suptitle("Distribution of number of characters in description")
for i in range(4):
    ax[i // 2, i % 2].set_xlabel(" ") if i // 2 == 0 else ax[i // 2, i % 2].set_xlabel("Number of characters")
    if i % 2 != 0: ax[i // 2, i % 2].set_ylabel(" ")
```

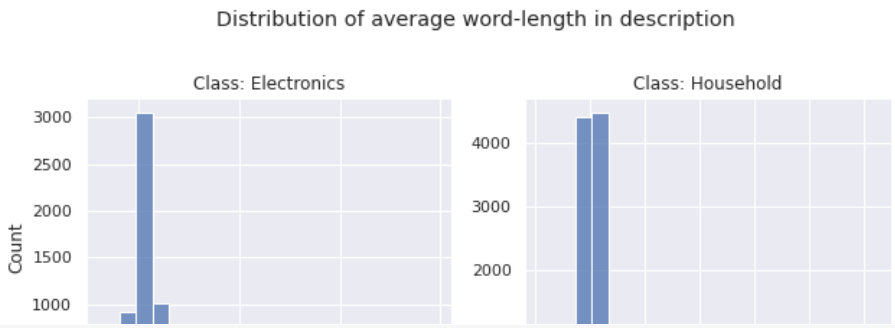



```
# Distribution of number of words in description
data_e_word = data_e['description'].str.split().map(lambda x: len(x))
data_h_word = data_h['description'].str.split().map(lambda x: len(x))
data_b_word = data_b['description'].str.split().map(lambda x: len(x))
data_c_word = data_c['description'].str.split().map(lambda x: len(x))

fig, ax = plt.subplots(2, 2, figsize = (10, 8.4), sharey = False)
sns.histplot(x = data_e_word, bins = 20, ax = ax[0, 0]).set_title('Class: Electronics')
sns.histplot(x = data_h_word, bins = 20, ax = ax[0, 1]).set_title('Class: Household')
sns.histplot(x = data_b_word, bins = 20, ax = ax[1, 0]).set_title('Class: Books')
sns.histplot(x = data_c_word, bins = 20, ax = ax[1, 1]).set_title('Class: Clothing & Accessories')

fig.suptitle("Distribution of number of words in description")
for i in range(4):
```

```
    ax[i // 2, i % 2].set_xlabel(" ") if i // 2 == 0 else ax[i // 2, i % 2].set_xlabel("Number of words")
    if i % 2 != 0: ax[i // 2, i % 2].set_ylabel(" ")
```




```
# Distribution of average word-length in description
data_e_avg = data_e['description'].str.split().apply(lambda x : [len(i) for i in x]).map(lambda x: np.mean(x))
data_h_avg = data_h['description'].str.split().apply(lambda x : [len(i) for i in x]).map(lambda x: np.mean(x))
data_b_avg = data_b['description'].str.split().apply(lambda x : [len(i) for i in x]).map(lambda x: np.mean(x))
data_c_avg = data_c['description'].str.split().apply(lambda x : [len(i) for i in x]).map(lambda x: np.mean(x))

fig, ax = plt.subplots(2, 2, figsize = (10, 8.4), sharey = False)
sns.histplot(x = data_e_avg, bins = 20, ax = ax[0, 0]).set_title('Class: Electronics')
sns.histplot(x = data_h_avg, bins = 20, ax = ax[0, 1]).set_title('Class: Household')
sns.histplot(x = data_b_avg, bins = 20, ax = ax[1, 0]).set_title('Class: Books')
sns.histplot(x = data_c_avg, bins = 20, ax = ax[1, 1]).set_title('Class: Clothing & Accessories')

fig.suptitle("Distribution of average word-length in description")
for i in range(4):
    ax[i // 2, i % 2].set_xlabel(" ") if i // 2 == 0 else ax[i // 2, i % 2].set_xlabel("Average word-length")
    if i % 2 != 0: ax[i // 2, i % 2].set_ylabel(" ")
```




```
# Feature-target split
X, y = data.drop('label', axis = 1), data['label']

# Train-test split (from complete data)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 40)
data_train = pd.concat([X_train, y_train], axis = 1)

# Validation-test split (from test data)
X_val, X_test, y_val, y_test = train_test_split(X_test, y_test, test_size = 0.5, random_state = 40)
data_val, data_test = pd.concat([X_val, y_val], axis = 1), pd.concat([X_test, y_test], axis = 1)

# Comparison of sizes of training set, validation set and test set
values = np.array([len(data_train), len(data_val), len(data_test)])
labels = ['Training set', 'Validation Set', 'Test set']
fig = go.Figure(data = [go.Pie(values = values, labels = labels, hole = 0.5, textinfo = 'percent', title = " ")])
text_title = "Comparison of sizes of training set, validation set and test set"
fig.update_layout(height = 500, width = 800, showlegend = True, title = dict(text = text_title, x = 0.5, y = 0.95))
fig.show()
```

```
# RegexpTokenizer
regexp = RegexpTokenizer("[\w']+")
```

```
# Converting to lowercase
def convert_to_lowercase(text):
    return text.lower()

text = "This is a FUNCTION that CoNvErTs a Text to lowercase"
print("Input: {}".format(text))
print("Output: {}".format(convert_to_lowercase(text)))
```

```
    Input: This is a FUNCTION that CoNvErTs a Text to lowercase
    Output: this is a function that converts a text to lowercase
```

```
# Removing whitespaces
def remove_whitespace(text):
    return text.strip()

text = " \t This is a string \t "
print("Input: {}".format(text))
print("Output: {}".format(remove_whitespace(text)))
```

```
    Input:           This is a string
    Output: This is a string
```

```
# Removing punctuations
def remove_punctuation(text):
    punct_str = string.punctuation
    punct_str = punct_str.replace("'", "") # discarding apostrophe from the string to keep the contractions intact
    return text.translate(str.maketrans("", "", punct_str))

text = "Here's [an] example? {of} &a string. with.? punctuations!!!!"
print("Input: {}".format(text))
print("Output: {}".format(remove_punctuation(text)))
```

```
    Input: Here's [an] example? {of} &a string. with.? punctuations!!!!
    Output: Here's an example of a string with punctuations
```

```
# Removing HTML tags
def remove_html(text):
    html = re.compile(r'<.*?>')
    return html.sub(r'', text)

text = '<a href = "https://www.kaggle.com/datasets/saurabhshahane/ecommerce-text-classification"> Ecommerce Text Classi
print("Input: {}".format(text))
print("Output: {}".format(remove_html(text)))
```

```
    Input: <a href = "https://www.kaggle.com/datasets/saurabhshahane/ecommerce-text-classification"> Ecommerce Text Cl
    Output:  Ecommerce Text Classification
```

```
# Removing emojis
def remove_emoji(text):
    emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", flags = re.UNICODE)
    return emoji_pattern.sub(r'', text)

text = "This innovative hd printing technique results in durable and spectacular looking prints 😊 "
print("Input: {}".format(text))
print("Output: {}".format(remove_emoji(text)))
```

```
    Input: This innovative hd printing technique results in durable and spectacular looking prints 😊
    Output: This innovative hd printing technique results in durable and spectacular looking prints
```

```
# Removing other unicode characters
def remove_http(text):
    http = "https?://\S+|www\.\S+" # matching strings beginning with http (but not just "http")
    pattern = r"({})".format(http) # creating pattern
    return re.sub(pattern, "", text)

text = "It's a function that removes links starting with http: or https such as https://en.wikipedia.org/wiki/Unicode_s
print("Input: {}".format(text))
print("Output: {}".format(remove_http(text)))
```

```
    Input: It's a function that removes links starting with http: or https such as https://en.wikipedia.org/wiki/Unico
```

```
Output: It's a function that removes links starting with http: or https such as
```

```
# Dictionary of acronyms
acronyms_url = 'https://raw.githubusercontent.com/sugatagh/E-commerce-Text-Classification/main/JSON/english_acronyms.js
acronyms_dict = pd.read_json(acronyms_url, typ = 'series')

print("Example: Original form of the acronym 'fyi' is '{}'".format(acronyms_dict['fyi']))
```

```
Example: Original form of the acronym 'fyi' is 'for your information'
```

```
# Dataframe of acronyms
pd.DataFrame(acronyms_dict.items(), columns = ['acronym', 'original']).head(20)
```

	acronym	original
0	aka	also known as
1	asap	as soon as possible
2	brb	be right back
3	btw	by the way
4	dob	date of birth
5	faq	frequently asked questions
6	fyi	for your information
7	idk	i don't know
8	idc	i don't care
9	iirc	if i recall correctly
10	imo	in my opinion
11	irl	in real life
12	lmk	let me know
13	lol	laugh out loud
14	ngl	not gonna lie
15	noyb	none of your business
16	nvm	never mind
17	ofc	of course
18	omg	oh my god
19	pfa	please find attached

```
# List of acronyms
acronyms_list = list(acronyms_dict.keys())
```

```
# Function to convert contractions in a text
def convert_acronyms(text):
    words = []
    for word in regexp.tokenize(text):
        if word in acronyms_list:
            words = words + acronyms_dict[word].split()
        else:
            words = words + word.split()

    text_converted = " ".join(words)
    return text_converted

text = "btw you've to fill in the details including dob"
print("Input: {}".format(text))
print("Output: {}".format(convert_acronyms(text)))
```

```
Input: btw you've to fill in the details including dob
```

```
Output: by the way you've to fill in the details including date of birth
```

```
# Dictionary of contractions
contractions_url = 'https://raw.githubusercontent.com/sugatagh/E-commerce-Text-Classification/main/JSON/english_contract
contractions_dict = pd.read_json(contractions_url, typ = 'series')

print("Example: Original form of the contraction 'aren't' is '{}'".format(contractions_dict["aren't"]))
```

```
Example: Original form of the contraction 'aren't' is 'are not'
```

```
# Dataframe of contractions
pd.DataFrame(contractions_dict.items(), columns = ['contraction', 'original']).head()
```

	contraction	original
0	'aight	alright
1	ain't	are not
2	amn't	am not
3	arencha	are not you
4	aren't	are not

```
# List of contractions
contractions_list = list(contractions_dict.keys())
```

```
# Function to convert contractions in a text
def convert_contractions(text):
    words = []
    for word in regexp.tokenize(text):
        if word in contractions_list:
            words = words + contractions_dict[word].split()
        else:
            words = words + word.split()

    text_converted = " ".join(words)
    return text_converted

text = "he's doin' fine"
print("Input: {}".format(text))
print("Output: {}".format(convert_contractions(text)))
```

```
Input: he's doin' fine
Output: he is doing fine
```

```
nltk.download('all')
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/alpino.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package book_grammars to
```

```
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown_tei.zip.
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_cat.zip.
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_esp.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]    | Downloading package comtrans to /root/nltk_data...
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package conll2007 to /root/nltk_data...
[nltk_data]    | Downloading package crubadan to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/crubadan.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dependency_treebank.zip.
[nltk_data]    | Downloading package dolch to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dolch.zip.
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     /root/nltk_data
```

```
# Stopwords
import nltk
stops = nltk.corpus.stopwords.words('english') # stopwords
addstops = ["among", "onto", "shall", "thrice", "thus", "twice", "unto", "us", "would"] # additional stopwords
allstops = stops + addstops

print(allstops)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your'
```

```
# Function to remove stopwords from a list of texts
def remove_stopwords(text):
    return " ".join([word for word in regexp.tokenize(text) if word not in allstops])

text = "This is a function that removes stopwords in a given text"
print("Input: {}".format(text))
print("Output: {}".format(remove_stopwords(text)))
```

```
Input: This is a function that removes stopwords in a given text
Output: This function removes stopwords given text
```

```
spell = SpellChecker()

def pyspellchecker(text):
    word_list = regexp.tokenize(text)
    word_list_corrected = []
    for word in word_list:
        if word in spell.unknown(word_list):
            word_corrected = spell.correction(word)
            if word_corrected == None:
                word_list_corrected.append(word)
            else:
                word_list_corrected.append(word_corrected)
        else:
            word_list_corrected.append(word)
    text_corrected = " ".join(word_list_corrected)
```

```
    return text_corrected

text = "I'm goinng therre"
print("Input: {}".format(text))
print("Output: {}".format(pyspellchecker(text)))
```

```
Input: I'm goinng therre
Output: I'm going there
```

```
# Stemming
stemmer = PorterStemmer()
def text_stemmer(text):
    text_stem = " ".join([stemmer.stem(word) for word in regexp.tokenize(text)])
    return text_stem

text = "Introducing lemmatization as an improvement over stemming"
print("Input: {}".format(text))
print("Output: {}".format(text_stemmer(text)))
```

```
Input: Introducing lemmatization as an improvement over stemming
Output: introduc lemmat as an improv over stem
```

```
# Lemmatization
spacy_lemmatizer = spacy.load("en_core_web_sm", disable = ['parser', 'ner'])
#lemmatizer = WordNetLemmatizer()

def text_lemmatizer(text):
    text_spacy = " ".join([token.lemma_ for token in spacy_lemmatizer(text)])
    #text_wordnet = " ".join([lemmatizer.lemmatize(word) for word in word_tokenize(text)]) # regexp.tokenize(text)
    return text_spacy
    #return text_wordnet

text = "Introducing lemmatization as an improvement over stemming"
print("Input: {}".format(text))
print("Output: {}".format(text_lemmatizer(text)))
```

```
Input: Introducing lemmatization as an improvement over stemming
Output: introduce lemmatization as an improvement over stem
```

```
# Discardment of non-alphabetic words
def discard_non_alpha(text):
    word_list_non_alpha = [word for word in regexp.tokenize(text) if word.isalpha()]
    text_non_alpha = " ".join(word_list_non_alpha)
    return text_non_alpha

text = "It is an ocean of thousands and 1000s of crowd"
print("Input: {}".format(text))
print("Output: {}".format(discard_non_alpha(text)))
```

```
Input: It is an ocean of thousands and 1000s of crowd
Output: It is an ocean of thousands and of crowd
```

```
def keep_pos(text):
    tokens = regexp.tokenize(text)
    tokens_tagged = nltk.pos_tag(tokens)
    #keep_tags = ['NN', 'NNS', 'NNP', 'NNPS', 'FW']
    keep_tags = ['NN', 'NNS', 'NNP', 'NNPS', 'FW', 'PRP', 'PRPS', 'RB', 'RBR', 'RBS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP',
    keep_words = [x[0] for x in tokens_tagged if x[1] in keep_tags]
    return " ".join(keep_words)

text = "He arrived at seven o'clock on Wednesday evening"
print("Input: {}".format(text))
tokens = regexp.tokenize(text)
print("Tokens: {}".format(tokens))
tokens_tagged = nltk.pos_tag(tokens)
print("Tagged Tokens: {}".format(tokens_tagged))
print("Output: {}".format(keep_pos(text)))
```

```
Input: He arrived at seven o'clock on Wednesday evening
```

```
Tokens: ['He', 'arrived', 'at', 'seven', "o'clock", 'on', 'Wednesday', 'evening']
Tagged Tokens: [('He', 'PRP'), ('arrived', 'VBD'), ('at', 'IN'), ('seven', 'CD'), ("o'clock", 'NN'), ('on', 'IN'),
Output: He arrived o'clock Wednesday evening
```

```
# Additional stopwords

alphabets = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "
prepositions = ["about", "above", "across", "after", "against", "among", "around", "at", "before", "behind", "below", "
prepositions_less_common = ["aboard", "along", "amid", "as", "beneath", "beyond", "but", "concerning", "considering", "
coordinating_conjunctions = ["and", "but", "for", "nor", "or", "so", "and", "yet"]
correlative_conjunctions = ["both", "and", "either", "or", "neither", "nor", "not", "only", "but", "whether", "or"]
subordinating_conjunctions = ["after", "although", "as", "as if", "as long as", "as much as", "as soon as", "as though"
others = ["ã", "å", "ì", "û", "ûªm", "ûó", "ûò", "ìñ", "ûªre", "ûªve", "ûª", "ûªs", "ûówe"]
additional_stops = alphabets + prepositions + prepositions_less_common + coordinating_conjunctions + correlative_conjun

def remove_additional_stopwords(text):
    return " ".join([word for word in regexp.tokenize(text) if word not in additional_stops])
```

```
def text_normalizer(text):
    text = convert_to_lowercase(text)
    text = remove_whitespace(text)
    text = re.sub('\n' , '', text) # converting text to one line
    text = re.sub('\[.*?\]', '', text) # removing square brackets
    text = remove_http(text)
    text = remove_punctuation(text)
    text = remove_html(text)
    text = remove_emoji(text)
    text = convert_acronyms(text)
    text = convert_contractions(text)
    text = remove_stopwords(text)
#     text = pyspellchecker(text)
    text = text_lemmatizer(text) # text = text_stemmer(text)
    text = discard_non_alpha(text)
    text = keep_pos(text)
    text = remove_additional_stopwords(text)
    return text

text = "We'll combine all functions into 1 SINGLE FUNCTION 🙂 & apply on @product #descriptions https://en.wikipedia.or
print("Input: {}".format(text))
print("Output: {}".format(text_normalizer(text)))
```

```
Input: We'll combine all functions into 1 SINGLE FUNCTION 🙂 & apply on @product #descriptions https://en.wikipedi
Output: combine function function apply product description
```

```
# Implementing text normalization
data_train_norm, data_val_norm, data_test_norm = pd.DataFrame(), pd.DataFrame(), pd.DataFrame()

data_train_norm['normalized description'] = data_train['description'].apply(text_normalizer)
data_val_norm['normalized description'] = data_val['description'].apply(text_normalizer)
data_test_norm['normalized description'] = data_test['description'].apply(text_normalizer)

data_train_norm['label'] = data_train['label']
data_val_norm['label'] = data_val['label']
data_test_norm['label'] = data_test['label']

data_train_norm
```

	normalized description	label
15525	approach acupuncture author author graduate al...	2
1536	nice leatherette office arm chair chair seat h...	1
21984	ekan fedora hat girl boy fedora hat hat man fe...	3
25056	concert showlightning controller lighting cont...	0
25213	marantz fully beltdrive premounte cartridge bu...	0

```
#tfidf
# Features and labels
X_train_norm, y_train = data_train_norm['normalized description'].tolist(), data_train_norm['label'].tolist()
X_val_norm, y_val = data_val_norm['normalized description'].tolist(), data_val_norm['label'].tolist()
X_test_norm, y_test = data_test_norm['normalized description'].tolist(), data_test_norm['label'].tolist()
```

```
# TF-IDF vectorization
TfidfVec = TfidfVectorizer(ngram_range = (1, 1))
X_train_tfidf = TfidfVec.fit_transform(X_train_norm)
X_val_tfidf = TfidfVec.transform(X_val_norm)
X_test_tfidf = TfidfVec.transform(X_test_norm)
```

```
#TF-IDF Baseline Modeling
# Classifiers
names = [
    "KNN Classifier",
    "Decision Tree",
    "Linear SVM",
    "Random Forest",
    "XGBoost",
]

models = [
    KNeighborsClassifier(n_neighbors = 149, n_jobs = -1),
    DecisionTreeClassifier(),
    svm.SVC(kernel = 'linear'),
    RandomForestClassifier(n_estimators = 100),
    XGBClassifier(),
]
```

```
# Function to return summary of baseline models
def score(X_train, y_train, X_val, y_val, names = names, models = models):
    score_df, score_train, score_val = pd.DataFrame(), [], []
    x = time.time()
    for model in models:
        model.fit(X_train, y_train)
        y_train_pred, y_val_pred = model.predict(X_train), model.predict(X_val)
        score_train.append(accuracy_score(y_train, y_train_pred))
        score_val.append(accuracy_score(y_val, y_val_pred))

    score_df["Classifier"], score_df["Training accuracy"], score_df["Validation accuracy"] = names, score_train, score_
    score_df.sort_values(by = 'Validation accuracy', ascending = False, inplace = True)
    return score_df
```

```
# Summary of baseline models
score(X_train_tfidf, y_train, X_val_tfidf, y_val, names = names, models = models)
```

```
# Hyperparameter tuning for linear SVM
svm_classifier = svm.SVC()
params_svm = {
    'kernel': ['linear'],
    'C': [0.1, 1, 10, 100]
}

best_model_svm, best_params_svm, best_score_svm, count = svm_classifier, ParameterGrid(params_svm)[0], 0, 0
for g in ParameterGrid(params_svm):
    time_start = time.time()
    count += 1
    print(f"Gridpoint #{count}: {g}")
    svm_classifier.set_params(**g)
    svm_classifier.fit(X_train_tfidf, y_train)
    y_train_pred, y_val_pred = svm_classifier.predict(X_train_tfidf), svm_classifier.predict(X_val_tfidf)
    score_train, score_val = accuracy_score(y_train, y_train_pred), accuracy_score(y_val, y_val_pred)
    time_stop = time.time()
    m, s = int(time_stop - time_start) // 60, int(time_stop - time_start) % 60
    print(f"Training accuracy: {score_train}, Validation accuracy: {score_val}, Runtime: {m}m{s}s")
    print(" ")
    if score_val > best_score_svm:
        best_params_svm, best_score_svm = g, score_val

best_model_tfidf, best_params_tfidf, best_score_tfidf = svm.SVC(), best_params_svm, best_score_svm
best_model_tfidf.set_params(**best_params_tfidf)
print(f"Best model: {best_model_tfidf}")
print(" ")
print(f"Best parameters: {best_params_tfidf}")
print(f"Best validation accuracy: {best_score_tfidf}")
```

```
Gridpoint #1: {'C': 0.1, 'kernel': 'linear'}
Training accuracy: 0.9354345577986601, Validation accuracy: 0.9251798561151079, Runtime: 2m38s

Gridpoint #2: {'C': 1, 'kernel': 'linear'}
Training accuracy: 0.9783283125758734, Validation accuracy: 0.9510791366906475, Runtime: 1m37s

Gridpoint #3: {'C': 10, 'kernel': 'linear'}
Training accuracy: 0.9982015197158401, Validation accuracy: 0.9449640287769784, Runtime: 1m42s

Gridpoint #4: {'C': 100, 'kernel': 'linear'}
Training accuracy: 0.99932556989344, Validation accuracy: 0.9392086330935252, Runtime: 1m40s

Best model: SVC(C=1, kernel='linear')

Best parameters: {'C': 1, 'kernel': 'linear'}
Best validation accuracy: 0.9510791366906475
```

```
from sklearn.svm import LinearSVC

svm_clf = LinearSVC(C=1, max_iter=2000)
svm_clf.fit(X_train_tfidf,y_train_pred)
feature_ranks = sorted(zip(svm_clf.coef_[0], TfidfVec.get_feature_names_out()))
top_10 = feature_ranks[-10:]
print("Top 10 features for Label 0 (Electronics)")
for i in range(0, len(top_10)):
    print(top_10[i])
print()
```

```
Top 10 features for Label 0 (Electronics)
(2.0698448187886522, 'tablet')
(2.156475619454373, 'printer')
(2.3020304507745935, 'lens')
(2.31944455281781, 'laptop')
(2.3693543247422584, 'phone')
(2.4405140320675374, 'projector')
(2.6354470046107696, 'gb')
(2.7451113485419767, 'speaker')
(2.8427752973451117, 'cable')
(2.957726119625348, 'camera')
```

```
from sklearn.svm import LinearSVC

svm_clf = LinearSVC(C=1, max_iter=2000)
svm_clf.fit(X_train_tfidf,y_train_pred)
feature_ranks = sorted(zip(svm_clf.coef_[1], TfidfVec.get_feature_names_out()))
top_10 = feature_ranks[-10:]
print("Top 10 features for Label 1 (Household)")
for i in range(0, len(top_10)):
    print(top_10[i])
print()
```

```
Top 10 features for Label 1 (Household)
(1.8106072750590345, 'iron')
(1.877189462605275, 'decoration')
(1.8878750801965147, 'pillow')
(2.087079720080107, 'oven')
(2.121320321949743, 'chair')
(2.197025070855988, 'vacuum')
(2.2213224145245327, 'steel')
(2.2467569796853692, 'bed')
(2.344309774496887, 'home')
(2.357733430581633, 'kitchen')
```

```
from sklearn.svm import LinearSVC

svm_clf = LinearSVC(C=1, max_iter=2000)
svm_clf.fit(X_train_tfidf,y_train_pred)
feature_ranks = sorted(zip(svm_clf.coef_[2], TfidfVec.get_feature_names_out()))
top_10 = feature_ranks[-10:]
print("Top 10 features for Label 2 (Books)")
for i in range(0, len(top_10)):
    print(top_10[i])
print()
```

```
Top 10 features for Label 2 (Books)
(1.4954454265715187, 'history')
(1.4959572759042539, 'snooker')
(1.6001931610097244, 'story')
(1.6038133579106855, 'science')
(1.6390564228538895, 'engineering')
(1.6857404933563946, 'edition')
(1.8196809618561487, 'guide')
(1.8233508215603684, 'review')
(2.6988994341598738, 'book')
(4.206443473342522, 'author')
```

```
from sklearn.svm import LinearSVC

svm_clf = LinearSVC(C=1, max_iter=2000)
svm_clf.fit(X_train_tfidf,y_train_pred)
feature_ranks = sorted(zip(svm_clf.coef_[3], TfidfVec.get_feature_names_out()))
top_10 = feature_ranks[-10:]
print("Top 10 features for Label 3 (Clothing and Accessories)")
for i in range(0, len(top_10)):
    print(top_10[i])
print()
```

```
Top 10 features for Label 3 (Clothing and Accessories)
(1.9174005661931068, 'sock')
(1.9845942616303909, 'cotton')
(2.221637889553609, 'waist')
(2.2999113351805094, 'sunglass')
(2.3033112385366303, 'boy')
(2.3431949732675452, 'men')
(2.471984229328264, 'bra')
(2.717018814156706, 'wear')
(2.8927472954528524, 'man')
(2.9540115613151823, 'woman')
```

```
#Word 2 Vec model
# Relevant text normalization processes
def convert_to_lowercase(text): return text.lower()

contractions_url = 'https://raw.githubusercontent.com/sugatagh/E-commerce-Text-Classification/main/JSON/english_contract
contractions_dict = pd.read_json(contractions_url, typ = 'series')
contractions_list = list(contractions_dict.keys())

def convert_contractions(text):
    words = []
    for word in regexp.tokenize(text):
        if word in contractions_list:
            words = words + contractions_dict[word].split()
        else:
            words = words + word.split()
    return " ".join(words)
```

```
# Text normalization for Word2Vec
for df in [data_train, data_val, data_test]:
    df['tokens'] = (df["description"].apply(convert_to_lowercase)
                                     .apply(convert_contractions)
                                     .apply(regexp.tokenize))
data_train[['tokens', 'label']]
```

	tokens	label
15525	[practical, approach, to, acupuncture, 1, abou...	2
1536	[nice, goods, leatherette, office, arm, chair,...	1
21984	[ekan, fashionable, fedora, hat, for, girls, b...	3
25056	[techyshop, dmx512, professional, concert, sho...	0
25213	[marantz, tt5005, fully, automatic, belt, driv...	0
...	...	...
23992	[apple, ipad, pro, mpf12hn, a, tablet, 10, 5, ...	0
27640	[printelligent, laptop, skins, stickers, super...	0
14501	[the, challenger, sale, taking, control, of, t...	2
14555	[international, mathematics, olympiad, work, b...	2
11590	[operating, system, concepts, 8th, edition, wi...	2

22241 rows × 2 columns

```
from gensim.models import KeyedVectors
```

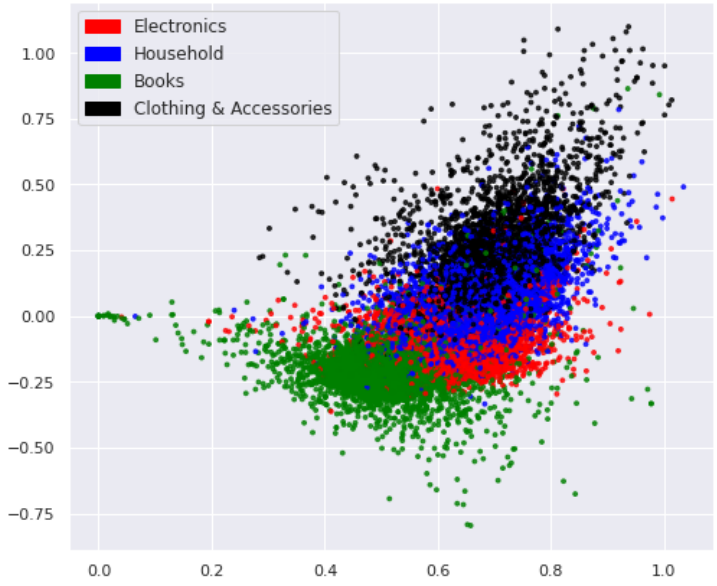
```
#import gensim.downloader as api

#wv = api.load('word2vec-google-news-300')
```

```
word2vec_path='/content/drive/MyDrive/Colab Notebooks/GoogleNews-vectors-negative300.bin.gz'
word2vec = gensim.models.KeyedVectors.load_word2vec_format(word2vec_path, binary = True)
```

```
# Some useful functions for Word2Vec
def get_average_word2vec(tokens_list, vector, generate_missing = False, k = 300):
    if len(tokens_list) < 1:
        return np.zeros(k)
    if generate_missing:
        vectorized = [vector[word] if word in vector else np.random.rand(k) for word in tokens_list]
    else:
        vectorized = [vector[word] if word in vector else np.zeros(k) for word in tokens_list]
    length = len(vectorized)
    summed = np.sum(vectorized, axis = 0)
```

```
    averaged = np.divide(summed, length)
    return averaged

def get_word2vec_embeddings(vectors, tokens, generate_missing = False):
    embeddings = tokens.apply(lambda x: get_average_word2vec(x, vectors, generate_missing = generate_missing))
    return list(embeddings)

def plot_embedding(X, y):
    truncated_SVD = TruncatedSVD(n_components = 2)
    truncated_SVD.fit(X)
    scores = truncated_SVD.transform(X)
    color_mapper = {label:idx for idx, label in enumerate(set(y))}
    color_column = [color_mapper[label] for label in y]
    colors = ['red', 'blue', 'green', 'black']

    plt.scatter(scores[:, 0], scores[:, 1], s = 8, alpha = 0.8, c = y, cmap = matplotlib.colors.ListedColormap(colors))
    red_patch = mpatches.Patch(color = 'red', label = 'Electronics')
    blue_patch = mpatches.Patch(color = 'blue', label = 'Household')
    green_patch = mpatches.Patch(color = 'green', label = 'Books')
    black_patch = mpatches.Patch(color = 'black', label = 'Clothing & Accessories')
    plt.legend(handles = [red_patch, blue_patch, green_patch, black_patch], prop = {"size": 12})
```

```
# Word2Vec embedding
X_train_embed = get_word2vec_embeddings(word2vec, data_train['tokens'])
X_val_embed = get_word2vec_embeddings(word2vec, data_val['tokens'])
X_test_embed = get_word2vec_embeddings(word2vec, data_test['tokens'])

fig = plt.figure(figsize = (8, 7))
plot_embedding(X_train_embed, y_train)
plt.show()
```




```
# Converting to Compressed Sparse Row matrix
X_train_w2v = scipy.sparse.csr_matrix(X_train_embed)
X_val_w2v = scipy.sparse.csr_matrix(X_val_embed)
X_test_w2v = scipy.sparse.csr_matrix(X_test_embed)
```

```
# Summary of baseline models
score(X_train_w2v, y_train, X_val_w2v, y_val, names = names, models = models)
```

	Classifier	Training accuracy	Validation accuracy
2	Linear SVM	0.937233	0.934173
3	Random Forest	0.999910	0.928417

```
# Hyperparameter tuning for linear SVM
svm_classifier = svm.SVC()
params_svm = {
    'kernel': ['linear'],
    'C': [0.1, 1, 10, 100]
}

best_model_svm, best_params_svm, best_score_svm, count = svm_classifier, ParameterGrid(params_svm)[0], 0, 0
for g in ParameterGrid(params_svm):
    time_start = time.time()
    count += 1
    print(f"Gridpoint #{count}: {g}")
    svm_classifier.set_params(**g)
    svm_classifier.fit(X_train_w2v, y_train)
    y_train_pred, y_val_pred = svm_classifier.predict(X_train_w2v), svm_classifier.predict(X_val_w2v)
    score_train, score_val = accuracy_score(y_train, y_train_pred), accuracy_score(y_val, y_val_pred)
    time_stop = time.time()
    m, s = int(time_stop - time_start) // 60, int(time_stop - time_start) % 60
    print(f"Training accuracy: {score_train}, Validation accuracy: {score_val}, Runtime: {m}m{s}s")
    print(" ")
    if score_val > best_score_svm:
        best_params_svm, best_score_svm = g, score_val

best_model_w2v, best_params_w2v, best_score_w2v = svm.SVC(), best_params_svm, best_score_svm
best_model_w2v.set_params(**best_params_w2v)
print(f"Best model: {best_model_w2v}")
print(" ")
print(f"Best parameters: {best_params_w2v}")
print(f"Best validation accuracy: {best_score_w2v}")
```

```
Gridpoint #1: {'C': 0.1, 'kernel': 'linear'}
Training accuracy: 0.922665347781125, Validation accuracy: 0.9190647482014388, Runtime: 3m31s

Gridpoint #2: {'C': 1, 'kernel': 'linear'}
Training accuracy: 0.93723303808282, Validation accuracy: 0.9341726618705036, Runtime: 1m59s

Gridpoint #3: {'C': 10, 'kernel': 'linear'}
Training accuracy: 0.9468998696101794, Validation accuracy: 0.9366906474820144, Runtime: 1m37s

Gridpoint #4: {'C': 100, 'kernel': 'linear'}
Training accuracy: 0.9502270581358752, Validation accuracy: 0.9298561151079137, Runtime: 1m57s

Best model: SVC(C=10, kernel='linear')

Best parameters: {'C': 10, 'kernel': 'linear'}
Best validation accuracy: 0.9366906474820144
```

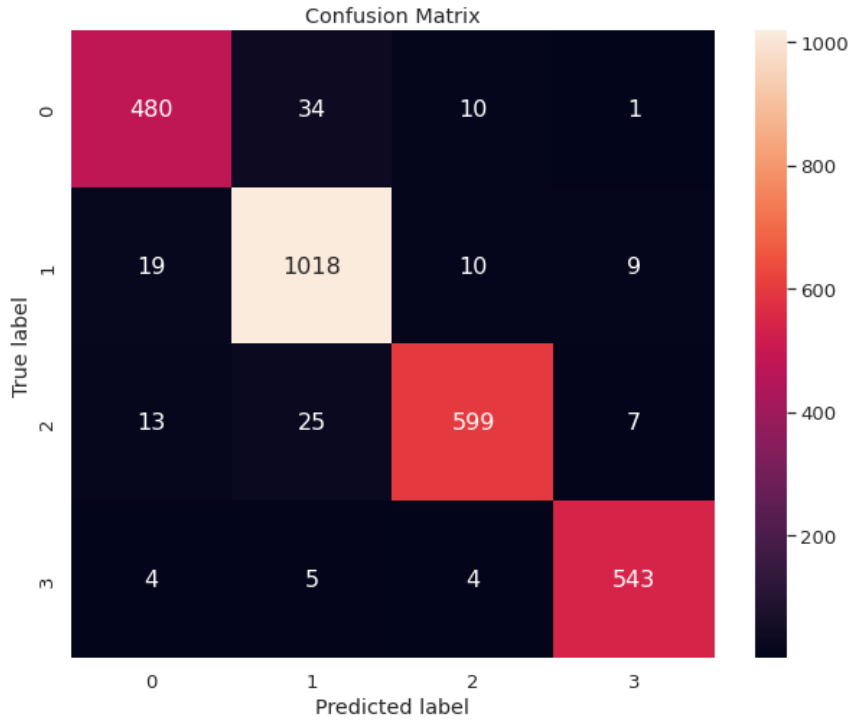
```
# Function to compute and print confusion matrix
def conf_mat(y_test, y_test_pred, figsize = (10, 8), font_scale = 1.2, annot_kws_size = 16):
    class_names = [0, 1, 2, 3] # ['Electronics', 'Household', 'Books', 'Clothing & Accessories']
    tick_marks_y = [0.5, 1.5, 2.5, 3.5]
    tick_marks_x = [0.5, 1.5, 2.5, 3.5]
    confusion_matrix = metrics.confusion_matrix(y_test, y_test_pred)
    confusion_matrix_df = pd.DataFrame(confusion_matrix, range(4), range(4))
    plt.figure(figsize = figsize)
    sns.set(font_scale = font_scale) # label size
    plt.title("Confusion Matrix")
    sns.heatmap(confusion_matrix_df, annot = True, annot_kws = {"size": annot_kws_size}, fmt = 'd') # font size
    plt.yticks(tick_marks_y, class_names, rotation = 'vertical')
    plt.xticks(tick_marks_x, class_names, rotation = 'horizontal')
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.grid(False)
    plt.show()
```

```
# Best model
```

```
if best_score_tfidf >= best_score_w2v:
    best_model, X_train_vec, X_test_vec = best_model_tfidf, X_train_tfidf, X_test_tfidf
else:
    best_model, X_train_vec, X_test_vec = best_model_w2v, X_train_w2v, X_test_w2v
```

```
# Prediction and evaluation on test set
best_model.fit(X_train_vec, y_train)
y_test_pred = best_model.predict(X_test_vec)
score_test = accuracy_score(y_test, y_test_pred)
print(pd.Series({"Test accuracy": score_test}).to_string())
print(" ")
conf_mat(y_test, y_test_pred, figsize = (10, 8), font_scale = 1.2, annot_kws_size = 16) # Confusion matrix
```

```
Test accuracy    0.949299
```




```
from sklearn.metrics import confusion_matrix
y_test_pred = best_model.predict(X_test_vec)
cm=confusion_matrix(y_test, y_test_pred, labels=[0,1,2,3])
print(cm)
print()

from sklearn.metrics import classification_report
target_names = ['0','1','2','3']
print(classification_report(y_test, y_test_pred, target_names=target_names))
```

```
[[ 480   34   10    1]
 [  19 1018   10    9]
 [  13   25  599    7]
 [   4    5    4  543]]

              precision    recall  f1-score   support

           0       0.93      0.91      0.92       525
           1       0.94      0.96      0.95      1056
           2       0.96      0.93      0.95       644
           3       0.97      0.98      0.97       556

    accuracy                           0.95      2781
   macro avg       0.95      0.95      0.95      2781
weighted avg       0.95      0.95      0.95      2781
```

```
#error analysis
#clothing as electronics
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==3 and y_test_pred[i]==0):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
keetron pu leather tpu wrist strap credit card holder flip folio shockproof cover closure slim protector note oran
lg phone caselg grace lte case screen protectoranoke cute protective cell phone cover girl woman lg ch phone model
expedition man watch manufacturing marketing brand watch come watch call timex expedition unisex watch timepiece d
case galaxy tab case dteck cartoon flip folio case leather stand wallet cover samsung tab tabletbutterfly bb butte
errors: 4
```

```
#error analysis
#clothing as household
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==3 and y_test_pred[i]==1):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
india shawl kutch embroider border mirror dimension ft ft
chronex ponytail hair band holder rubber band girlswomen hair band tool hold hair tightly metal hurt hair thicknes
piece mermaid glitter cupcake cake pick decoration baby shower birthday party favor mermaid seahorse food pick pac
hornbull man leather wallet belt hornbull help people world easily simply quality wallet card case passport holder
packit traveler lunch bag polka dot color dot packit traveler lunch bag polka dot
errors: 5
```

```
#clothing as books
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==3 and y_test_pred[i]==2):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
deal man suspender neck bow suspender lowestpricedeal toped selling know everybody caresso come lot collection sus
eccellente fedora hat design sale diwali offer
spiderman glove disc fan crawl marvel superhero spiderman glove disc launcher surely become see child save disc th
man adriano running shoe bear part name founder brand name lotto sport world research design innovation accompany
errors: 4
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==0 and y_test_pred[i]==1):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
morpho icon fingerprint scanner service version product description detail feature maintenance design capture high
readat sheet mm thickness elevator write desk height smoke colour readat provide quality product readat give quali
lapcare multi function stand station ergonomically station height auto lock joint station people habit sift bed fl
xpro ball head quickrelease plate
cable world mini screwdriver bit set extension rod cable world mini screwdriver bit set extension rod home applian
toyz scale die cast roll royce toy car pull back blink lightsassorte sizeroll royce scale die cast roll royce phan
toyhouse audi battery operate rideon car kid paint blue battery rideon car toyhouse operate use surface let build
trendmax night lead selfie light lead bulb provide enough take provide lead light light scene light device use cli
morpheus mm ceiling fan fan visually body look room motor supreme performance consume power aerodynamically blade
moriah mount premium tea tea start become malty get taste palate tea prepare milk blend start day also get schedul
beetel telephone set well soho office home office well soho office home office ringer volume control tone pulse mu
elcor aspect auto lock projection screen inch
philips photo frame clock calendar keep track time style scrapbook add touch slideshow slideshow share memorie sel
saco comfort travel bag mobilepower bankcablesusb drivesmemory cardssim cardssmall accessory grey
bracket handle stand shoe screw zhiyun crane moza air cross accessory screw zhiyun crane moza air cross feiyu dslr
character lcd display module lcd blacklight condition brandnew unopened item packaging
pr fog lamp projector lens beam beam strobe function angel color blue devil eye color motorcycle bike scooter lead
royaloak tv stand wengy design entertainment unit oak need lend sophistication living room rack back require appli
silver compound silver highdensity silver compound enhance compound performance stability silver use shape size pu
advance inch hire photo frame motion sensor lead brand photo year experience dedicated customer service team commi
zozo premium auto removal multiuse stainless steel fastener combination set car dash audio radio door panel repair
orient mm premium ceiling fan color ceiling fan promise ever air category heavenly silence orient aero series rang
color cc colour concealer stick highlighter moisturize concealer stick color correction menow brand type concealer
```

```
art street wood engrave personalized world husband photo picture frame birthday gift photo size scentworld husband
quantem keyboard keypad malfunctioning work properly replace quantum keyboard serve time feature quantum keyboard
dishankart cash drawer point sale pos system dishankart cash drawer combination robustness performance application
inch ipadtablet sleeve cover cute satin stitch background pink check use contrast pink background leave impression
amazon brand solimo swirl mm ceiling fan color solimo introduce range ceiling fan help beat heat weather blade cei
dot detox color correction mask even woman paraben reason become coarser time reason detoxify mask specifically te
stand silver inch
piston fit earphone space gray
inateck port pcie expansion card interface usb express card desktop window mini pcie hub controller adapter power
bluewud kunsua tv entertainment set box stand shelf wenge sizelarge selfassembly product require installation come
mosquick teakwood multipurpose sofa mat teak wood sofa tray laptop desk place mat also use arm product easily shap
errors: 34
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==0 and y_test_pred[i]==2):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
basher dinosaur bare bone author simon basher creator illustrator ten concept book child include physics biology m
ecodeal vermicompost fertilizer kg vermicompost make earthworm decompose cow dung waste
shutter huggers monkey hugger style shutter hugger monkey
emergency chronicle indira gandhi democracy turning point author prakash daytonstockton professor history princeto
gift year prime
karaoke microphone ahuja research development centre recognise government india take pride develop product know re
antenna theory analysis design wiley interscience author constantine balanis receive bsee virginia tech mee form u
ccnp routing switching switch cert guide dvd author david hucaby ccie lead network engineer university work health
armor radiation shield pad latte beige medium
pantone colormunki design colormunki solution give freedom design color spectrum even never think swing inspiratio
errors: 10
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==0 and y_test_pred[i]==3):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
cross coverage product description coverage style feature provide cup seperation support branddescription start co
errors: 1
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==1 and y_test_pred[i]==0):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
gbs wifi attendance system cloud storage lifetime maintenance ios support report globally
ceuta combo offer stand bracket phone stand metal lazy stand use quality ab design beddesktop mobile mount holder
sellnship travel adaptor allinone world conversion plug aus europe sp device style travel abroad study business tr
magideal alloy accessory toolbox backpack silver description aluminum toolbox hold kind machine accessory light we
adisa light weight laptop backpack
dmg rotate flip cover book case apple inch colourblack rotate cover apple inch premium construct pu leather degree
belline voltage tv inch dth music system home theatre year warranty brand bellinemodel belcvt work range voltage t
pc scratch proof feel pad furniture pad padfloor protector furniture padsfurniture sofa leg pad lukzer scratch pro
mx surge protector spike guard universal socket switch circuit breaker power cable ampere mx surge protector socke
kva pure sine cruze wave inverter home establishment power need also power cut hamper performance appliance kva ba
besmelody way splitter adapter tshape jack plug surveillance equipment besmelody besmelody way splitter adapter ts
samsung cm inch hd tv model cm hd tv series display size inch resolution picture engine hyperreal dts codec dts pr
import case cover bag sony playstation ps vita psv quality product
sanyo cm inch hd ips lead tv dark grey size inch indulge entertainment neverbefore television enjoy stunning pictu
chris kate polyester ltr school backpack colourblue chris kate polyester school backpack school collegegoer make p
lg cm inch hd tv silver model size inch secret tv color view panel quality determine quality coffee quality panel
dell precision tower workstationdesktop core generation gb gb tb hdd do monitor dvd drive addin card thunderbolt p
ebcoteliscope channel ball bearing channel mm inch zinc teliscope channel ball bearing channel mm inch zinc
amicivision waterproof rgb lead rf tv background amicivision light strip use home decoration hotel club shopping p
errors: 19
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==1 and y_test_pred[i]==2):
```

```
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
latch import jimi lead group call band gypsy quartet parisbase gypsy guitarist thing steeve lafonte leave channel
pig want eat think author baggini editor cofounder magazine write regularly sunday prospect tes appear time author
elle jan issue elle fashion woman inspire develop style inform intelligence optimism entertaining frivolity
rapidex home tailoring course
abacus pack book abacus way number kid way teach number kid abacus tool teach child math counting addition subtrac
york face studio master blur primer redness control colourredness control blur away concern way master face studio
border bleed insider account relation author engineer training profession writer choice rajiv dogra member service
border time magazine author editor time magazine
outdoor camping replacement canopy porch tent pole tent cover awne fuchsia sections tent polethey contact use repl
conde nast traveller india
errors: 10
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==1 and y_test_pred[i]==3):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
snezhinka silk cover charmeuse weave momme size colour pillow cover make silk fabric charmeuse weave momme use sil
york pearl colourpearl formula oil sweat give look day time bid goodbye touchup stay hour
barbie frock baby girl empire maxi dress pastel blue flower gown
charm anklet woman show fashion sense foot uniquely anklet charm crop trouser tealength skirt way flaunt
dimension baby hugs fairy bibs stamp cross stitch set baby hug stamp cross stitch cute cuddly baby design fabric b
woman girl kutchi art work sle sling bag polyester sle bag wear give look carry handcraft embroider sling bag hous
beauty craft rajasthani hand embroidery slipper woman trendy pair make material design make rajasthani day long co
baby face towel pack pack quality baby hanky baby touch
cotton oven glove pack cotton cotton waste glove
errors: 9
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==2 and y_test_pred[i]==0):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
lead hd projector hd display change living room movie theatre get design lead lcd home projector display color hig
toreto pocket max water resistant speaker tor toreto provide premium quality brand world focus offer charger cable
mattel card game style card game meet charade card game ever draw require race act clue team use nothing image pic
instax mini film sheet style pack sheet offer stunning picture wallet mini photo frame film ensure reproduction vi
point talk
silver cm interesting canoe fishing sport decor silhouette vinyl car sticker item type sticker package package ite
audiology connect car headrest mount car tablet mount car holder tablet mount holder smartphones tablets audiology
lenovo aio fhd borderless allinone desktop gen core homeintegrated graphicsblack lenovo ideacentre aio feature wid
rolltop computerintel core duo ghz motherboard lead monitor ram gb size product rolltop computerintel core duo ghz
blufury ledlcd screen computer dust cover combo duty blufury computer dust cover combo product description protect
music amplifier ax line tablet speaker amplifier display product support devicetablet ipad tv lead tv desktop lapt
bloomerang lora range rf wireless power module arduino
audit progress cd import reissue pop album pissedoff protopunk rapidfire riff member rocket crypt drive obit inclu
errors: 13
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==2 and y_test_pred[i]==1):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
book coconut product cultivation processing coconut crop grow india presently cover country find tropic area cocon
amazing sign toilet sign board number quality printing material glowcy lamination paste mm foam sheet back side si
periodic table
butterfly design fly fishing lure hook description vivid butterfly design work well size approx inch cm hook lengt
pc bag seed lettuce family health seed pc bag seed lettuce family health seed vegetable
tape feature design steam sterilizationwater base indicator color change
mind market money journey intraday trading use market profile order flow
schwarzkopf taft weather look marathon power gel long hold schwarzkopf taft marathon look long power gel ml schwar
sweet home mini resin crafts fairy miniature pc set sweet home mini resin crafts fairy miniature pc set
gift card design product description gift freedom choicegift love business associate lifestyle gift card let choos
```

```
disruption energy transportation valley make oil gas coal utility car
wing fire wing fire
style rubber stamp shape scrapbooke craft card make ink stamp craft stamp shape offer cute craft stamp stamp offer
store hand carve fold book stand holder carving store indya travel forget hand craft skill india store something r
sprinkler head tee mist water mg agriculture lawn garden patio greenhouse swimming mist fog coolingirrigation wate
etailor page handmade vintage genuine leather paper diary box brown
vip self defence rod metal travel bag stroll walking dog camping steel steak care kind use safety purpose travel b
aart store sofa cum bed seater sleep comfortably color colorblack aart store surprised piece furniture sofa cum be
mintkraft foam sheet multicolor self beware area sign board inch inch product safety sign board house mintkraft ma
home buy bag premium quality waterproof adult bag camping hiking adventure trip multi color namemulti quality slee
strike back olympics massacre israel response author aaron klein time magazine intelligence affair correspondent j
dog toy dog chew knot rainbow rubber cord weave toy jingle bell cat training playing iq toy gift pet rainbow rubbe
shark park reader
craft hand wooden doll hand paint matryoshka stack doll set piece lady colorre matryoshka doll also know nesting d
foot ankle radiology
errors: 25
```

```
err_cnt = 0
for i in range(0, len(y_test)):
    if(y_test[i]==2 and y_test_pred[i]==3):
        print(X_test_norm[i])
        err_cnt = err_cnt+1
print("errors:", err_cnt)
```

```
anne frank girl take day
bell sock man train crew
chimes doctor man eye collection woman chime bring fashion jewellery design renowned jewellery manufacturer world
chhattisgarh pet product make color feature pu laceup closure toe design use occasion well office footwear further
woman cap
tinkle pack pack include ever tinkle series
compression activity reduce postexercise muscle soreness improve performance microfiber spandexwarpknit constructi
errors: 7
```