Data_kitchen

December 6, 2021

[1]:
```
import gzip
import pandas as pd
import numpy as np
import nltk
import requests
from io import BytesIO
import imblearn
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('wordnet')
import re
import contractions
from sklearn.preprocessing import LabelEncoder
from bs4 import BeautifulSoup
pd.set_option('display.max_colwidth', None)
import requests
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import gensim.downloader as api
from gensim.models import Word2Vec
wv = api.load('word2vec-google-news-300')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/khalid/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/khalid/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-
packages/gensim/similarities/__init__.py:15: UserWarning: The
gensim.similarities.levenshtein submodule is disabled, because the optional
Levenshtein package <https://pypi.org/project/python-Levenshtein/> is
unavailable. Install Levenhstein (e.g. `pip install python-Levenshtein`) to
suppress this warning.
  warnings.warn(msg)
```

1 Functions

[2]:
```
def read_file(link):
    url = link
    data_compressed = requests.get(url).content
    data = pd.read_csv(BytesIO(data_compressed), compression='gzip',␣
↪sep='\t',error_bad_lines=False, warn_bad_lines=False)
    return data


# Remove HTML tags funtion
def remove_tags(txt):
    # parse html content
    soup = BeautifulSoup(txt, "html.parser")

    # get tags content
    for data in soup(['style', 'script']):
        data.get_text()

    # return html's tag content
    return ' '.join(soup.stripped_strings)

# Remove URLS funtion
def remove_urls(txt):
    return re.sub(r"http\S+", "", txt)

# Apply contraction to words
def contractionfunction(s):
    expanded_words = []
    for word in s.split():
        expanded_words.append(contractions.fix(word))
    result = ' '.join(expanded_words)
    return result

def remove_non_alphabetical(txt):
    regex = re.compile('[\W_0-9]+')
    dirty_list = txt.split()
    clean_list = [regex.sub(' ', word) for word in dirty_list]
    clean_string = ' '.join(clean_list)
    return clean_string



# remove stop words function
def remove_stop_words(txt):

    stop = stopwords.words('english')
```

```
    word_list = txt.split()
    clean_list = []
    clean_string = ''
    for word in word_list:
        if word not in stop:
            clean_list.append(word)
    clean_string = ' '.join(clean_list)
    return clean_string

def leammatize_review(txt):
    lemmatizer = WordNetLemmatizer()
    word_list = txt.split()
    clean_list = []
    clean_string = ''
    for word in word_list:
        new_word = lemmatizer.lemmatize(word)
        clean_list.append(new_word)
    clean_string = ' '.join(clean_list)
    return clean_string

def review_num_words(txt):
    return len(txt.split())
```

2 Data

[3]:
```
data_kitchen = read_file("https://s3.amazonaws.com/amazon-reviews-pds/tsv/
 ↪amazon_reviews_us_Kitchen_v1_00.tsv.gz")
#data_home = read_file("https://s3.amazonaws.com/amazon-reviews-pds/tsv/
 ↪amazon_reviews_us_Home_Improvement_v1_00.tsv.gz")
```

```
/var/folders/qf/k0f0k7s94bd63pjpts_176c40000gn/T/ipykernel_25230/1245615390.py:1
: FutureWarning: The warn_bad_lines argument has been deprecated and will be
removed in a future version.


  data_kitchen = read_file("https://s3.amazonaws.com/amazon-reviews-
pds/tsv/amazon_reviews_us_Kitchen_v1_00.tsv.gz")
/var/folders/qf/k0f0k7s94bd63pjpts_176c40000gn/T/ipykernel_25230/1245615390.py:1
: FutureWarning: The error_bad_lines argument has been deprecated and will be
removed in a future version.


  data_kitchen = read_file("https://s3.amazonaws.com/amazon-reviews-
pds/tsv/amazon_reviews_us_Kitchen_v1_00.tsv.gz")
```

3 Pre Processing

[4]:
```
###### Pre Processing

#removed the na values
data_kitchen = data_kitchen.dropna().reset_index(drop=True)

#removed the 3 dates from the product category
a = data_kitchen['product_category'] == 'Kitchen'
data_kitchen = data_kitchen[a]

#make decoder
data_kitchen['vine'] = data_kitchen['vine'].astype('category').cat.codes
data_kitchen['verified_purchase'] = data_kitchen['verified_purchase'].
 ↪astype('category').cat.codes

#removing unnecessary columns
data_kitchen = data_kitchen.drop( columns=['customer_id', 'product_id',␣
 ↪'marketplace', 'product_category', 'review_id', 'product_parent',␣
 ↪'product_title'] )

#combining two features in one feature
data_kitchen['reviews'] = data_kitchen['review_headline'] + ' ' +␣
 ↪data_kitchen['review_body']
data_kitchen = data_kitchen.drop( columns=['review_headline', 'review_body'] ) ␣
 ↪# drop the 2 columns
data_kitchen['reviews'] = data_kitchen['reviews'].str.lower()  # make lowercase

# make time
data_kitchen["review_date"] = pd.to_datetime(data_kitchen["review_date"])
data_kitchen["review_date"] = data_kitchen["review_date"].dt.isocalendar().week

print(data_kitchen.shape)
data_kitchen.head(10)
```

```
(4874562, 7)
```

[4]:
```
   star_rating  helpful_votes  total_votes  vine  verified_purchase  \
0          5.0            0.0          0.0     0                  1
1          5.0            0.0          1.0     0                  1
2          5.0            0.0          0.0     0                  1
3          5.0            0.0          1.0     0                  1
4          5.0            0.0          0.0     0                  1
5          1.0            1.0          1.0     0                  1
6          5.0            0.0          0.0     0                  1
7          5.0            0.0          0.0     0                  1
8          5.0            0.0          0.0     0                  1
```

```
9          3.0            0.0           0.0     0                  1

   review_date  \
0           36
1           36
2           36
3           36
4           36
5           36
6           36
7           36
8           36
9           36

                                                                      reviews
0
beautiful. looks great on counter beautiful.  looks great on counter.
1  awesome & self-ness i personally have 5 days sets and have also bought 2 sets
for other people in my home. the two other sets i have decided to keep for
myself. the purpose of keeping them for myself is to use them for other other
than salt and pepper. they stay perfect, i use them constantly! i have a couple
of people here that use them, say that there's just awesome. i did have a salt
shaker that had a little problem converting from sea salt to himalaya salt.  did
not fall out correctly so what i did was i just took a top of part which is
really simple it's like five pieces total and just kind of cleaned around the
teflon and you know that the salt buildup from there was caused by humidity,
cleaned out, my gosh it's unbelievable how much better, its almost better than
new thank you bavaria these are top of the line in my book i hope in the near
future you have some more come out.  i sure would like to buy some more and some
for my kids and family for christmas otherwise i'm keeping what i have. i think
i deserve it. thank you… don't know if this makes much sense doesn't need to,
self-ness !!!
2
fabulous and worth every penny fabulous and worth every penny. used for cleaning
corn from the cob in seconds :) would recommend its purchase
3
five stars a must if you love garlic on tomato marinara sauce.
4
better than sex worth every penny! buy one now and be a pizza slice master!!
5
does not work on induction stoves! (not suitable for all type of surfaces) the
description says &#34;suitable for all type of surfaces&#34; but it is not
true!<br /><br />we ordered this item for our large family (we like to cook a
lot) but this does not work as advertised!<br /><br />this pressure cooker does
not work for induction stoves.  i called the manufacturer (magefesa) and they
confirmed that it will not work. we are exchanging this pot for the stainless
steel version that is supposed to work on induction stoves.  magefesa said that
```

```
the 14.3 quart pot will work on induction stoves and the description here on
amazon clearly states that it will work on induction stoves.<br /><br
/>thankfully amazon has a great return process so they will send a truck to pick
this up and they will send the other version in a couple days.
6
awesome! first fryer i have owned and what a … awesome! first fryer i have
owned and what a luxury item! love the magnetic cord attachment. the baskets are
a great size for home use and gets up to temp no problem. couldn't be happier
with my purchase.
7
five stars very good item. quick delivery.
8
five stars sharp and look great
9
three stars should have come with a kit to install drain tube system.
```

[27]:
```python
data_kitchen['star_rating'].value_counts()
```

[27]:
```
1    3856246
0    1018316
Name: star_rating, dtype: int64
```

[28]:
```python
data_kitchen['star_rating'] = np.where(data_kitchen['star_rating'] > 3.0 , 1, 0)
```

[29]:
```python
data_kitchen['star_rating'].value_counts()
```

[29]:
```
0    4874562
Name: star_rating, dtype: int64
```

[25]:
```python
data_k_sampled = data_kitchen.sample(n=700000, replace=False, ignore_index =␣
↪True)
```

[26]:
```python
data_k_sampled['star_rating'].value_counts()
```

[26]:
```
1    553431
0    146569
Name: star_rating, dtype: int64
```

[30]:
```python
data_k_sampled
```

[30]:
```
        star_rating  helpful_votes  total_votes  vine  verified_purchase  \
0                 0            5.0          5.0     0                  1
1                 0            1.0          1.0     0                  1
2                 0            1.0          1.0     0                  1
3                 1            1.0          2.0     0                  0
4                 0           19.0         26.0     0                  0
…               …              …           …    …                …
699995            1            0.0          0.0     0                  1
```

```
699996            0            0.0           0.0     0                   1
699997            1            0.0           0.0     0                   1
699998            1            0.0           0.0     0                   1
699999            1            0.0           0.0     0                   1

        review_date  \
0                21
1                52
2                26
3                 8
4                27
…              …
699995           24
699996           33
699997           25
699998           31
699999           40

                                                                  reviews
0
one star cheap, could only use these for craft supplies. even that, they break
and split.
1
not worth your time. i can do better and faster. not even work the time to
return. trash bin it went.
2
one star this microwave died after 3 months of normal use.  the manufacturer did
nothing to help.
3       what you should know before you buy! i was provided a freddie and sebbie
freezable lunch bag/cooler bag free for testing and review and was asked only to
give my honest review, so this is what i found.<br /><br />what you should know
before you buy:<br />  this is a medium sized lunch bag/cooler bag.  large
enough to hold quite a few items.<br />  this lunch bag has the reusable ice
packs built right into the sides of the bag, so no fumbling around with ice
packs that take up space in your typical bag.<br />  very nice quality and seems
to be quite sturdy.<br />  did a great job of keeping my lunch cold at work.<br
/>  folds up pretty small with a velcro closure so it is easy to store in your
freezer while you are waiting for it to cool off.<br />  has extra pockets on
the sides.<br /><br />conclusion:<br />overall, this is great cooler bag/lunch
bag.  this bag has the reusable ice packs that you normally put in the bag built
right into the walls of the bag.  this is great because you don't have to take
up bag space with separate ice packs or have to fumble through them to get to
your food.  it has a really nice, compact design when folded, so it doesn't take
up much space while it is sitting in your freezer cooling off.  when opened up,
it holds quite a bit of food and drink items.  i take my lunch to work pretty
much every day, so i gave this bag a try and it did a great job keeping my lunch
perfectly cold.  we always take a small cooler or cooler bag with us on day
```

```
trips that have lunch items and/or drinks for our family of four, and this bag
can easily accommodate all of that.  if you are in the market for a lunch bag
and the integrated ice packs sound like a feature you would like, this bag is a
great choice.
4
difficult to suck i was very excited to order these popsicle molds after reading
all the positive reviews.  they are very easy to make and look lovely.  eating
them is another story!  a popsicle should fit comfortably in any size mouth and
i'm a woman with a very average size mouth.  the popsicle was too big and has
too many ridges to allow you to suck it…you never have enough surface area in
your mouth at one time to melt any of the juice!  it's not what you expect from
a popsicle at all! i now know that a popsicle needs smooth sides, not
ridges…it doesn't work for me at all and it drips!
…
…
699995
best 14-inch pan i've found perfect for large portion cooking. very sturdy and
the non stick surface works as well as any pan i have used.<br /><br />hand wash
for maximum life of the non stick surface -- there's something in most dish
washer detergents that is very hard on any non stick surface. the surface on
this pan is so slick that a couple of swipes with a soapy dish cloth will take
anything right off.<br /><br />highly recommended.
699996
the pitcher remains cloudy only after 3 uses. the … the pitcher remains cloudy
only after 3 uses. the spout opening is flimsy and will not stay locked on. i
would have given it a 2 star, but the fruit infused water does meet my
expectation.
699997
awesome i love this magnet.  it is high quality and looks awesome.  i absolutely
cannot complain.  not to mention the shipping was super fast! thanks!
699998
the perfect machine for the job i love this product.  i bought this one as a
gift but i've had mine for decades.  it works perfectly and making readying
apples for cooking super easy and super fast.  i have searched for other apple
peeler, corer and slicer devices and haven't found any.  that's because there is
no better device.  the design simply can't be improved upon because it works
perfectly just as it is.
699999
five stars easy to use.makes stir fry so easy and delicious highly recommend .

[700000 rows x 7 columns]
```

[31]:
```python
# Applying preproccessing
data_k_sampled['reviews'] = data_k_sampled['reviews'].apply(lambda body :
 ↪remove_tags(body))
data_k_sampled['reviews'] = data_k_sampled['reviews'].apply(lambda body :
 ↪remove_urls(body))
```

```
data_k_sampled['reviews'] = data_k_sampled['reviews'].apply(lambda body :
 ↪contractionfunction(body))
data_k_sampled['reviews'] = data_k_sampled['reviews'].apply(lambda body :
 ↪remove_non_alphabetical(body))
data_k_sampled['reviews'] = data_k_sampled['reviews'].apply(lambda review:
 ↪remove_stop_words(review))
data_k_sampled['reviews'] = data_k_sampled['reviews'].apply(lambda txt :
 ↪leammatize_review(txt))
```

[58]:
```
data_k = data_k_sampled.copy()
```

[59]:
```
data_k['star_rating'].value_counts()
```

[59]:
```
1    553431
0    146569
Name: star_rating, dtype: int64
```

[60]:
```
from sklearn.utils import resample
df_majority = data_k[data_k["star_rating"]==1]
df_minority = data_k[data_k["star_rating"]==0]
df_majority_downsampled = resample(df_majority,
                                 replace=False,    # sample without replacement
                                 n_samples=len(df_minority),     # to match␣
 ↪minority class
                                 random_state=123) # reproducible results
# Combine minority class with downsampled majority class
data_k_concat = pd.concat([df_majority_downsampled, df_minority])
# Display new class counts
data_k_concat['star_rating'].value_counts()
```

[60]:
```
1    146569
0    146569
Name: star_rating, dtype: int64
```

[62]:
```
data_k_concat = data_k_concat.reset_index(drop=True)
```

[63]:
```
data_k_concat
```

[63]:
```
        star_rating  helpful_votes  total_votes  vine  verified_purchase  \
0                 1           51.0         51.0     0                  1
1                 1            0.0          0.0     0                  1
2                 1            0.0          0.0     0                  1
3                 1            0.0          0.0     0                  1
4                 1            0.0          0.0     0                  1
…               …            …          …   …            …
293133            0            0.0          0.0     0                  1
293134            0           15.0         16.0     0                  0
```

```
293135            0            3.0           4.0     0                  1
293136            0            9.0           9.0     0                  1
293137            0            0.0           0.0     0                  1

        review_date  \
0                 9
1                41
2                 8
3                34
4                28
…              …
293133           28
293134           28
293135           36
293136           11
293137           33

     reviews
0
work great novice candy maker bit scared use one thermometer worked great easy
read made homemade marshallows exactly needed
1
never expected searching top flight pepper mill reading review ordered oxo good
grip pepper mill product awesome volume flake size forehead sweating food hot
enough fresh ground pepper help quest product fantastic say fantastic
2
wonderful work wonderful easy use came great recipe aebleskivers stick following
instruction fall right making pefect pan ton fun using
3
great quality great color went piece space case lpve smooth heavy space case
great quality great color I glad got steal price
4
useful replaced one used year expect one useful camping road trip etc
…
…
293133
three star ok
293134  buy cheap knockoff scam anyone ever tell seems good true probably well
knew better bought anyway amazingly low price anyone know decent banana sliced
least seeing lost head clicked buy button soon got knew cheap knockoff probably
made china con warranty slicer gave one banana horrible customer service called
demanded money back said would take least business day wtf bank sorry sign lend
money amazon banana scam terrible color choice good luck finding banana piece
mixed slicer accidentally ate piece slicer nearly died instruction idea start
way turn banana peel first leave peel sorry gourmet chef instruction booklet
would useful pro none
293135
```

```
work well dried sea salt seems like really good grinder sure easy hold use find
grind well dried fleur de sel bought amazon maybe salt salt specifically
designated grinder friendly sure problem hard recommend
293136
poor quality manufacture printing disappointed quality coaster expected much
better coaster clearly white half slight yellow ish coloration lot yellow
discoloration edge coaster excessive amount glazing element dripping edge dried
porcelain yellow stain even colored packaging bit yellow addition coaster seem
poor quality print spottiness I returning soon get chance really disappointing
much wanted
293137
pitcher remains cloudy us pitcher remains cloudy us spout opening flimsy stay
locked would given star fruit infused water meet expectation

[293138 rows x 7 columns]
```

[64]:
```python
y = data_k_concat['star_rating']
X = data_k_concat.drop(columns =['star_rating'])
```

[65]:
```python
indexs = X['reviews'].index
X['google_word2vec'] = pd.Series(dtype=object)
for idx , review in zip(indexs, X['reviews']):
    unseen_words = 0
    n = len(review.split())
    x = 0
    for word in review.split():
        try:
            x = x + wv[word]
        except KeyError:
            unseen_words = unseen_words + 1
    if unseen_words == n:
        X.at[idx, 'google_word2vec'] = np.NaN
        continue
    x = x/(n-unseen_words)
    x1 = x.reshape(-1, 1)
    x1 = x1.T
    X.at[idx, 'google_word2vec'] = x1[0]
```

[66]:
```python
X_data = X.copy()
X_data = X_data.drop( columns=['reviews'] )
```

[67]:
```python
X_data.isna().sum()
```

[67]:
```
helpful_votes        0
total_votes          0
vine                 0
verified_purchase    0
```

```
review_date          0
google_word2vec     12
dtype: int64
```

[68]:
```python
index_na = X_data.loc[pd.isna(X_data["google_word2vec"]), :].index
X_data = X_data.drop(index_na)
```

[69]:
```python
X_data.shape
```

[69]:
```
(293126, 6)
```

[70]:
```python
y_data = y.copy()
y_data = y_data.drop(index_na)
y_data.shape
```

[70]:
```
(293126,)
```

[71]:
```python
X_data = X_data.reset_index(drop=True)
y_data = y_data.reset_index(drop=True)
```

[72]:
```python
X_data
```

[72]:
```
        helpful_votes  total_votes  vine  verified_purchase  review_date  \
0                51.0         51.0     0                  1            9
1                 0.0          0.0     0                  1           41
2                 0.0          0.0     0                  1            8
3                 0.0          0.0     0                  1           34
4                 0.0          0.0     0                  1           28
…                  …          …    …                  …            …
293121            0.0          0.0     0                  1           28
293122           15.0         16.0     0                  0           28
293123            3.0          4.0     0                  1           36
293124            9.0          9.0     0                  1           11
293125            0.0          0.0     0                  1           33

                                          google_word2vec
0                              [0.043111164, -0.009412978, 0.017001681,
0.10070801, -0.01950582, 0.029783461, 0.09827338, -0.069442324, 0.00504303,
0.06669447, 0.06746928, -0.109176636, -0.029086642, -0.014607747, -0.08432346,
0.05312093, 0.06363254, 0.07739258, 0.012125651, -0.104522705, 0.020956835,
0.13087294, 0.008290608, 0.053798676, 0.021396212, -0.050035264, -0.06092665,
0.045321755, -0.024315728, -0.0092909075, -0.027358161, 0.018141005,
-0.0039435495, -0.023379855, 0.0421346, 0.00018056233, 0.019263374, 0.01530626,
0.0846795, 0.083852135, 0.1111518, -0.08039686, 0.17212592, -0.024376763,
-0.032324895, 0.027286105, -0.054139033, 0.05094401, -0.02123854, 0.013142903,
-0.054829914, 0.053521052, -0.027808296, -0.071876526, -0.030619303,
0.044833712, 0.065159164, -0.118006386, 0.020070395, -0.025552537, -0.053578693,
0.06951226, -0.10331556, -0.051201716, 0.06762017, -0.031789143, -0.07822333,
```

0.0380622, -0.02637397, 0.045051575, 0.081448026, 0.059139676, 0.1043413,
-0.05417209, -0.16880968, -0.010518392, 0.074028865, 0.07558356, 0.025814481,
0.06929694, 0.014274597, -0.039175246, 0.07344733, -0.006245931, -0.038087633,
-0.08363512, -0.0368042, 0.1151869, 0.04592853, 0.03438992, -0.005420261,
-0.011522081, -0.046963163, -0.036787245, -0.016893174, -0.10346137, 0.05199602,
0.030963473, 0.0020898182, -0.0257704, …]
1 [-0.0069059483, 0.063821234, 0.041082945, 0.040674098, -0.014671775,
0.008881513, 0.053462982, -0.11723417, 0.057921465, 0.121474326, 0.013371187,
-0.10051413, -0.059484147, 0.018082563, -0.10670382, 0.05928309, -0.007194519,
0.07959523, -0.027881399, -0.09387131, -0.04881915, 0.029970955, 0.06072998,
-0.020480886, 0.006492839, -0.0646856, -0.050005745, 0.11693618, 0.050608914,
0.0098392265, -0.029147878, -0.017540427, 0.012160357, -0.03584918, -0.08498607,
-0.005053352, 0.0067982394, -0.035407573, 0.048472684, 0.017780976, 0.053495154,
-0.14274372, 0.13602582, -0.018969368, -0.07463163, -0.15564683, 0.0011201747,
-0.021469565, 0.020450369, 0.040609024, -0.008593391, 0.018644445, -0.06489608,
-0.006889792, 0.017166138, 0.005570356, -0.008422852, -0.074957564, 0.07561134,
-0.07616649, -0.053016774, 0.055786133, -0.119641475, -0.05316925, 0.03584918,
0.04771513, -0.045914594, -0.041008864, 0.021627314, 0.029835533, 0.051286697,
-0.034017228, 0.0496889, -0.016616821, -0.19045033, -0.051412247, 0.027770996,
0.024994794, 0.023182588, 0.0019567152, 0.042848475, -0.025708366,
-0.0010941449, 0.035553876, 0.0033569336, -0.07500682, -0.11225891, 0.15803258,
0.034312528, 0.030070586, 0.015170827, 0.02107149, -0.035930857, -0.018253103,
-0.058747236, -0.0650527, 5.6939966e-06, 0.0019102658, 0.021086749,
-0.0039978027, …]
2 [0.030339291, 0.0396849, 0.050813373, 0.12044164,
0.0077594956, -0.016965365, 0.09022281, -0.08133738, -0.026020251, 0.059557464,
-0.011858087, -0.0810322, -0.027909128, 0.0009058902, -0.07804148, 0.059679534,
0.08434095, 0.067916065, -0.033545244, -0.12472373, 0.026411232, 0.1403941,
0.01977539, 0.041388262, 0.037064403, -0.030378694, -0.03434673, 0.089599386,
-0.0042853104, -0.007537842, -0.04081164, -0.013536955, 0.061827008,
0.027093185, 0.016723633, -0.022829557, 0.013331363, -0.028416684, 0.07611601,
0.06340669, 0.13658383, -0.068166636, 0.17797852, -0.015535857, -0.018935354,
-0.065342955, -0.03414114, 0.018040707, 0.0583464, 0.031204676, -0.037809674,
0.04871248, -0.005673057, 0.004722194, -0.023120679, 0.0025104724, 0.012403689,
-0.13642642, 0.030334473, -0.09983344, -0.057932, 0.08075272, -0.07457211,
-0.04153603, 0.05402254, 0.0048057153, -0.092079565, -0.031400982, -0.025403475,
0.06503457, 0.07299845, 0.008329692, 0.12034205, -0.013988294, -0.08702971,
-0.033693817, 0.0049599097, 0.054284345, 0.017578928, 0.1096866, -0.014030055,
-0.045917712, 0.039325915, -0.007512143, 0.035903126, -0.072553135, -0.09068379,
0.090405114, -0.004548725, 0.04323618, 0.040122584, 0.01965332, -0.04855186,
-0.03908338, -0.045371607, -0.037685193, -0.019465396, 0.057152998, 0.017604828,
-0.0042250785, …]
3 [0.03931536, 0.14934866, -0.03687686, 0.06997826,
0.026163736, -0.064729236, 0.13439651, -0.061476935, 0.056105636, 0.11165946,
-0.050129846, -0.12902251, 0.00247919, -0.038917176, -0.07876732, 0.028654553,
0.080566406, 0.14867583, 0.0053129653, -0.12917364, -0.026502792, 0.06690034,
0.06001209, 0.010637556, 0.020263672, 0.025078183, -0.0661127, 0.021030972,

0.020308722, -0.027532669, -0.05586751, 0.04446266, 0.054344542, 0.036779493,
0.032586962, -0.03358714, 0.021388099, 0.022880191, -0.0114252, 0.06844076,
0.0777646, -0.041097004, 0.14267114, 0.0070364815, -0.032860167, -0.049560547,
0.009008499, 0.0041896277, 0.035199847, -0.022722516, 0.0125616165, 0.088663734,
-0.051887147, -0.078186035, -0.013815744, 0.0073620025, 0.0008639381,
-0.06354777, 0.12387521, -0.11221168, -0.00999814, 0.095476426, -0.12486049,
-0.0743074, 0.017050607, 0.014225551, -0.055547804, -0.03230649, -0.020996094,
0.10645984, 0.057698566, 0.0047084265, 0.1077474, -0.031168256, -0.1866019,
-0.027895972, 0.066096716, 0.100256056, 0.014858427, 0.12954566, 0.043285552,
0.011599586, 0.10241118, -0.027648926, -0.018409366, -0.10975574, -0.10714867,
0.17626372, 0.043858845, 0.064596266, 0.08822051, 0.030767532, -0.053263348,
0.019330706, -0.047427222, -0.088012695, 0.0035146077, 0.13139126, -0.014404297,
-0.039475214, …]
4 [0.049275715, 0.06921387, 0.063079834, 0.09336344,
-0.10079956, 0.05645752, 0.07104492, -0.10560608, 0.038317423, 0.08178711,
0.106486, -0.07477824, -0.076174416, 0.045756023, 0.0034395854, 0.011967977,
0.07246653, -0.029184977, -0.07230123, -0.048497517, 0.014434814, 0.02705129,
-0.10316976, 0.024709066, 0.013402303, -0.047907513, -0.083740234, 0.009318034,
0.02456665, 0.032104492, -0.0026931763, -0.030265808, -0.019836426, 0.06219991,
0.008005778, -0.07713827, -0.026748657, -0.033828735, 0.04542033, 0.0017217001,
0.09643555, 0.0002339681, 0.17527263, 0.01776123, -0.056050617, -0.14034016,
-0.018351236, 0.031562805, 0.014419556, 0.034606934, -0.00544858, -0.04771932,
-0.029647827, -0.013257344, -0.023708979, -0.019439697, -0.009737651,
-0.07274628, 0.18127441, -0.021690369, -0.108052574, 0.0071004233, -0.03988393,
-0.0938975, -0.02480062, 0.030893961, -0.12545776, 0.035512287, -0.06664022,
-0.043492477, 0.06363932, 0.023325602, 0.06928507, -0.035418194, -0.13576253,
-0.07285563, 0.007507324, 0.10695394, -0.008010864, 0.09218343, 0.03698985,
-0.06663767, 0.03546397, 0.031026205, 0.0152282715, -0.02706655, -0.0692037,
-0.020004272, -0.025911966, 0.08627319, 0.038533527, -0.013366699, -0.012939294,
-0.11568197, -0.013442993, -0.13920085, 0.06447347, 0.054706573, -0.0320638,
0.025349936, …]
…
…
293121 [0.071772255, 0.026529947, 0.044799805,
0.058308918, 0.039835613, 0.014485677, 0.013509114, -0.11311849, 0.092936195,
0.013916016, -0.052897137, -0.049397785, -0.045898438, -0.010498047,
-0.06518555, 0.2466634, 0.042236328, -0.03963216, -0.05904134, -0.013102214,
-0.02360026, 0.13964844, 0.040039062, -0.06542969, 0.04675293, -0.029459635,
-0.051554363, 0.04425049, 0.09000651, 0.003092448, -0.051371258, 0.00056966144,
0.0349528, -0.016520182, -0.034016926, 0.0015055338, 0.044433594, 0.10123698,
-0.04795329, 0.08092499, 0.11832682, -0.2718099, 0.17976888, 0.0875651,
0.06941732, 0.03270467, 0.071370445, -0.08780924, 0.12817383, 0.034016926,
-0.034261066, 0.09944662, -0.065348305, 0.027913412, -0.12736003, -0.04650879,
-0.040740967, 0.016927084, 0.008219401, -0.09773763, -0.06217702, 0.15413411,
-0.03149414, -0.061645508, -0.010050456, 0.055664062, -0.06241862, 0.044840496,
-0.0820109, 0.08886719, -0.0538737, 0.034261066, 0.03564453, 0.104654945,
-0.2998047, -0.008995692, 0.023234049, 0.089274086, 0.07779948, 0.01570638,

-0.029622396, -0.08439127, -0.022135416, -0.024902344, -0.013671875,
-0.024251303, -0.12727864, 0.047851562, 0.007832845, -0.020833334, -0.07744344,
0.10374451, -0.097717285, -0.06730143, -0.041097004, 0.00032552084, 0.04703776,
0.0061035156, 0.07255045, -0.010253906, …]
293122 [0.022968965, 0.015930925, 0.011097628,
0.101491295, -0.05075223, 0.033032734, 0.05974923, -0.057849247, 0.08404438,
0.09658634, -0.0119134495, -0.110803045, -0.01959827, -0.0007728128, -0.0971793,
0.13195471, 0.04240927, 0.087653965, -0.018551696, -0.057896186, 0.020751055,
0.08156706, 0.0747172, 0.01603938, 0.031492196, -0.032588586, -0.06948553,
0.055533502, 0.040329427, 0.024298275, -0.049521353, 0.039261162, -0.035400428,
0.02336734, 0.026723394, 0.0074889986, 0.070183545, -0.01691736, 0.015896965,
0.05385814, 0.070485696, -0.051564835, 0.14659359, -0.05394311, -0.031650916,
-0.06460601, -0.019306406, 0.037514333, 0.023024615, 0.0258117, -0.023512447,
0.04367978, 0.015651328, -0.06067104, -0.02120089, 0.0075285668, -0.011056488,
-0.08557099, 0.01861318, -0.08151858, -0.016976263, 0.10547705, -0.09833227,
-0.04528472, 0.009221357, -0.06994943, -0.07273505, -0.011592201, -0.026310341,
0.0590112, 0.060134962, 0.051227495, 0.06263374, -0.007139617, -0.17555462,
-0.037142288, 0.008073943, 0.08446338, 0.011762207, 0.053726796, 0.027515337,
-0.021374123, 0.02944677, 0.01571775, -0.05405725, -0.034111246, -0.07601001,
0.16585436, 0.027549295, -0.004707486, 0.00710809, 0.048782572, -0.055291757,
-0.054830812, -0.03152466, -0.088275686, 0.019074533, 0.062412936, 0.031937916,
-0.016069, …]
293123 [0.015420353, 0.06759195, -0.0011565265, 0.13397935,
-0.05861257, 0.04734533, 0.036395803, -0.03002436, 0.034133013, 0.05351347,
-0.023799224, -0.11978868, -0.05045184, 0.026900347, -0.09385412, 0.08242259,
0.025115069, 0.11162612, 0.022541719, -0.05940516, -0.0019369686, 0.023447625,
0.023687026, -0.002568862, 0.059943706, -0.008150886, -0.078704834, 0.06904858,
-0.05774285, -0.04693065, -0.055844925, 0.10195373, 0.05354399, -0.025802612,
-0.044726204, -0.0126001695, 0.047821045, 0.017259486, 0.10038443, -0.044240166,
0.15946871, -0.09880335, 0.13900308, -0.029104793, -0.065506876, -0.11351641,
-0.10621458, -0.010673972, -0.0045489143, 0.024616353, -0.019348145,
0.102689855, -0.022223977, -0.070335835, -0.034645528, 0.043035172, -0.04565211,
-0.045266543, -0.012876342, -0.07496329, -0.06628059, 0.08120009, -0.11882737,
-0.03226112, 0.011257396, -0.080052994, -0.09484145, -0.026334874, -0.09020929,
0.04684179, 0.073876776, 0.023686128, 0.072412826, 0.004758947, -0.16550289,
-0.0559836, 0.06146689, 0.06086731, -0.011069129, 0.07941751, 0.012820973,
-0.032250796, 0.06036826, -0.0039547193, -0.03549374, 0.0015193294, -0.08861407,
0.1907739, 0.025480382, 0.061117735, -0.015563965, 0.10442756, -0.03970696,
-0.01550293, 0.011260986, -0.07460583, 0.046369664, 0.016756844, 0.05735386,
-0.0047257366, …]
293124 [0.038205747, 0.0958981, -0.0072262376,
0.06053614, -0.05842873, 0.052176017, 0.09844141, -0.13076895, 0.05081629,
0.13700132, -0.030585254, -0.11263925, -0.016624168, -0.054972332, -0.156649,
0.0020729348, -0.033642802, 0.077184044, -0.015081335, -0.0958964, -0.011787697,
0.064579435, 0.027279606, 0.011066013, 0.008176451, -0.021274708, -0.043468334,
0.11509684, 0.039140347, -0.003927019, -0.04845061, -0.0056172125, 0.054887418,
-0.013574247, 0.029556839, -0.04338992, 0.06540623, 0.016590824, 0.0033351758,

```
0.055721812, 0.10723199, -0.10597738, 0.1606044, -0.0253262, -0.032678585,
-0.12503588, -0.033578377, 0.0026457044, 0.02456326, 0.02311085, -0.0012187251,
0.0495153, -0.01928372, -0.08509205, 0.05500059, 0.06591288, 0.023976645,
-0.09231984, 0.042348687, -0.07681585, -0.0024097583, 0.049498945, -0.1407222,
-0.13482991, -0.0040204083, 0.020864593, -0.08013068, 0.036035042, -0.005819815,
0.10029581, 0.05610452, 0.0044425684, 0.076853715, -0.05296043, -0.1803312,
-0.052105162, 0.09406422, 0.051899664, 0.035418898, 0.10554787, 0.028632764,
-0.083141185, -0.028951716, -0.01657274, -0.04328523, -0.04631537, -0.066754945,
0.12846996, 0.10245344, 0.020291928, 0.048367817, 0.03612363, -0.04224537,
0.08496037, -0.04438386, -0.06132917, 0.02270423, 0.07141679, 0.029787134,
0.009420889, …]
293125                 [0.05538214, 0.07372466, 0.041464668, 0.07800874,
-0.060195196, 0.024576822, 0.10026914, -0.098379955, 0.092523485, 0.12611172,
-0.08502488, -0.09343611, -0.031261627, 0.031627838, -0.09937686, 0.08493042,
0.07966832, 0.10999407, -0.010633196, -0.062470935, -0.03979347, 0.12105742,
0.027450925, -0.017290387, 0.06804548, -0.083592005, -0.012549991, 0.08967227,
-0.0135825025, -0.011327471, 0.0399664, -0.018287295, -0.0500481, 0.020491827,
-0.01696196, -0.04703776, -0.016834076, -0.032586962, 0.008701869, 0.08329991,
0.06142462, -0.10306803, 0.09247598, -0.028921945, 0.054568335, -0.11108253,
-0.04539635, 0.0015389578, 0.02214559, 0.03482782, 0.020237515, 0.025800433,
-0.04494513, -0.008783249, -0.019880023, -0.055152528, 0.026320685,
-0.051573798, 0.027063278, -0.07673209, -0.072599865, 0.02255685, -0.060221355,
-0.095342726, -0.007975261, -0.019360498, -0.0073169526, 0.05350749, 0.03735642,
0.04263451, 0.032565095, -0.0058710007, 0.07720657, -0.055191766, -0.12954566,
-0.04704648, 0.05968657, 0.037001837, 0.028013524, 0.055537634, 0.031113397,
-0.053801037, -0.053305488, -0.04898507, -0.0069943382, -0.018156506,
-0.12706938, 0.048312232, 0.039659046, 0.044320244, 0.012486049, -0.08548628,
-0.018824985, -0.023742676, -0.056387763, -0.018572126, 0.080444336, 0.08758254,
0.04879325, -0.06321716, …]

[293126 rows x 6 columns]
```

[73]:

```python
x_review = np.array(X_data['google_word2vec'].values.tolist())
df_temp = pd.DataFrame(x_review, columns=[f"wv_{i}" for i in range(300)])
```

[74]:

```python
df_temp
```

[74]:

```
            wv_0      wv_1      wv_2      wv_3      wv_4      wv_5      wv_6  \
0       0.043111 -0.009413  0.017002  0.100708 -0.019506  0.029783  0.098273
1      -0.006906  0.063821  0.041083  0.040674 -0.014672  0.008882  0.053463
2       0.030339  0.039685  0.050813  0.120442  0.007759 -0.016965  0.090223
3       0.039315  0.149349 -0.036877  0.069978  0.026164 -0.064729  0.134397
4       0.049276  0.069214  0.063080  0.093363 -0.100800  0.056458  0.071045
…            …         …         …         …         …         …
293121  0.071772  0.026530  0.044800  0.058309  0.039836  0.014486  0.013509
293122  0.022969  0.015931  0.011098  0.101491 -0.050752  0.033033  0.059749
293123  0.015420  0.067592 -0.001157  0.133979 -0.058613  0.047345  0.036396
```

```
293124  0.038206  0.095898 -0.007226  0.060536 -0.058429  0.052176  0.098441
293125  0.055382  0.073725  0.041465  0.078009 -0.060195  0.024577  0.100269

            wv_7      wv_8      wv_9  …    wv_290    wv_291    wv_292  \
0      -0.069442  0.005043  0.066694  … -0.103058 -0.003955 -0.114782
1      -0.117234  0.057921  0.121474  … -0.026797 -0.039454 -0.047541
2      -0.081337 -0.026020  0.059557  … -0.109985  0.056882 -0.095061
3      -0.061477  0.056106  0.111659  … -0.082159  0.017270 -0.173549
4      -0.105606  0.038317  0.081787  … -0.070882  0.039983 -0.040914
…            …         …         …  …         …         …         …
293121 -0.113118  0.092936  0.013916  …  0.069407  0.125570 -0.047201
293122 -0.057849  0.084044  0.096586  … -0.033710  0.053456 -0.089812
293123 -0.030024  0.034133  0.053513  … -0.057383 -0.024374 -0.073713
293124 -0.130769  0.050816  0.137001  … -0.053699 -0.014183 -0.094532
293125 -0.098380  0.092523  0.126112  … -0.033180  0.075056 -0.069287

          wv_293    wv_294    wv_295    wv_296    wv_297    wv_298    wv_299
0       0.019799 -0.105607 -0.052619  0.042555 -0.049991  0.016832 -0.047917
1       0.066873  0.034781 -0.067570  0.033986 -0.008218 -0.010667  0.011889
2       0.009425 -0.040884 -0.083371  0.089539 -0.062484 -0.013627 -0.006785
3      -0.027745 -0.014704 -0.023193  0.032708 -0.128273  0.020790 -0.065505
4      -0.021336 -0.037570 -0.023885 -0.034219 -0.055471  0.044899 -0.019174
…            …         …         …         …         …         …         …
293121 -0.085815 -0.112691 -0.085286 -0.128092 -0.057739 -0.144043  0.010396
293122  0.052977  0.013515 -0.016510 -0.006385 -0.070121  0.031087 -0.006506
293123  0.088860 -0.025497 -0.005793  0.028541 -0.020043  0.020144 -0.051395
293124  0.022514  0.021254 -0.018901  0.049044 -0.019012  0.041439 -0.069279
293125 -0.011533  0.015564  0.009677 -0.000605 -0.054683  0.091115 -0.035851

[293126 rows x 300 columns]
```

[75]:
```python
x_data_final = pd.concat([X_data, df_temp], axis=1)
x_data_final = x_data_final.drop(columns=['google_word2vec'])
```

[76]:
```python
x_data_final
```

[76]:
```
        helpful_votes  total_votes  vine  verified_purchase  review_date  \
0                51.0         51.0     0                  1            9
1                 0.0          0.0     0                  1           41
2                 0.0          0.0     0                  1            8
3                 0.0          0.0     0                  1           34
4                 0.0          0.0     0                  1           28
…                   …            …   …                  …            …
293121            0.0          0.0     0                  1           28
293122           15.0         16.0     0                  0           28
293123            3.0          4.0     0                  1           36
293124            9.0          9.0     0                  1           11
```

```
293125            0.0          0.0     0                  1           33

            wv_0      wv_1      wv_2      wv_3      wv_4  …    wv_290  \
0       0.043111 -0.009413  0.017002  0.100708 -0.019506  … -0.103058
1      -0.006906  0.063821  0.041083  0.040674 -0.014672  … -0.026797
2       0.030339  0.039685  0.050813  0.120442  0.007759  … -0.109985
3       0.039315  0.149349 -0.036877  0.069978  0.026164  … -0.082159
4       0.049276  0.069214  0.063080  0.093363 -0.100800  … -0.070882
…            …         …         …         …         …  …       …
293121  0.071772  0.026530  0.044800  0.058309  0.039836  …  0.069407
293122  0.022969  0.015931  0.011098  0.101491 -0.050752  … -0.033710
293123  0.015420  0.067592 -0.001157  0.133979 -0.058613  … -0.057383
293124  0.038206  0.095898 -0.007226  0.060536 -0.058429  … -0.053699
293125  0.055382  0.073725  0.041465  0.078009 -0.060195  … -0.033180

          wv_291    wv_292    wv_293    wv_294    wv_295    wv_296    wv_297  \
0      -0.003955 -0.114782  0.019799 -0.105607 -0.052619  0.042555 -0.049991
1      -0.039454 -0.047541  0.066873  0.034781 -0.067570  0.033986 -0.008218
2       0.056882 -0.095061  0.009425 -0.040884 -0.083371  0.089539 -0.062484
3       0.017270 -0.173549 -0.027745 -0.014704 -0.023193  0.032708 -0.128273
4       0.039983 -0.040914 -0.021336 -0.037570 -0.023885 -0.034219 -0.055471
…            …         …         …         …         …         …         …
293121  0.125570 -0.047201 -0.085815 -0.112691 -0.085286 -0.128092 -0.057739
293122  0.053456 -0.089812  0.052977  0.013515 -0.016510 -0.006385 -0.070121
293123 -0.024374 -0.073713  0.088860 -0.025497 -0.005793  0.028541 -0.020043
293124 -0.014183 -0.094532  0.022514  0.021254 -0.018901  0.049044 -0.019012
293125  0.075056 -0.069287 -0.011533  0.015564  0.009677 -0.000605 -0.054683

          wv_298    wv_299
0       0.016832 -0.047917
1      -0.010667  0.011889
2      -0.013627 -0.006785
3       0.020790 -0.065505
4       0.044899 -0.019174
…            …         …
293121 -0.144043  0.010396
293122  0.031087 -0.006506
293123  0.020144 -0.051395
293124  0.041439 -0.069279
293125  0.091115 -0.035851

[293126 rows x 305 columns]
```

[77]:
```
y_data.shape
```

[77]:
```
(293126,)
```

[78]:

```
np.save('X_data_Kitch_finalized.npy', x_data_final ) #data_k_concat
np.save('X_Kitch_review_wv.npy', df_temp ) #data_k_concat
np.save('y_data_Kitch_finalized.npy', y_data ) #data_k_concat
```

Data_home

December 6, 2021

[1]:
```
import gzip
import pandas as pd
import numpy as np
import nltk
import requests
from io import BytesIO
import imblearn
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('wordnet')
import re
import contractions
from sklearn.preprocessing import LabelEncoder
from bs4 import BeautifulSoup
pd.set_option('display.max_colwidth', None)
import requests
from sklearn.linear_model import Perceptron
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
import gensim.downloader as api
from gensim.models import Word2Vec
wv = api.load('word2vec-google-news-300')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\yasmi\AppData\Roaming\nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\yasmi\AppData\Roaming\nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[2]:
```
def read_file(link):
    url = link
    data_compressed = requests.get(url).content
    data = pd.read_csv(BytesIO(data_compressed), compression='gzip',␣
 ↪sep='\t',error_bad_lines=False, warn_bad_lines=False)
    return data
```

```python
# Remove HTML tags funtion
def remove_tags(txt):
    # parse html content
    soup = BeautifulSoup(txt, "html.parser")

    # get tags content
    for data in soup(['style', 'script']):
        data.get_text()

    # return html's tag content
    return ' '.join(soup.stripped_strings)

# Remove URLS funtion
def remove_urls(txt):
    return re.sub(r"http\S+", "", txt)

# Apply contraction to words
def contractionfunction(s):
    expanded_words = []
    for word in s.split():
        expanded_words.append(contractions.fix(word))
    result = ' '.join(expanded_words)
    return result

def remove_non_alphabetical(txt):
    regex = re.compile('[\W_0-9]+')
    dirty_list = txt.split()
    clean_list = [regex.sub(' ', word) for word in dirty_list]
    clean_string = ' '.join(clean_list)
    return clean_string



# remove stop words function
def remove_stop_words(txt):

    stop = stopwords.words('english')
    word_list = txt.split()
    clean_list = []
    clean_string = ''
    for word in word_list:
        if word not in stop:
            clean_list.append(word)
    clean_string = ' '.join(clean_list)
    return clean_string
```

```
def leammatize_review(txt):
    lemmatizer = WordNetLemmatizer()
    word_list = txt.split()
    clean_list = []
    clean_string = ''
    for word in word_list:
        new_word = lemmatizer.lemmatize(word)
        clean_list.append(new_word)
    clean_string = ' '.join(clean_list)
    return clean_string
```

[15]:
```
data_kitchen = read_file("https://s3.amazonaws.com/amazon-reviews-pds/tsv/
↪amazon_reviews_us_Kitchen_v1_00.tsv.gz")
data_home = read_file("https://s3.amazonaws.com/amazon-reviews-pds/tsv/
↪amazon_reviews_us_Home_Improvement_v1_00.tsv.gz")
```

```
C:\Users\yasmi\anaconda3\lib\site-
packages\IPython\core\interactiveshell.py:3338: DtypeWarning: Columns (7) have
mixed types.Specify dtype option on import or set low_memory=False.
  if (await self.run_code(code, result,  async_=asy)):
```

[16]:
```
print(data_kitchen.shape)
print(data_home.shape)
```

```
(4874890, 15)
(2629867, 15)
```

[17]:
```
data_kitchen['star_rating'].value_counts()
```

[17]:
```
5.0    3124759
4.0     731733
1.0     426900
3.0     349547
2.0     241948
Name: star_rating, dtype: int64
```

[18]:
```
data_home['star_rating'].value_counts()
```

[18]:
```
5             1188230
5              465211
4              307848
1              182407
3              138622
4              110012
2               91848
1               63180
3               50053
```

```
2               32418
2015-07-03          1
2015-02-15          1
2014-11-17          1
2014-12-03          1
2015-06-03          1
2014-08-09          1
2013-10-30          1
2015-05-15          1
2014-03-13          1
2014-09-01          1
2014-01-19          1
Name: star_rating, dtype: int64
```

1 Data_home Preproc

[19]:
```
data_home_1 = data_home.copy()
```

[26]:
```
b = data_home['star_rating'].isin ([1,2,3,4,5])
data_home = data_home[b]
```

[33]:
```
data_home['star_rating'].value_counts()
```

[33]:
```
5    1188230
4     307848
1     182407
3     138622
2      91848
Name: star_rating, dtype: int64
```

[37]:
```
data_home.head()
```

[37]:
```
  marketplace  customer_id       review_id  product_id  product_parent  \
0          US     48881148  R215C9BDXTDQOW  B00FR4YQYK       381800308
1          US     47882936  R1DTPUV1J57YHA  B00439MYYE       921341748
2          US     44435471   RFAZK5EWKJWOU  B00002N762        56053291
3          US     28377689  R2XT8X000WS1AL  B000QFCP1G       595928517
4          US     50134766  R14GRNANKO2Y2J  B00WRCRKOI       417053744

                                       product_title  \
0                                                                SadoTech Model C
Wireless Doorbell Operating at over 500-feet Range with Over 50 Chimes, No
Batteries Required for Receiver, (Various Colors)
1
iSpring T32M 3.2 Gallon Residential Pressurized Water Storage Tank for Reverse
Osmosis (RO) Systems
2
```

```
Schlage F10CS V ELA 626 Elan Light Commercial Passage Lever, Satin Chrome
3
Citri-Strip QCG731 Paint and Varnish Stripping Gel, 1-Quart
4  SleekLighting Bulb Adapters / Converts Standard (E26) bulb base to (E12)
Chandelier Socket Candelabra/ Screw Enlarger Adapter/ Wear and Tear Resistant/
Generates Low Heat-Lightweight/UL Listed Set of 12

   product_category star_rating  helpful_votes  total_votes vine  \
0  Home Improvement           4            0.0          0.0    N
1  Home Improvement           5            0.0          0.0    N
2  Home Improvement           5            0.0          0.0    N
3  Home Improvement           5            0.0          0.0    N
4  Home Improvement           5            0.0          0.0    N

  verified_purchase  \
0                 Y
1                 Y
2                 Y
3                 Y
4                 Y

                                                       review_headline  \
0                                                           Four Stars
1                                           Good price, quick shipment
2                                                           Five Stars
3  Although *slightly* stronger paint removers can be found, this …
4                                                       Great Adapters

review_body  \
0
good product
1                                               Good price, quick shipment.
Adequate packaging. Hooked right up to my existing plumbing so it was an easy
replacement.
2
Excellent…!
3  Although *slightly* stronger paint removers can be found, this is far less
hazardous (got it on my skin with minimal consequence; other strippers burned
horribly)
4                                                                        These
adapters are well made and easy to use.  Much easier than rebuilding a light
fitting.

  review_date
0  2015-08-31
1  2015-08-31
2  2015-08-31
```

```
3   2015-08-31
4   2015-08-31
```

[38]:
```
def review_num_words(txt):
    return len(txt.split())
```

[39]:
```
data_home = data_home.dropna().reset_index(drop=True)
data_home['review_length'] = data_home['review_body'].apply(lambda body :
 ↪review_num_words(body))
```

[48]:
```
data_h = data_home.loc[data_home['review_length'] > 30]
```

[49]:
```
data_h.shape
```

[49]:
```
(960885, 16)
```

[50]:
```
data_h['star_rating'].value_counts()
```

[50]:
```
5    524072
4    174341
1    115756
3     86193
2     60523
Name: star_rating, dtype: int64
```

[51]:
```
data_h['star_rating'] = np.where(data_h['star_rating'] > 3.0 , 1, 0)
```

```
<ipython-input-51-61c54dfb3408>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_h['star_rating'] = np.where(data_h['star_rating'] > 3.0 , 1, 0)
```

[52]:
```
data_h['star_rating'].value_counts()
```

[52]:
```
1    698413
0    262472
Name: star_rating, dtype: int64
```

[53]:
```
data_h1 = data_h.copy()
```

[54]:
```
from sklearn.utils import resample

df_majority = data_h[data_h["star_rating"]==1]
df_minority = data_h[data_h["star_rating"]==0]
```

```
df_majority_downsampled = resample(df_majority,
                                   replace=False,    # sample without replacement
                                   n_samples=len(df_minority),     # to match␣
 ↪minority class
                                   random_state=123) # reproducible results

# Combine minority class with downsampled majority class
data_h_concat = pd.concat([df_majority_downsampled, df_minority])

# Display new class counts
data_h_concat['star_rating'].value_counts()
```

[54]:
```
1    262472
0    262472
Name: star_rating, dtype: int64
```

[64]:
```
data_h_concat['star_rating'].value_counts()
```

[64]:
```
1    262472
0    262472
Name: star_rating, dtype: int64
```

[58]:
```
#make decoder
data_h_concat['vine'] = data_h_concat['vine'].astype('category').cat.codes
data_h_concat['verified_purchase'] = data_h_concat['verified_purchase'].
 ↪astype('category').cat.codes
data_h_concat['product_id'] = data_h_concat['product_id'].astype('category').
 ↪cat.codes
data_h_concat['customer_id'] = data_h_concat['customer_id'].astype('category').
 ↪cat.codes

#removing unnecessary columns
data_h_concat = data_h_concat.drop( columns=['marketplace', 'product_category',␣
 ↪'review_id', 'product_parent', 'product_title'] )

#combining two features in one feature
data_h_concat['reviews'] = data_h_concat['review_headline'] + ' ' +␣
 ↪data_h_concat['review_body']
data_h_concat = data_h_concat.drop( columns=['review_headline', 'review_body']␣
 ↪)  # drop the 2 columns
data_h_concat['reviews'] = data_h_concat['reviews'].str.lower()  # make␣
 ↪lowercase

# make time
data_h_concat["review_date"] = pd.to_datetime(data_h_concat["review_date"])
data_h_concat["review_date"] = data_h_concat["review_date"].dt.isocalendar().
 ↪week
```

```python
print(data_h_concat.shape)
data_h_concat.head(10)
```

```
(524944, 10)
```

[58]:

```
         customer_id  product_id  star_rating  helpful_votes  total_votes  \
1138359         9320       45936            1            2.0          2.0
854689         49498      106816            1            0.0          0.0
897248        413739      124090            1            0.0          0.0
1829174        44843       54237            1            0.0          0.0
976671         25113       61644            1            0.0          0.0
1655589       331123       37864            1            0.0          0.0
329081        168713      133820            1            0.0          0.0
725521        162100       15552            1            2.0          2.0
219245        314754       86409            1            0.0          0.0
1375122       418636        4632            1            6.0          7.0

         vine  verified_purchase  review_date  review_length  \
1138359     0                  1           49             68
854689      0                  0           30            130
897248      1                  0           28            193
1829174     0                  1            7             52
976671      0                  1           21             52
1655589     0                  1           11             37
329081      0                  1           15             55
725521      0                  1           43             42
219245      0                  1           25             45
1375122     0                  1           15            132

                             reviews
1138359
wireless battery doorbell button-oil rubbed bronze this rating because it looks
good this is an oil-rubbed bronze . when you buy it and before installing it
make sure<br />you pre drill the holes with the smallest drill then you can use
the screw driver to put the screw in total there are 4<br />screws . and you put
the cover there is a tiny screw to hold it. be careful not to loose it.
854689
reliable control! i had the thermostat professionally installed along with a new
lennox heat pump purchased from costco. i have had it just over a couple of
weeks now and it is working flawlessly.  very easy to fully control unit from
the iphone app and i never have any connectivity issues whatsoever.  there are a
ton of options on the thermostat that would be nice to be able to control from
the phone, in particular the one to disable its ability to start cooling before
the designated time in order to make sure the temp is correct when you hit that
time. when you are trying to avoid running pwr at a designated time that option
sucks.<br /><br />other than that i would purchase this unit in a heartbeat all
```

over again.
897248   love these command hooks in all kinds on sizes! i use lots of command hooks in my house. i use them of course for hanging clothes, but also the ones for mops, the cord clips, the shower baskets (wish they still sold the metal ones!), etc. i use them with the water resistant strips to hold other baskets in the shower because those suction cups never work well for me. i have hung clocks, pictures and small magazine racks with the strips even with items not sold by command. these are another size in my arsenal!

they are nice and small and clear. i use them in my kitchen behind my coffee maker to sort of hold the cord out of the way. i know they are designed for outdoor lights, but since it is summer, i don't need them right now. i have noticed that it really does help to clean the area with rubbing alcohol first. i have been lazy and skipped it, and while they usually hold for me, they don't hold as well. i have also used hand sanitizer (not scented/colored) when i couldn't find my bottle of rubbing alcohol and it seemed to work well, too.
1829174
so adorable. this item shipped fast and is entirely too cute. i have a 5-year-old that loves monkeys. we put this at the entrance to her bedroom and it looks great and was easy to put up and i am sure will be just as easy to remove. what a wonderful product. thank you.
976671
works for minor items and bottle opener is neat don't know if i would put this thing through any serious use. i just keep one on my pack in case i need to slap on something quickly. works for naglene bottle, eyeglass case, house keys, flashlight, chemstick when night hiking, etc. bottle opener has come in handy as well. neat item.
1655589
great purchase! i received package promptly… very secure packaging. product in great condition.. fit perfectly.. easy to install! you don't even need a wrench as recommended on package! matches my new decor perfectly! you can tell it's high quality!
329081
the perfect lamp for soldiers or contractors this product is perfect for the troops or contractors who have to live in austere conditions and need a dependable lamp. sometimes incandescent lamps are available in px's but don't last long and break all too easily. this lamp is heavy duty, can stand up to abuse and keep lit night after night. bravo phillips!
725521
top sink for the money!!! great sink…installed over 100 of these. still looks new after 10 years. don't buy the blanco sink cleaner..junk. use a mr. clean magic eraser cleaner, then seal with a stone/granite sealer…do this every six months and it will always look new!! enjoy
219245
set it and forget it… works great. this is the version that can be used with led lights. it has a "click" when turning on and off but that solid switch is required for led's and higher wattage loads. it's easy enough to program but the display is very small.

```
1375122
this hobbyist and collector loves museum gel i am a hobbyist who must pose
dolls, often holding fans or other small objects in their hands. glue is either
too permanent or doesn't hold, depending on the materials being secure. i now
use museum gel or wax to anchor the objects in the dolls' hands, and also to
secure dolls to their stands. i have not had problems removing either museum wax
or gel from lacquered stands, and i have yet to encounter staining problems, but
that could change as i use the products on more dolls. i prefer museum gel or
wax due to their transparent qualities. i first used neutral museum putty to
secure my limoges boxes, but museum wax and gel are not as noticeable under the
feet of some of the more elaborate limoges box figures.
```

[60]:
```
 # Applying preproccessing
data_h_concat['reviews'] = data_h_concat['reviews'].apply(lambda body :
 ↪remove_tags(body))
data_h_concat['reviews'] = data_h_concat['reviews'].apply(lambda body :
 ↪remove_urls(body))
data_h_concat['reviews'] = data_h_concat['reviews'].apply(lambda body :
 ↪contractionfunction(body))
data_h_concat['reviews'] = data_h_concat['reviews'].apply(lambda body :
 ↪remove_non_alphabetical(body))
data_h_concat['reviews'] = data_h_concat['reviews'].apply(lambda review:
 ↪remove_stop_words(review))
data_h_concat['reviews'] = data_h_concat['reviews'].apply(lambda txt :
 ↪leammatize_review(txt))
```

[65]:
```
#data_h_concat.to_csv(index=False)
np.save('data_home_preproc',data_h_concat)
```

[67]:
```
X = data_h_concat.drop( columns=['star_rating'] )
y = data_h_concat['star_rating']
```

[68]:
```
indexs = X['reviews'].index
X['google_word2vec'] = pd.Series(dtype=object)
for idx , review in zip(indexs, X['reviews']):
    unseen_words = 0
    n = len(review.split())
    x = 0
    for word in review.split():
        try:
            x = x + wv[word]
        except KeyError:
            unseen_words = unseen_words + 1
    if unseen_words == n:
        X.at[idx, 'google_word2vec'] = np.NaN
        continue
    x = x/(n-unseen_words)
```

```
    x1 = x.reshape(-1, 1)
    x1 = x1.T
    X.at[idx, 'google_word2vec'] = x1[0]
```

[137]:
```
#x_data = X.copy()
x_data_1 = X.copy()
x_data_2 = X.copy()
x_data_3 = X.copy()
```

[71]:
```
x_data.head()
x_data= x_data.drop( columns=['reviews'] )
```

[72]:
```
x_data.isna().sum()
```

[72]:
```
customer_id          0
product_id           0
helpful_votes        0
total_votes          0
vine                 0
verified_purchase    0
review_date          0
review_length        0
google_word2vec      1
dtype: int64
```

[73]:
```
index_na = x_data.loc[pd.isna(x_data["google_word2vec"]), :].index
x_data = x_data.drop(index_na)
```

[75]:
```
y_data = y.copy()
y_data = y_data.drop(index_na)
y_data.shape
```

[75]:
```
(524943,)
```

[77]:
```
x_data.head()
```

[77]:
```
         customer_id  product_id  helpful_votes  total_votes  vine  \
1138359         9320       45936            2.0          2.0     0
854689         49498      106816            0.0          0.0     0
897248        413739      124090            0.0          0.0     1
1829174        44843       54237            0.0          0.0     0
976671         25113       61644            0.0          0.0     0

         verified_purchase  review_date  review_length  \
1138359                  1           49             68
854689                   0           30            130
897248                   0           28            193
1829174                  1            7             52
```

976671 1 21 52

google_word2vec

1138359 [-0.0019353231, 0.033991072, -0.0065320334, 0.007509073, -0.061800003, -0.063817345, 0.044388667, -0.1020813, 0.111635, 0.12527126, -0.063338384, -0.087317996, -0.029407077, 0.056371585, -0.120023094, 0.13001019, 0.029518763, 0.078440346, -0.04760064, -0.11397934, 0.06069692, 0.051098295, 0.025704278, 0.076730095, 0.06168874, 0.023102654, -0.08256361, 0.11957423, 0.047231037, -0.04843309, -0.07201958, 0.108181424, -0.026432462, -0.028561063, 0.022979736, -0.11117008, 0.124242574, 0.07008464, 0.050707076, 0.101881236, 0.09727754, 0.03925986, 0.20121786, -0.024442753, 0.027826944, -0.017541673, -0.039977998, 0.026918411, 0.039871216, 0.005352444, -0.071758695, 0.04900911, 0.008936564, -0.122918025, 0.0076963636, 0.02687645, -0.060066223, -0.100750394, 0.0067215497, -0.036629573, -0.0121114515, -0.010609097, -0.08617655, -0.061703153, 0.043986004, -0.047125604, -0.057662115, -0.013578627, -0.022769503, 0.0719045, 0.07237074, -0.0043538413, 0.07975939, -0.098241806, -0.17044279, -0.13766988, 0.08961487, 0.06496599, 0.03226047, 0.017685361, 0.045296643, -0.030741373, 0.010861291, 0.06644016, -0.006873237, -0.033524234, -0.045980666, 0.10571374, -0.017659506, 0.056980133, 0.08440544, 0.102313995, 0.056226093, 0.0090806745, -0.020680746, -0.0820516, 0.0021091038, -0.042958576, 0.04707633, -0.003968557, …]

854689 [0.023615602, 0.02136612, -0.014350766, 0.08793065, -0.09409045, 0.012962966, 0.034358792, -0.05522256, 0.10358967, 0.03548544, -0.006372796, -0.09998437, -0.01909431, -0.015580201, -0.07437171, 0.0389737, 0.028192239, 0.03236189, 0.030874284, -0.079905026, 0.019980008, 0.06815338, 0.0038312068, 0.036571063, 0.016687112, 0.014041338, -0.108033106, 0.111968175, -0.039645337, -0.0020274178, -0.019974506, 0.008182213, -0.03420064, -0.018038016, -0.0037441566, -0.07646354, 0.017165387, -0.07152119, 0.0017095942, 0.038785342, 0.089394495, -0.059645858, 0.11927814, -0.049067263, -0.03915105, -0.058131985, 0.009340398, 0.0154660335, -0.04823028, -0.0033952994, -0.023551567, 0.025857395, 0.0163184, -0.06353885, -0.019750375, 0.04094771, -0.022922484, -0.07370045, 0.041088667, -0.02456765, -0.0246527, 0.041629855, -0.108763896, -0.028739555, 0.020223148, -0.023667632, -0.08405779, 0.087284274, -0.07858633, 0.07024396, 0.08508288, 0.04218855, 0.07737685, -0.062583424, -0.14200279, -0.119793504, 0.10644681, 0.04805418, 0.0037183918, 0.048693046, 0.07865343, 0.022508966, 0.06251688, 0.0025245792, 0.012562379, 0.0017826518, -0.025722316, 0.1208441, -0.0069828345, 0.059383832, 0.05765421, 0.026532533, -0.06360124, -0.07541844, -0.033930544, -0.050897818, -0.00025522514, 0.021186203, 0.047301747, 0.008874112, …]

897248 [0.047544874, 0.047309197, -0.013163047, 0.0661334, -0.08214547, 0.0037330778, 0.06863653, -0.11218322, 0.0741428, 0.101349555, -0.03356669, -0.13560198, -0.0030550815, 0.01579904, -0.10682629, 0.04618503, 0.02733529, 0.09277102, 0.0033932677, -0.07859213, 0.0330945, 0.06269217, 0.011297132, 0.0305907, 0.020050956, -0.019888321, -0.07383109, 0.09879099, -0.0022854002, -0.030739058, -0.02898777, 0.013334897, -0.008174783, -0.02697382, 0.010568043, -0.058901947, 0.05490082, -0.020846829, 0.0020328937, 0.04096422, 0.065138415, -0.013813963, 0.13123374, -0.05828578, -0.03961544,

-0.052979764, -0.06649569, 0.0061758817, 0.020718716, 0.023212094, -0.010775122,
0.0059605776, -0.0058151092, -0.073311, 0.02533382, 0.0019328806, 0.011032558,
-0.039032284, 0.020784643, -0.030946108, -0.020223146, 0.05824072, -0.089479804,
-0.07050962, 0.031081397, 0.0035219097, -0.07448729, 0.03456992, -0.027764803,
0.08381638, 0.09473231, 0.016339444, 0.09176651, -0.004735871, -0.15488608,
-0.06612789, 0.05411318, 0.076462545, 0.04037649, 0.044587087, 0.033960212,
-0.06473194, 0.06388515, 0.0022090496, -0.02583026, -0.008607675, -0.06205538,
0.11983233, 0.030476259, 0.036770057, -0.013356879, 0.056070007, -0.027783083,
-0.059916582, -0.016190028, -0.085060954, 0.04067505, -0.019403892, 0.010204277,
-0.0048593953, …]
1829174 [0.03422878, 0.04633298, -0.04310807, 0.10887544, -0.046490215,
-0.05856456, 0.11932108, -0.071522586, 0.08140763, 0.08678669, -0.022142757,
-0.102395765, -0.018349025, -0.07013736, -0.106255576, 0.07125821, 0.104030274,
0.07175612, -0.027008057, 0.006426604, 0.07456164, 0.05057028, 0.033926923,
0.01852815, 0.04986307, -0.049271293, -0.007090693, 0.031010794, 0.035984207,
-0.08862305, -0.059522547, 0.027126146, 0.030540135, 0.054432746, 0.060333252,
-0.074792616, 0.08081934, -0.032170836, -0.055567864, 0.11552098, 0.09398352,
-0.026128354, 0.13503133, -0.013849673, -0.018830672, -0.10705699, -0.021697288,
0.02913898, -0.0028832478, -0.007446289, -0.029341988, 0.034446385,
-0.0027678118, -0.10300877, 0.0066929073, 0.028596962, -0.023819799,
-0.09535085, 0.06485848, -0.045650315, -0.04037874, 0.06760705, -0.10111137,
-0.06966765, 0.03132762, -0.025241852, -0.08836431, 0.030182237, -0.093307495,
0.08400826, 0.13591203, 0.06983301, -0.017822266, -0.034516044, -0.16754416,
-0.002105713, 0.041909922, 0.024342412, 0.08663011, 0.10384269, -0.020672342,
-0.024886422, 0.031103466, 0.032608695, -0.017795729, -0.044358626, -0.07607245,
0.102732785, 0.06948919, 0.046396006, -0.0020619268, 0.0700259, -0.092322305,
-0.048500393, -0.047732145, -0.050797172, 0.011920432, 0.10182124, 0.016617484,
-0.03802092, …]
976671 [0.058544263, 0.039916992, -0.059540413, 0.07721525, -0.068433605,
0.050970953, 0.13428105, -0.07326858, 0.055212896, 0.10193717, 0.005168193,
-0.13075875, -0.033471186, 0.014821233, -0.09955494, 0.084436364, 0.08692396,
0.11412914, 0.020678753, -0.047908474, 0.08908494, 0.04359972, -0.012765833,
-0.014662871, 4.4539167e-05, 0.024010323, -0.038609684, 0.047254406,
-0.003858824, 0.0011332744, -0.056227814, 0.029233571, -0.0024628511,
0.01180123, 0.02041894, -0.063756995, 0.090112634, -0.009497565, 0.013491244,
0.04006556, 0.023379764, 0.026136244, 0.19521682, -0.0021465404, -0.06180057,
-0.034680136, -0.04625475, -0.021673873, 0.048503153, 0.011954643, -0.073154345,
0.008894637, -0.006946873, -0.09184286, 0.044012893, -0.042948954, 0.034969434,
-0.046889022, 0.068829924, -0.050095018, -0.07475157, 0.043615393, -0.069884844,
-0.052924905, -0.02163449, 0.021533864, -0.04703666, 0.0035375648, -0.019788587,
0.029748762, 0.09740056, 0.02656184, 0.14094007, -0.040804476, -0.1884778,
-0.01383766, 0.034770243, 0.08736358, 0.068874046, -0.0015347454, -0.0054448103,
-0.078397594, 0.016789617, 0.011357488, 0.002203864, -0.053705577, -0.083432175,
0.074712396, 0.03936922, 0.07261864, 0.02286014, 0.104544975, -0.023737727,
-0.07776539, -0.021282298, -0.07619909, 0.024304364, 0.049033914, -0.024582941,
0.03415082, …]

[78]:
```python
x_review = np.array(x_data['google_word2vec'].values.tolist())
```

[79]:
```python
x_review.shape
```

[79]:
```
(524943, 300)
```

[149]:
```python
x_data.index
```

[149]:
```
RangeIndex(start=0, stop=524943, step=1)
```

[80]:
```python
df_temp = pd.DataFrame(x_review, columns=[f"wv_{i}" for i in range(300)])
```

[81]:
```python
df_temp
```

[81]:
```
            wv_0      wv_1      wv_2      wv_3      wv_4      wv_5      wv_6  \
0      -0.001935  0.033991 -0.006532  0.007509 -0.061800 -0.063817  0.044389
1       0.023616  0.021366 -0.014351  0.087931 -0.094090  0.012963  0.034359
2       0.047545  0.047309 -0.013163  0.066133 -0.082145  0.003733  0.068637
3       0.034229  0.046333 -0.043108  0.108875 -0.046490 -0.058565  0.119321
4       0.058544  0.039917 -0.059540  0.077215 -0.068434  0.050971  0.134281
…            …         …         …         …         …         …         …
524938  0.030233  0.010781  0.008298  0.101231  0.016175  0.029355  0.086017
524939  0.044684  0.017321 -0.002653  0.039392 -0.094506  0.072984  0.078428
524940  0.022168  0.012010 -0.032303  0.072634 -0.092907  0.037132  0.008967
524941  0.021636  0.010144  0.025300  0.060762 -0.048598  0.005075  0.124751
524942 -0.032589  0.081841 -0.008103  0.070638 -0.032776  0.132317  0.065097

            wv_7      wv_8      wv_9  …    wv_290    wv_291    wv_292  \
0      -0.102081  0.111635  0.125271  … -0.002348  0.118568 -0.120003
1      -0.055223  0.103590  0.035485  … -0.062802  0.096626 -0.095528
2      -0.112183  0.074143  0.101350  … -0.058634  0.032414 -0.076663
3      -0.071523  0.081408  0.086787  … -0.069612  0.091083 -0.077241
4      -0.073269  0.055213  0.101937  … -0.035501 -0.005313 -0.082816
…            …         …         …  …         …         …         …
524938 -0.011671  0.091160  0.030610  … -0.076649  0.045086 -0.040504
524939 -0.065056  0.085207  0.093854  … -0.103059  0.029510 -0.076460
524940 -0.005439  0.031051  0.018739  … -0.035898  0.059629 -0.078002
524941  0.000735  0.089655  0.166371  … -0.019741  0.111197 -0.044805
524942 -0.108927  0.050151  0.057962  … -0.049353  0.012610 -0.093004

          wv_293    wv_294    wv_295    wv_296    wv_297    wv_298    wv_299
0       0.030765 -0.056905 -0.055947 -0.000999 -0.003524 -0.004047 -0.067437
1       0.002015 -0.018051  0.017691 -0.070210 -0.038866 -0.024481 -0.014990
2       0.025040 -0.013628 -0.070761  0.022408 -0.033992 -0.006797 -0.017296
3      -0.004081 -0.039301 -0.060527  0.065801 -0.031075 -0.019414 -0.031172
4       0.065549 -0.041247 -0.031801  0.009999 -0.028617  0.021594 -0.006137
…            …         …         …         …         …         …         …
524938  0.029914 -0.015632 -0.001988  0.071113 -0.014698  0.020694 -0.041840
```

```
524939  0.043552 -0.040091 -0.015740  0.029869  0.012749 -0.043952 -0.024929
524940  0.045342 -0.004021 -0.004525 -0.023154 -0.032958 -0.013741 -0.038500
524941  0.088955 -0.056041  0.042162 -0.013039 -0.053318 -0.005345 -0.079354
524942  0.103728  0.043366 -0.027395 -0.036107  0.038730  0.027465 -0.025499

[524943 rows x 300 columns]
```

[83]:
```
x_data.head()
```

[83]:
```
         customer_id  product_id  helpful_votes  total_votes  vine  \
1138359         9320       45936            2.0          2.0     0
854689         49498      106816            0.0          0.0     0
897248        413739      124090            0.0          0.0     1
1829174        44843       54237            0.0          0.0     0
976671         25113       61644            0.0          0.0     0

         verified_purchase  review_date  review_length  \
1138359                  1           49             68
854689                   0           30            130
897248                   0           28            193
1829174                  1            7             52
976671                   1           21             52

                                       google_word2vec
1138359      [-0.0019353231, 0.033991072, -0.0065320334, 0.007509073,
-0.061800003, -0.063817345, 0.044388667, -0.1020813, 0.111635, 0.12527126,
-0.063338384, -0.087317996, -0.029407077, 0.056371585, -0.120023094, 0.13001019,
0.029518763, 0.078440346, -0.04760064, -0.11397934, 0.06069692, 0.051098295,
0.025704278, 0.076730095, 0.06168874, 0.023102654, -0.08256361, 0.11957423,
0.047231037, -0.04843309, -0.07201958, 0.108181424, -0.026432462, -0.028561063,
0.022979736, -0.11117008, 0.124242574, 0.07008464, 0.050707076, 0.101881236,
0.09727754, 0.03925986, 0.20121786, -0.024442753, 0.027826944, -0.017541673,
-0.039977998, 0.026918411, 0.039871216, 0.005352444, -0.071758695, 0.04900911,
0.008936564, -0.122918025, 0.0076963636, 0.02687645, -0.060066223, -0.100750394,
0.0067215497, -0.036629573, -0.0121114515, -0.010609097, -0.08617655,
-0.061703153, 0.043986004, -0.047125604, -0.057662115, -0.013578627,
-0.022769503, 0.0719045, 0.07237074, -0.0043538413, 0.07975939, -0.098241806,
-0.17044279, -0.13766988, 0.08961487, 0.06496599, 0.03226047, 0.017685361,
0.045296643, -0.030741373, 0.010861291, 0.06644016, -0.006873237, -0.033524234,
-0.045980666, 0.10571374, -0.017659506, 0.056980133, 0.08440544, 0.102313995,
0.056226093, 0.0090806745, -0.020680746, -0.0820516, 0.0021091038, -0.042958576,
0.04707633, -0.003968557, …]
854689        [0.023615602, 0.02136612, -0.014350766, 0.08793065, -0.09409045,
0.012962966, 0.034358792, -0.05522256, 0.10358967, 0.03548544, -0.006372796,
-0.09998437, -0.01909431, -0.015580201, -0.07437171, 0.0389737, 0.028192239,
0.03236189, 0.030874284, -0.079905026, 0.019980008, 0.06815338, 0.0038312068,
0.036571063, 0.016687112, 0.014041338, -0.108033106, 0.111968175, -0.039645337,
```

-0.0020274178, -0.019974506, 0.008182213, -0.03420064, -0.018038016,
-0.0037441566, -0.07646354, 0.017165387, -0.07152119, 0.0017095942, 0.038785342,
0.089394495, -0.059645858, 0.11927814, -0.049067263, -0.03915105, -0.058131985,
0.009340398, 0.0154660335, -0.04823028, -0.0033952994, -0.023551567,
0.025857395, 0.0163184, -0.06353885, -0.019750375, 0.04094771, -0.022922484,
-0.07370045, 0.041088667, -0.02456765, -0.0246527, 0.041629855, -0.108763896,
-0.028739555, 0.020223148, -0.023667632, -0.08405779, 0.087284274, -0.07858633,
0.07024396, 0.08508288, 0.04218855, 0.07737685, -0.062583424, -0.14200279,
-0.119793504, 0.10644681, 0.04805418, 0.0037183918, 0.048693046, 0.07865343,
0.022508966, 0.06251688, 0.0025245792, 0.012562379, 0.0017826518, -0.025722316,
0.1208441, -0.0069828345, 0.059383832, 0.05765421, 0.026532533, -0.06360124,
-0.07541844, -0.033930544, -0.050897818, -0.00025522514, 0.021186203,
0.047301747, 0.008874112, …]
897248 [0.047544874, 0.047309197, -0.013163047, 0.0661334,
-0.08214547, 0.0037330778, 0.06863653, -0.11218322, 0.0741428, 0.101349555,
-0.03356669, -0.13560198, -0.0030550815, 0.01579904, -0.10682629, 0.04618503,
0.02733529, 0.09277102, 0.0033932677, -0.07859213, 0.0330945, 0.06269217,
0.011297132, 0.0305907, 0.020050956, -0.019888321, -0.07383109, 0.09879099,
-0.0022854002, -0.030739058, -0.02898777, 0.013334897, -0.008174783,
-0.02697382, 0.010568043, -0.058901947, 0.05490082, -0.020846829, 0.0020328937,
0.04096422, 0.065138415, -0.013813963, 0.13123374, -0.05828578, -0.03961544,
-0.052979764, -0.06649569, 0.0061758817, 0.020718716, 0.023212094, -0.010775122,
0.0059605776, -0.0058151092, -0.073311, 0.02533382, 0.0019328806, 0.011032558,
-0.039032284, 0.020784643, -0.030946108, -0.020223146, 0.05824072, -0.089479804,
-0.07050962, 0.031081397, 0.0035219097, -0.07448729, 0.03456992, -0.027764803,
0.08381638, 0.09473231, 0.016339444, 0.09176651, -0.004735871, -0.15488608,
-0.06612789, 0.05411318, 0.076462545, 0.04037649, 0.044587087, 0.033960212,
-0.06473194, 0.06388515, 0.0022090496, -0.02583026, -0.008607675, -0.06205538,
0.11983233, 0.030476259, 0.036770057, -0.013356879, 0.056070007, -0.027783083,
-0.059916582, -0.016190028, -0.085060954, 0.04067505, -0.019403892, 0.010204277,
-0.0048593953, …]
1829174 [0.03422878, 0.04633298, -0.04310807, 0.10887544, -0.046490215,
-0.05856456, 0.11932108, -0.071522586, 0.08140763, 0.08678669, -0.022142757,
-0.102395765, -0.018349025, -0.07013736, -0.106255576, 0.07125821, 0.104030274,
0.07175612, -0.027008057, 0.006426604, 0.07456164, 0.05057028, 0.033926923,
0.01852815, 0.04986307, -0.049271293, -0.007090693, 0.031010794, 0.035984207,
-0.08862305, -0.059522547, 0.027126146, 0.030540135, 0.054432746, 0.060333252,
-0.074792616, 0.08081934, -0.032170836, -0.055567864, 0.11552098, 0.09398352,
-0.026128354, 0.13503133, -0.013849673, -0.018830672, -0.10705699, -0.021697288,
0.02913898, -0.0028832478, -0.007446289, -0.029341988, 0.034446385,
-0.0027678118, -0.10300877, 0.0066929073, 0.028596962, -0.023819799,
-0.09535085, 0.06485848, -0.045650315, -0.04037874, 0.06760705, -0.10111137,
-0.06966765, 0.03132762, -0.025241852, -0.08836431, 0.030182237, -0.093307495,
0.08400826, 0.13591203, 0.06983301, -0.017822266, -0.034516044, -0.16754416,
-0.002105713, 0.041909922, 0.024342412, 0.08663011, 0.10384269, -0.020672342,
-0.024886422, 0.031103466, 0.032608695, -0.017795729, -0.044358626, -0.07607245,
0.102732785, 0.06948919, 0.046396006, -0.0020619268, 0.0700259, -0.092322305,

```
-0.048500393, -0.047732145, -0.050797172, 0.011920432, 0.10182124, 0.016617484,
-0.03802092, …]
976671   [0.058544263, 0.039916992, -0.059540413, 0.07721525, -0.068433605,
0.050970953, 0.13428105, -0.07326858, 0.055212896, 0.10193717, 0.005168193,
-0.13075875, -0.033471186, 0.014821233, -0.09955494, 0.084436364, 0.08692396,
0.11412914, 0.020678753, -0.047908474, 0.08908494, 0.04359972, -0.012765833,
-0.014662871, 4.4539167e-05, 0.024010323, -0.038609684, 0.047254406,
-0.003858824, 0.0011332744, -0.056227814, 0.029233571, -0.0024628511,
0.01180123, 0.02041894, -0.063756995, 0.090112634, -0.009497565, 0.013491244,
0.04006556, 0.023379764, 0.026136244, 0.19521682, -0.0021465404, -0.06180057,
-0.034680136, -0.04625475, -0.021673873, 0.048503153, 0.011954643, -0.073154345,
0.008894637, -0.006946873, -0.09184286, 0.044012893, -0.042948954, 0.034969434,
-0.046889022, 0.068829924, -0.050095018, -0.07475157, 0.043615393, -0.069884844,
-0.052924905, -0.02163449, 0.021533864, -0.04703666, 0.0035375648, -0.019788587,
0.029748762, 0.09740056, 0.02656184, 0.14094007, -0.040804476, -0.1884778,
-0.01383766, 0.034770243, 0.08736358, 0.068874046, -0.0015347454, -0.0054448103,
-0.078397594, 0.016789617, 0.011357488, 0.002203864, -0.053705577, -0.083432175,
0.074712396, 0.03936922, 0.07261864, 0.02286014, 0.104544975, -0.023737727,
-0.07776539, -0.021282298, -0.07619909, 0.024304364, 0.049033914, -0.024582941,
0.03415082, …]
```

[88]:
```
x_data = x_data.drop(columns = ['review_length','google_word2vec'])
```

[94]:
```
x_data.head()
```

[94]:
```
         customer_id  product_id  helpful_votes  total_votes  vine  \
1138359         9320       45936            2.0          2.0     0
854689         49498      106816            0.0          0.0     0
897248        413739      124090            0.0          0.0     1
1829174        44843       54237            0.0          0.0     0
976671         25113       61644            0.0          0.0     0

         verified_purchase  review_date
1138359                  1           49
854689                   0           30
897248                   0           28
1829174                  1            7
976671                   1           21
```

[99]:
```
df_temp
```

[99]:
```
          wv_0      wv_1      wv_2      wv_3      wv_4      wv_5      wv_6  \
0    -0.001935  0.033991 -0.006532  0.007509 -0.061800 -0.063817  0.044389
1     0.023616  0.021366 -0.014351  0.087931 -0.094090  0.012963  0.034359
2     0.047545  0.047309 -0.013163  0.066133 -0.082145  0.003733  0.068637
3     0.034229  0.046333 -0.043108  0.108875 -0.046490 -0.058565  0.119321
4     0.058544  0.039917 -0.059540  0.077215 -0.068434  0.050971  0.134281
```

```
…            …         …         …         …         …         …         …
524938   0.030233  0.010781  0.008298  0.101231  0.016175  0.029355  0.086017
524939   0.044684  0.017321 -0.002653  0.039392 -0.094506  0.072984  0.078428
524940   0.022168  0.012010 -0.032303  0.072634 -0.092907  0.037132  0.008967
524941   0.021636  0.010144  0.025300  0.060762 -0.048598  0.005075  0.124751
524942  -0.032589  0.081841 -0.008103  0.070638 -0.032776  0.132317  0.065097

            wv_7      wv_8      wv_9  …    wv_290    wv_291    wv_292  \
0      -0.102081  0.111635  0.125271  … -0.002348  0.118568 -0.120003
1      -0.055223  0.103590  0.035485  … -0.062802  0.096626 -0.095528
2      -0.112183  0.074143  0.101350  … -0.058634  0.032414 -0.076663
3      -0.071523  0.081408  0.086787  … -0.069612  0.091083 -0.077241
4      -0.073269  0.055213  0.101937  … -0.035501 -0.005313 -0.082816
…            …         …         …  …  …         …         …
524938 -0.011671  0.091160  0.030610  … -0.076649  0.045086 -0.040504
524939 -0.065056  0.085207  0.093854  … -0.103059  0.029510 -0.076460
524940 -0.005439  0.031051  0.018739  … -0.035898  0.059629 -0.078002
524941  0.000735  0.089655  0.166371  … -0.019741  0.111197 -0.044805
524942 -0.108927  0.050151  0.057962  … -0.049353  0.012610 -0.093004

          wv_293    wv_294    wv_295    wv_296    wv_297    wv_298    wv_299
0       0.030765 -0.056905 -0.055947 -0.000999 -0.003524 -0.004047 -0.067437
1       0.002015 -0.018051  0.017691 -0.070210 -0.038866 -0.024481 -0.014990
2       0.025040 -0.013628 -0.070761  0.022408 -0.033992 -0.006797 -0.017296
3      -0.004081 -0.039301 -0.060527  0.065801 -0.031075 -0.019414 -0.031172
4       0.065549 -0.041247 -0.031801  0.009999 -0.028617  0.021594 -0.006137
…            …         …         …         …         …         …         …
524938  0.029914 -0.015632 -0.001988  0.071113 -0.014698  0.020694 -0.041840
524939  0.043552 -0.040091 -0.015740  0.029869  0.012749 -0.043952 -0.024929
524940  0.045342 -0.004021 -0.004525 -0.023154 -0.032958 -0.013741 -0.038500
524941  0.088955 -0.056041  0.042162 -0.013039 -0.053318 -0.005345 -0.079354
524942  0.103728  0.043366 -0.027395 -0.036107  0.038730  0.027465 -0.025499

[524943 rows x 300 columns]
```

[124]:
```
#df_temp1 = df_temp.copy()
df_temp6=df_temp1.copy()
df_temp7= df_temp1.copy()
```

[107]:
```
df_temp1
```

[107]:
```
            wv_0      wv_1      wv_2      wv_3      wv_4      wv_5      wv_6  \
0      -0.001935  0.033991 -0.006532  0.007509 -0.061800 -0.063817  0.044389
1       0.023616  0.021366 -0.014351  0.087931 -0.094090  0.012963  0.034359
2       0.047545  0.047309 -0.013163  0.066133 -0.082145  0.003733  0.068637
3       0.034229  0.046333 -0.043108  0.108875 -0.046490 -0.058565  0.119321
4       0.058544  0.039917 -0.059540  0.077215 -0.068434  0.050971  0.134281
```

```
…            …         …         …         …         …         …         …
524938   0.030233  0.010781  0.008298  0.101231  0.016175  0.029355  0.086017
524939   0.044684  0.017321 -0.002653  0.039392 -0.094506  0.072984  0.078428
524940   0.022168  0.012010 -0.032303  0.072634 -0.092907  0.037132  0.008967
524941   0.021636  0.010144  0.025300  0.060762 -0.048598  0.005075  0.124751
524942  -0.032589  0.081841 -0.008103  0.070638 -0.032776  0.132317  0.065097

            wv_7      wv_8      wv_9  …    wv_290    wv_291    wv_292  \
0      -0.102081  0.111635  0.125271  … -0.002348  0.118568 -0.120003
1      -0.055223  0.103590  0.035485  … -0.062802  0.096626 -0.095528
2      -0.112183  0.074143  0.101350  … -0.058634  0.032414 -0.076663
3      -0.071523  0.081408  0.086787  … -0.069612  0.091083 -0.077241
4      -0.073269  0.055213  0.101937  … -0.035501 -0.005313 -0.082816
…            …         …         …  …        …         …         …
524938 -0.011671  0.091160  0.030610  … -0.076649  0.045086 -0.040504
524939 -0.065056  0.085207  0.093854  … -0.103059  0.029510 -0.076460
524940 -0.005439  0.031051  0.018739  … -0.035898  0.059629 -0.078002
524941  0.000735  0.089655  0.166371  … -0.019741  0.111197 -0.044805
524942 -0.108927  0.050151  0.057962  … -0.049353  0.012610 -0.093004

          wv_293    wv_294    wv_295    wv_296    wv_297    wv_298    wv_299
0       0.030765 -0.056905 -0.055947 -0.000999 -0.003524 -0.004047 -0.067437
1       0.002015 -0.018051  0.017691 -0.070210 -0.038866 -0.024481 -0.014990
2       0.025040 -0.013628 -0.070761  0.022408 -0.033992 -0.006797 -0.017296
3      -0.004081 -0.039301 -0.060527  0.065801 -0.031075 -0.019414 -0.031172
4       0.065549 -0.041247 -0.031801  0.009999 -0.028617  0.021594 -0.006137
…            …         …         …         …         …         …         …
524938  0.029914 -0.015632 -0.001988  0.071113 -0.014698  0.020694 -0.041840
524939  0.043552 -0.040091 -0.015740  0.029869  0.012749 -0.043952 -0.024929
524940  0.045342 -0.004021 -0.004525 -0.023154 -0.032958 -0.013741 -0.038500
524941  0.088955 -0.056041  0.042162 -0.013039 -0.053318 -0.005345 -0.079354
524942  0.103728  0.043366 -0.027395 -0.036107  0.038730  0.027465 -0.025499

[524943 rows x 300 columns]
```

[120]:
```
x_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 524943 entries, 1138359 to 1908791
Data columns (total 7 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   customer_id        524943 non-null  int32
 1   product_id         524943 non-null  int32
 2   helpful_votes      524943 non-null  float64
 3   total_votes        524943 non-null  float64
 4   vine               524943 non-null  int8
 5   verified_purchase  524943 non-null  int8
```

```
 6   review_date        524943 non-null  UInt32
dtypes: UInt32(1), float64(2), int32(2), int8(2)
memory usage: 39.5 MB
```

[129]:
```python
x_data = x_data.reset_index()
```

[131]:
```python
x_data_final = pd.concat([x_data, df_temp6], axis=1)
```

[133]:
```python
x_data_final=x_data_final.drop(columns = ['index'])
```

[146]:
```python
x_data_final.head(10)
```

[146]:
```
   customer_id  product_id  helpful_votes  total_votes  vine  \
0         9320       45936            2.0          2.0     0
1        49498      106816            0.0          0.0     0
2       413739      124090            0.0          0.0     1
3        44843       54237            0.0          0.0     0
4        25113       61644            0.0          0.0     0
5       331123       37864            0.0          0.0     0
6       168713      133820            0.0          0.0     0
7       162100       15552            2.0          2.0     0
8       314754       86409            0.0          0.0     0
9       418636        4632            6.0          7.0     0

   verified_purchase  review_date      wv_0      wv_1      wv_2  …  \
0                  1           49 -0.001935  0.033991 -0.006532  …
1                  0           30  0.023616  0.021366 -0.014351  …
2                  0           28  0.047545  0.047309 -0.013163  …
3                  1            7  0.034229  0.046333 -0.043108  …
4                  1           21  0.058544  0.039917 -0.059540  …
5                  1           11  0.001805  0.025168 -0.035034  …
6                  1           15  0.039904  0.103477 -0.023549  …
7                  1           43 -0.015724  0.102844  0.010731  …
8                  1           25  0.064803  0.120783  0.019293  …
9                  1           15  0.050998  0.066331 -0.046914  …

     wv_290    wv_291    wv_292    wv_293    wv_294    wv_295    wv_296  \
0 -0.002348  0.118568 -0.120003  0.030765 -0.056905 -0.055947 -0.000999
1 -0.062802  0.096626 -0.095528  0.002015 -0.018051  0.017691 -0.070210
2 -0.058634  0.032414 -0.076663  0.025040 -0.013628 -0.070761  0.022408
3 -0.069612  0.091083 -0.077241 -0.004081 -0.039301 -0.060527  0.065801
4 -0.035501 -0.005313 -0.082816  0.065549 -0.041247 -0.031801  0.009999
5 -0.095928  0.019469 -0.080421  0.016364  0.031195  0.046664  0.048239
6 -0.013316 -0.002373 -0.081317  0.059610 -0.043512 -0.012591 -0.037657
7 -0.039864  0.061187 -0.087471 -0.036678 -0.050098 -0.043412  0.057194
8 -0.061729  0.064918 -0.096920  0.044980 -0.058044 -0.060927  0.033427
9 -0.082608  0.064973 -0.036831  0.022224 -0.025337  0.018826  0.033573
```

```
     wv_297    wv_298    wv_299
0 -0.003524 -0.004047 -0.067437
1 -0.038866 -0.024481 -0.014990
2 -0.033992 -0.006797 -0.017296
3 -0.031075 -0.019414 -0.031172
4 -0.028617  0.021594 -0.006137
5 -0.066596  0.049888 -0.063947
6 -0.050718  0.009219 -0.054971
7 -0.016762 -0.007186 -0.049383
8 -0.087469  0.021839  0.021340
9  0.004684 -0.003388 -0.059410

[10 rows x 307 columns]
```

[147]:
```
x_data_final.columns
```

[147]:
```
Index(['customer_id', 'product_id', 'helpful_votes', 'total_votes', 'vine',
       'verified_purchase', 'review_date', 'wv_0', 'wv_1', 'wv_2',
       …
       'wv_290', 'wv_291', 'wv_292', 'wv_293', 'wv_294', 'wv_295', 'wv_296',
       'wv_297', 'wv_298', 'wv_299'],
      dtype='object', length=307)
```

[135]:
```
np.save('x_data_final.npy',x_data_final)
```

[145]:
```
np.save('x_data_final2.npy',x_data_final, allow_pickle = True)
```

[142]:
```
y_data.reset_index()
```

[142]:
```
          index  star_rating
0       1138359            1
1        854689            1
2        897248            1
3       1829174            1
4        976671            1
…           …          …
524938  1908760            0
524939  1908764            0
524940  1908773            0
524941  1908775            0
524942  1908791            0

[524943 rows x 2 columns]
```

[143]:
```
y_data = y_data.drop(columns = ['index'])
```

[144]:
```
np.save('y_data_final.npy',y_data)
```

[136]:
```
np.load()
```

[ ]:
```
```

In [1]:
```
import numpy as np
import pandas as pd
import random
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore', category=PendingDeprecationWarning)

from tqdm import tqdm
from sklearn.linear_model import Perceptron, LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split, StratifiedKFold, K
Fold, GridSearchCV
from sklearn.utils import resample
from sklearn.metrics import accuracy_score, classification_report, roc_a
uc_score, confusion_matrix
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.base import clone
from sklearn.utils.validation import check_array
from sklearn.mixture import GaussianMixture
from sklearn.semi_supervised import LabelSpreading
from sklearn.feature_extraction.text import TfidfVectorizer

from scipy.spatial import distance
from scipy.cluster.vq import kmeans2
from scipy.stats import multivariate_normal
from qns3vm import QN_S3VM

# import gzip
# import nltk
# import requests
# from io import BytesIO
# import imblearn
# from nltk.corpus import stopwords
# from nltk.stem import WordNetLemmatizer
# nltk.download('stopwords')
# nltk.download('wordnet')
# import re
# import contractions
# from sklearn.preprocessing import LabelEncoder
# from bs4 import BeautifulSoup
# pd.set_option('display.max_colwidth', None)
# import requests
# import gensim.downloader as api
# from gensim.models import Word2Vec
# wv = api.load('word2vec-google-news-300')
```

In [2]:
```
X_review_h = np.load('X_home_review_wv.npy')
y_h = np.load('y_data_home_finalized.npy')
x_h = np.load('X_data_home_finalized.npy', allow_pickle=True)
```

In [3]:
```
X_review_k = np.load('X_Kitch_review_wv.npy', allow_pickle = True)
y_k = np.load('y_data_Kitch_finalized.npy' ,allow_pickle = True)
x_k = np.load('X_data_Kitch_finalized.npy', allow_pickle = True)
```

In [4]:
```
X_review_h.shape
```

Out[4]:
```
(586121, 300)
```

In [5]:
```
X_review_k.shape
```

Out[5]:
```
(293126, 300)
```

In [6]:
```
# Train split for Supervised Models
```

In [7]:
```
X_train_h, X_test_h, Y_train_h, Y_test_h = train_test_split(X_review_h,
y_h, test_size=0.2, random_state=200)
```

In [8]:
```
X_train_k, X_test_k, Y_train_k, Y_test_k = train_test_split(X_review_k,
y_k, test_size=0.2, random_state=200)
```

In [10]:
```
# Reduced data for TL & SSL & USL
```

In [78]:
```
X_train_h_r, X_test_h_r, Y_train_h_r, Y_test_h_r = train_test_split(X_re
view_h, y_h, train_size = 3000,
                                                                    test_
size=600,random_state=200, stratify =y_h)
```

In [12]:
```
X_train_k_r, X_test_k_r, Y_train_k_r, Y_test_k_r = train_test_split(X_re
view_k, y_k, train_size = 400,
                                                                    test_
size=200,random_state=200, stratify =y_k)
```

Perceptron Model

In [12]:
```
clfPercep = Perceptron()
parameters = {'penalty':['l2','l1'] ,'alpha': [0.0001, 0.0003, 0.001, 0.
003, 0.01, 0.03, 0.1, 0.3]}
cv = StratifiedKFold(n_splits=5, shuffle = True, random_state =13)
gridsearch = GridSearchCV(clfPercep, parameters, cv=cv, scoring='accurac
y')
# Find the best params with grid search
gridsearch.fit(X_train_h, Y_train_h)
print("Best params: {}".format(gridsearch.best_params_))
print("Best f1 score: %.5f" % gridsearch.best_score_)
```

```
Best params: {'alpha': 0.0001, 'penalty': 'l2'}
Best f1 score: 0.72755
```

In [13]:
```
print("Perceptron model report on training and testing data: \n")
Y_train_pred = gridsearch.predict(X_train_h)
training_report = classification_report(Y_train_h, Y_train_pred, output_
dict=False)
Y_test_pred = gridsearch.predict(X_test_h)
testing_report = classification_report(Y_test_h, Y_test_pred, output_dic
t=False)
print("The training report is:\n", training_report)
print("The testing report is:\n", testing_report)
```

```
Perceptron model report on training and testing data:

The training report is:
               precision    recall  f1-score   support

           0       0.77      0.80      0.78    234456
           1       0.79      0.76      0.77    234440

    accuracy                           0.78    468896
   macro avg       0.78      0.78      0.78    468896
weighted avg       0.78      0.78      0.78    468896

The testing report is:
               precision    recall  f1-score   support

           0       0.76      0.80      0.78     58604
           1       0.79      0.75      0.77     58621

    accuracy                           0.78    117225
   macro avg       0.78      0.78      0.78    117225
weighted avg       0.78      0.78      0.78    117225
```

Decision Tree Model

In [14]:
```
clfTree = DecisionTreeClassifier(criterion = 'entropy')
parameters = {'max_depth': [2, 3, 4, 6]}
cv = StratifiedKFold(n_splits=5, shuffle = True, random_state =13)
gridsearch = GridSearchCV(clfTree, parameters, cv=cv, scoring='accuracy'
)
# Find the best params with grid search
gridsearch.fit(X_train_h, Y_train_h)
print("Best params: {}".format(gridsearch.best_params_))
print("Best f1 score: %.5f" % gridsearch.best_score_)
```

```
Best params: {'max_depth': 6}
Best f1 score: 0.67798
```

In [15]:
```
print("Decision Tree model report on training and testing data: \n")
training_report = classification_report(Y_train_h, gridsearch.predict(X_train_h), output_dict=False)
testing_report = classification_report(Y_test_h, gridsearch.predict(X_test_h), output_dict=False)
print("The training report is:\n", training_report)
print("The testing report is:\n", testing_report)
```

```
Decision Tree model report on training and testing data:

The training report is:
               precision    recall  f1-score   support

           0       0.67      0.71      0.69    234456
           1       0.69      0.65      0.67    234440

    accuracy                           0.68    468896
   macro avg       0.68      0.68      0.68    468896
weighted avg       0.68      0.68      0.68    468896

The testing report is:
               precision    recall  f1-score   support

           0       0.67      0.71      0.69     58604
           1       0.69      0.64      0.67     58621

    accuracy                           0.68    117225
   macro avg       0.68      0.68      0.68    117225
weighted avg       0.68      0.68      0.68    117225
```

Adaboost

In [16]:
```
clfAda = AdaBoostClassifier()
parameters = { 'n_estimators':[10,50,80], 'learning_rate':[0.001,0.01,0.1]}
cv = StratifiedKFold(n_splits=5, shuffle = True, random_state =13)
gridsearch = GridSearchCV(clfAda, parameters, cv=cv, scoring='accuracy')
# Find the best params with grid search
gridsearch.fit(X_train_h, Y_train_h)
print("Best params: {}".format(gridsearch.best_params_))
print("Best f1 score: %.5f" % gridsearch.best_score_)
```

```
Best params: {'learning_rate': 0.1, 'n_estimators': 80}
Best f1 score: 0.72466
```

In [17]:
```
print("Adaboost model report on training and testing data: \n")
training_report = classification_report(Y_train_h, gridsearch.predict(X_train_h), output_dict=False)
testing_report = classification_report(Y_test_h, gridsearch.predict(X_test_h), output_dict=False)
print("The training report is:\n", training_report)
print("The testing report is:\n", testing_report)
```

```
Adaboost model report on training and testing data:

The training report is:
               precision    recall  f1-score   support

           0       0.72      0.74      0.73    234456
           1       0.73      0.71      0.72    234440

    accuracy                           0.73    468896
   macro avg       0.73      0.73      0.73    468896
weighted avg       0.73      0.73      0.73    468896

The testing report is:
               precision    recall  f1-score   support

           0       0.72      0.74      0.73     58604
           1       0.73      0.71      0.72     58621

    accuracy                           0.72    117225
   macro avg       0.72      0.72      0.72    117225
weighted avg       0.72      0.72      0.72    117225
```

SVM

In [18]:
```
clfSVM = LinearSVC()
parameters = {'penalty':['l1', 'l2']}
cv = StratifiedKFold(n_splits=5, shuffle = True, random_state =13)
gridsearch = GridSearchCV(clfSVM, parameters, cv=cv, scoring='accuracy')
# Find the best params with grid search
gridsearch.fit(X_train_h, Y_train_h)
print("Best params: {}".format(gridsearch.best_params_))
print("Best f1 score: %.5f" % gridsearch.best_score_)
```

```
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_classes.py", line 234, in fit
    self.coef_, self.intercept_, self.n_iter_ = _fit_liblinear(
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 974, in _fit_liblinear
    solver_type = _get_liblinear_solver_type(multi_class, penalty, los
s, dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 830, in _get_liblinear_solver_type
    raise ValueError('Unsupported set of arguments: %s, '
ValueError: Unsupported set of arguments: The combination of penalty='l
1' and loss='squared_hinge' are not supported when dual=True, Parameter
s: penalty='l1', loss='squared_hinge', dual=True

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_classes.py", line 234, in fit
    self.coef_, self.intercept_, self.n_iter_ = _fit_liblinear(
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 974, in _fit_liblinear
    solver_type = _get_liblinear_solver_type(multi_class, penalty, los
s, dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 830, in _get_liblinear_solver_type
    raise ValueError('Unsupported set of arguments: %s, '
ValueError: Unsupported set of arguments: The combination of penalty='l
1' and loss='squared_hinge' are not supported when dual=True, Parameter
s: penalty='l1', loss='squared_hinge', dual=True

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
```

```
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_classes.py", line 234, in fit
    self.coef_, self.intercept_, self.n_iter_ = _fit_liblinear(
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 974, in _fit_liblinear
    solver_type = _get_liblinear_solver_type(multi_class, penalty, los
s, dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 830, in _get_liblinear_solver_type
    raise ValueError('Unsupported set of arguments: %s, '
ValueError: Unsupported set of arguments: The combination of penalty='l
1' and loss='squared_hinge' are not supported when dual=True, Parameter
s: penalty='l1', loss='squared_hinge', dual=True

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_classes.py", line 234, in fit
    self.coef_, self.intercept_, self.n_iter_ = _fit_liblinear(
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 974, in _fit_liblinear
    solver_type = _get_liblinear_solver_type(multi_class, penalty, los
s, dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 830, in _get_liblinear_solver_type
    raise ValueError('Unsupported set of arguments: %s, '
ValueError: Unsupported set of arguments: The combination of penalty='l
1' and loss='squared_hinge' are not supported when dual=True, Parameter
s: penalty='l1', loss='squared_hinge', dual=True

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_classes.py", line 234, in fit
    self.coef_, self.intercept_, self.n_iter_ = _fit_liblinear(
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 974, in _fit_liblinear
    solver_type = _get_liblinear_solver_type(multi_class, penalty, los
s, dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/svm/_base.py", line 830, in _get_liblinear_solver_type
```

```
    raise ValueError('Unsupported set of arguments: %s, '
ValueError: Unsupported set of arguments: The combination of penalty='l
1' and loss='squared_hinge' are not supported when dual=True, Parameter
s: penalty='l1', loss='squared_hinge', dual=True

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_search.py:922: UserWarning: One or more of the tes
t scores are non-finite: [       nan 0.80147197]
  warnings.warn(

Best params: {'penalty': 'l2'}
Best f1 score: 0.80147
```

In [19]:

```
print("SVM model report on training and testing data: \n")
training_report = classification_report(Y_train_h, gridsearch.predict(X_
train_h), output_dict=False)
testing_report = classification_report(Y_test_h, gridsearch.predict(X_te
st_h), output_dict=False)
print("The training report is:\n", training_report)
print("The testing report is:\n", testing_report)
```

```
SVM model report on training and testing data:

The training report is:
               precision    recall  f1-score   support

           0       0.80      0.81      0.80    234456
           1       0.81      0.79      0.80    234440

    accuracy                           0.80    468896
   macro avg       0.80      0.80      0.80    468896
weighted avg       0.80      0.80      0.80    468896

The testing report is:
               precision    recall  f1-score   support

           0       0.79      0.81      0.80     58604
           1       0.81      0.79      0.80     58621

    accuracy                           0.80    117225
   macro avg       0.80      0.80      0.80    117225
weighted avg       0.80      0.80      0.80    117225
```

Logistic Regression

In [20]:
```
clf = LogisticRegression()
parameters = {'penalty':['l2','l1'] }
cv = StratifiedKFold(n_splits=5, shuffle = True, random_state =13)
gridsearch = GridSearchCV(clf, parameters, cv=cv, scoring='accuracy')
gridsearch.fit(X_train_h, Y_train_h)
print("Best params: {}".format(gridsearch.best_params_))
print("Best f1 score: %.5f" % gridsearch.best_score_)
```

```
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

```
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 1306, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 443, in _check_solver
    raise ValueError("Solver %s supports only 'l2' or 'none' penalties,
"
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1
penalty.

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 1306, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 443, in _check_solver
    raise ValueError("Solver %s supports only 'l2' or 'none' penalties,
"
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1
penalty.

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 1306, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 443, in _check_solver
```

```
    raise ValueError("Solver %s supports only 'l2' or 'none' penalties,
"
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1
penalty.

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 1306, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 443, in _check_solver
    raise ValueError("Solver %s supports only 'l2' or 'none' penalties,
"
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1
penalty.

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_validation.py:615: FitFailedWarning: Estimator fit
failed. The score on this train-test partition for these parameters wil
l be set to nan. Details:
Traceback (most recent call last):
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/model_selection/_validation.py", line 598, in _fit_and_scor
e
    estimator.fit(X_train, y_train, **fit_params)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 1306, in fit
    solver = _check_solver(self.solver, self.penalty, self.dual)
  File "/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packa
ges/sklearn/linear_model/_logistic.py", line 443, in _check_solver
    raise ValueError("Solver %s supports only 'l2' or 'none' penalties,
"
ValueError: Solver lbfgs supports only 'l2' or 'none' penalties, got l1
penalty.

  warnings.warn("Estimator fit failed. The score on this train-test"
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/model_selection/_search.py:922: UserWarning: One or more of the tes
t scores are non-finite: [0.80150822        nan]
  warnings.warn(

Best params: {'penalty': 'l2'}
Best f1 score: 0.80151
```

```
/Users/khalid/opt/anaconda3/envs/EE660/lib/python3.8/site-packages/skle
arn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to
converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown
in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

In [22]:
```
print("Logistic model report on training and testing data: \n")
Y_train_pred = gridsearch.predict(X_train_h)
training_report = classification_report(Y_train_h, Y_train_pred, output_
dict=False)
Y_test_pred = gridsearch.predict(X_test_h)
testing_report = classification_report(Y_test_h, Y_test_pred, output_dic
t=False)
print("The training report is:\n", training_report)
print("The testing report is:\n", testing_report)
```

```
Logistic model report on training and testing data:

The training report is:
               precision    recall  f1-score   support

           0       0.80      0.81      0.80    234456
           1       0.81      0.79      0.80    234440

    accuracy                           0.80    468896
   macro avg       0.80      0.80      0.80    468896
weighted avg       0.80      0.80      0.80    468896

The testing report is:
               precision    recall  f1-score   support

           0       0.80      0.81      0.80     58604
           1       0.81      0.79      0.80     58621

    accuracy                           0.80    117225
   macro avg       0.80      0.80      0.80    117225
weighted avg       0.80      0.80      0.80    117225
```

Baseline Models

In [22]:

```
def base_model_1(y_train, y_test):
    unique, counts = np.unique(y_train, return_counts=True)
    total = len(y_train)
    pi = []
    for value in counts:
        pi.append(value/total)
    Error_train = []
    Error_test = []
    for i in range(10):
        random.seed(i)
        y_predicted = []
        y_predicted_test = []
        for j in range(total):
            y_predicted.append(random.choices(unique, weights=pi))
        for k in range(len(y_test)):
            y_predicted_test.append(random.choices(unique, weights=pi))
        Error_train.append( 1 - accuracy_score(y_train, y_predicted) )
        Error_test.append( 1 - accuracy_score(y_test, y_predicted_test)
)
    mean_error_train = np.mean(Error_train)
    std_train = np.std(Error_train)
    print('mean percent classification error (training): %.3f'\
          %mean_error_train)
    print('std of (training): %.4f'  %std_train)
    mean_error_test = np.mean(Error_test)
    std_test = np.std(Error_test)
    print('mean percent classification error (test): %.3f'%mean_error_te
st)
    print('std of (test): %.4f' %std_test)
    return
```

In [23]:

```
def base_model_2(y_train, y_test):
    unique, counts = np.unique(y_train, return_counts=True)
    total = len(y_train)
    pi = []
    for value in counts:
        pi.append(value/total)
    idx = np.argmax(pi)
    y_predicted = []
    y_predicted_test = []
    for j in range(total):
        y_predicted.append( unique[idx] )
    Error_train = 1 - accuracy_score(y_train, y_predicted)
    print('percent classification error (training): %.3f'%Error_train)
    for k in range(len(y_test)):
        y_predicted_test.append( unique[idx] )
    Error_test = 1 - accuracy_score(y_test, y_predicted_test)
    print('percent classification error (test): %.3f'%Error_test)
    return
```

In [24]:
```
base_model_1(Y_train_h, Y_test_h)
```

```
mean percent classification error (training): 0.500
std of (training): 0.0008
mean percent classification error (test): 0.500
std of (test): 0.0015
```

In [25]:
```
base_model_2(Y_train_h, Y_test_h)
```

```
percent classification error (training): 0.500
percent classification error (test): 0.500
```

Transfer Learning

In [13]:
```
# supervised Learninig for the reduced data as a basline for TL Source
clfSVM = LinearSVC(penalty= 'l2')
clfSVM.fit(X_train_h_r, Y_train_h_r)

print("SVM model report on training and testing data: \n")
training_report = classification_report(Y_train_h_r, clfSVM.predict(X_tr
ain_h_r), output_dict=False)
testing_report = classification_report(Y_test_h_r, clfSVM.predict(X_test
_h_r), output_dict=False)
print("The training report is:\n", training_report)
print("The testing report is:\n", testing_report)
```

```
SVM model report on training and testing data:

The training report is:
               precision    recall  f1-score   support

           0       0.82      0.83      0.83      1500
           1       0.83      0.82      0.83      1500

    accuracy                           0.83      3000
   macro avg       0.83      0.83      0.83      3000
weighted avg       0.83      0.83      0.83      3000

The testing report is:
               precision    recall  f1-score   support

           0       0.78      0.77      0.77       300
           1       0.77      0.78      0.77       300

    accuracy                           0.77       600
   macro avg       0.77      0.77      0.77       600
weighted avg       0.77      0.77      0.77       600
```

In [14]:
```
# using source predict target
testing_report = classification_report(Y_test_k_r, clfSVM.predict(X_test
_k_r), output_dict=False)
print("The testing report is:\n", testing_report)
```

```
The testing report is:
               precision    recall  f1-score   support

           0       0.82      0.69      0.75       100
           1       0.73      0.85      0.79       100

    accuracy                           0.77       200
   macro avg       0.78      0.77      0.77       200
weighted avg       0.78      0.77      0.77       200
```

In [15]:
```
# supervised Learninig for the reduced data as a basline for TL Target
clfSVM = LinearSVC(penalty= 'l2')
clfSVM.fit(X_train_k_r, Y_train_k_r)

print("SVM model report on training and testing data: \n")
training_report = classification_report(Y_train_k_r, clfSVM.predict(X_tr
ain_k_r), output_dict=False)
testing_report = classification_report(Y_test_k_r, clfSVM.predict(X_test
_k_r), output_dict=False)
print("The training report is:\n", training_report)
print("The testing report is:\n", testing_report)
```

```
SVM model report on training and testing data:

The training report is:
               precision    recall  f1-score   support

           0       0.92      0.96      0.94       200
           1       0.96      0.92      0.94       200

    accuracy                           0.94       400
   macro avg       0.94      0.94      0.94       400
weighted avg       0.94      0.94      0.94       400

The testing report is:
               precision    recall  f1-score   support

           0       0.75      0.79      0.77       100
           1       0.78      0.73      0.75       100

    accuracy                           0.76       200
   macro avg       0.76      0.76      0.76       200
weighted avg       0.76      0.76      0.76       200
```

In [17]:
```
X_uninon = np.concatenate((X_train_h_r, X_train_k_r), axis=0)
y_uninon = np.concatenate((Y_train_h_r, Y_train_k_r), axis=0)

clf_unio = LinearSVC(penalty='l2')
clf_unio.fit(X_uninon, y_uninon)
print("SVM model report on training and testing data: \n")
training_report = classification_report(y_uninon, clf_unio.predict(X_uni
non), output_dict=False)
testing_report = classification_report(Y_test_k_r, clf_unio.predict(X_te
st_k_r), output_dict=False)
print("The training report is:\n", training_report)
print("The testing report is:\n", testing_report)
```

```
SVM model report on training and testing data:

The training report is:
               precision    recall  f1-score   support

           0       0.82      0.83      0.83      1700
           1       0.83      0.82      0.82      1700

    accuracy                           0.82      3400
   macro avg       0.82      0.82      0.82      3400
weighted avg       0.82      0.82      0.82      3400

The testing report is:
               precision    recall  f1-score   support

           0       0.80      0.75      0.77       100
           1       0.76      0.81      0.79       100

    accuracy                           0.78       200
   macro avg       0.78      0.78      0.78       200
weighted avg       0.78      0.78      0.78       200
```

TrAdaboost

In [28]:
```
class TrAdaBoost(object):
    def __init__(self,N=10,base_estimator=DecisionTreeClassifier(),score
=roc_auc_score):
        self.N=N
        self.base_estimator=base_estimator
        self.score=score
        self.beta_all = None
        self.estimators=[]

    def _calculate_weights(self,weights):
        weights = weights.ravel()
        total = np.sum(weights)
        print("Total weight is: ",total," min Weight is: ",np.min(weight
s)," max Weight is: ",np.max(weights))
        return np.asarray(weights / total, order='C')

    def _calculate_error_rate(self,y_true, y_pred, weight):
        weight = weight.ravel()
        total = np.sum(weight)
        print("Total weight is: ",total," min Weight is: ",np.min(weight
)," max Weight is: ",np.max(weight))
        return np.sum(weight / total * np.abs(y_true - y_pred))

    def fit(self,source,target,source_label,target_label):
        source_shape=source.shape[0]
        target_shape=target.shape[0]
        trans_data = np.concatenate((source, target), axis=0)
        trans_label = np.concatenate((source_label,target_label), axis=0
)
        weights_source = np.ones([source_shape, 1])/source_shape
        weights_target = np.ones([target_shape, 1])/target_shape
        weights = np.concatenate((weights_source, weights_target), axis=
0)

        bata = 1 / (1 + np.sqrt(2 * np.log(source_shape / self.N)))
        self.beta_all = np.zeros([1, self.N])
        result_label = np.ones([source_shape+target_shape, self.N])

        trans_data = np.asarray(trans_data, order='C')
        trans_label = np.asarray(trans_label, order='C')

        best_round = 0
        score=0
        flag=0

        for i in tqdm(range(self.N)):
            P = self._calculate_weights(weights)
            est = clone(self.base_estimator).fit(trans_data,trans_label,
sample_weight=P.ravel())
            self.estimators.append(est)
            y_preds=est.predict(trans_data)
            result_label[:, i]=y_preds

            y_target_pred=est.predict(target)
            error_rate = self._calculate_error_rate(target_label, y_targ
et_pred,  \
```

```
                                                weights[source_shape:sourc
e_shape + target_shape, :])

            if error_rate >= 0.5 or error_rate == 0:
                self.N = i
                print('early stop! due to error_rate=%.2f'%(error_rate))
                break

            self.beta_all[0, i] = error_rate / (1 - error_rate)

            for j in range(target_shape):
                weights[source_shape + j] = weights[source_shape + j] *
 \
                np.power(self.beta_all[0, i],(-np.abs(result_label[sourc
e_shape + j, i] - target_label[j])))

            for j in range(source_shape):
                weights[j] = weights[j] * np.power(bata,np.abs(result_la
bel[j, i] - source_label[j]))

            tp=self.score(target_label,y_target_pred)
            print('The '+str(i)+' rounds score is '+str(tp))

    def _predict_one(self, x):
        """
        Output the hypothesis for a single instance
        :param x: array-like
            target label of a single instance from each iteration in ord
er
        :return: 0 or 1
        """
        x, N = check_array(x, ensure_2d=False), self.N
        # replace 0 by 1 to avoid zero division and remove it from the p
roduct
        beta = [self.beta_all[0,t] if self.beta_all[0,t] != 0 else 1 for
t in range(int(np.ceil(N/2)), N)]
        cond = np.prod([b ** -x for b in beta]) >= np.prod([b ** -0.5 fo
r b in beta])
        return int(cond)

    def predict(self, x_test):
        y_pred_list = np.array([est.predict(x_test) for est in self.esti
mators]).T
        y_pred = np.array(list(map(self._predict_one, y_pred_list)))
        return y_pred
```

In [268]:
```
X_train_h_r, X_test_h_r, Y_train_h_r, Y_test_h_r = train_test_split(X_re
view_h, y_h, train_size = 3000,
                                                                    test_
size=600,random_state=200, stratify =y_h)

base_estimator = LinearSVC(penalty = 'l2')
clf = TrAdaBoost(N=4,base_estimator=base_estimator,score=accuracy_score)
clf.fit(X_train_h_r, X_train_k_r, Y_train_h_r, Y_train_k_r)
```

```
 75%|                                        | 3/4 [00:00<00:00, 1
4.97it/s]

Total weight is:  2.0  min Weight is:  0.0003333333333333333  max Weigh
t is:  0.0025
Total weight is:  0.9999999999999998  min Weight is:  0.0025  max Weigh
t is:  0.0025
The 0 rounds score is 0.7825
Total weight is:  2.3508526805907266  min Weight is:  7.185919486047216
e-05  max Weight is:  0.008994252873563216
Total weight is:  1.565  min Weight is:  0.0025  max Weight is:  0.0089
94252873563216
The 1 rounds score is 0.5
Total weight is:  2.4302746267282815  min Weight is:  1.549123165798592
7e-05  max Weight is:  0.014313940883574195
Total weight is:  1.922183908045977  min Weight is:  0.0025  max Weight
is:  0.014313940883574195
The 2 rounds score is 0.5
Total weight is:  2.3703953738914683  min Weight is:  3.339562303019777
7e-06  max Weight is:  0.014313940883574202
Total weight is:  1.9221839080459775  min Weight is:  0.0025  max Weigh
t is:  0.014313940883574202
early stop! due to error_rate=0.50
```

In [269]:
```
ys_pred = clf.predict(X_test_h_r)
print('test accuracy of source domain:',accuracy_score(Y_test_h_r, ys_pr
ed))

yt_pred = clf.predict(X_train_k_r)
print('train accuracy of target domain:',accuracy_score(Y_train_k_r, yt_
pred))

yt_test_pred = clf.predict(X_test_k_r)
print('The testing report of TL in target domain is:\n',classification_r
eport(Y_test_k_r, yt_test_pred, output_dict=False))
```

```
test accuracy of source domain: 0.7
train accuracy of target domain: 0.7825
The testing report of TL in target domain is:
               precision    recall  f1-score   support

           0       0.85      0.58      0.69       100
           1       0.68      0.90      0.78       100

    accuracy                           0.74       200
   macro avg       0.77      0.74      0.73       200
weighted avg       0.77      0.74      0.73       200
```

In [270]:
```
#baseline_target
baseline_target = LinearSVC(penalty = 'l2')
baseline_target.fit(X_train_k_r, Y_train_k_r)
yt_test_pred = baseline_target.predict(X_test_k_r)
print('The testing report of SL in target domain is\n',classification_re
port(Y_test_k_r, yt_test_pred, output_dict=False))
```

```
The testing report of SL in target domain is
               precision    recall  f1-score   support

           0       0.75      0.79      0.77       100
           1       0.78      0.73      0.75       100

    accuracy                           0.76       200
   macro avg       0.76      0.76      0.76       200
weighted avg       0.76      0.76      0.76       200
```

In [24]:
```
X_uninon = np.concatenate((x_train_h_r, x_train_k_r), axis=0)
y_uninon = np.concatenate((y_h, y_train_k_r), axis=0)

clf_unio = LinearSVC(penalty='l2')
clf_unio.fit(X_uninon, y_uninon)
y_pred_test_unio = clf_unio.predict(x_test_k_r)
acc_adab_t = accuracy_score(y_test_k_r, y_pred_test_unio)
print('The accuracy score of the SVC on test target data = ', acc_adab_t
)
```

```
The accuracy score of the SVC on test target data =  0.785
```

In [25]:
```
clf_unio = AdaBoostClassifier(learning_rate=0.1, n_estimators=80)
clf_unio.fit(X_uninon, y_uninon)
y_pred_test_unio = clf_unio.predict(x_test_k_r)
acc_adab_t = accuracy_score(y_test_k_r, y_pred_test_unio)
print('The accuracy score of the AdaBoostClassifier onthe test target da
ta = ', acc_adab_t)
```

```
The accuracy score of the AdaBoostClassifier onthe test target data =
0.76
```

Importantance weighting

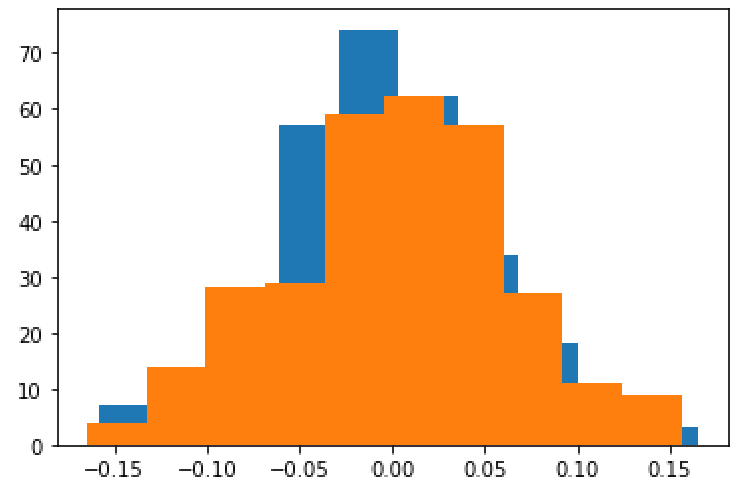
In [60]:
```
X_uninon = np.concatenate((X_train_h_r, X_train_k_r[:8,:]), axis=0)
y_uninon = np.concatenate((Y_train_h_r, Y_train_k_r[:8]), axis=0)


gm = GaussianMixture(n_components=2, random_state=0).fit(X_uninon)
means = gm.means_
cov_mat = gm.covariances_
print('The means = \n', means.shape)
print('The covariance matrices = \n', cov_mat.shape)
```

```
The means =
 (2, 300)
The covariance matrices =
 (2, 300, 300)
```

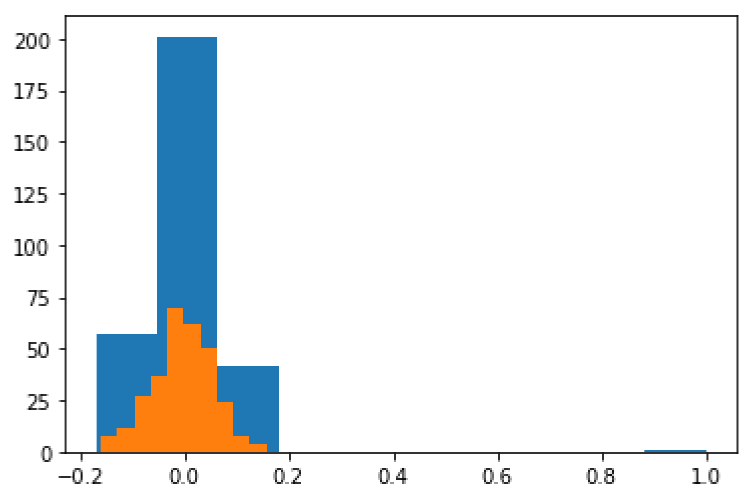
In [61]:
```
print('The means class zero 0 = \n', np.mean(means[0,:]-means[1,:]))
print('The means 1 = \n', np.mean(means[1,:]))
```

```
The means class zero 0 =
 -0.0001780652428123944
The means 1 =
 -0.003548618934282531
```

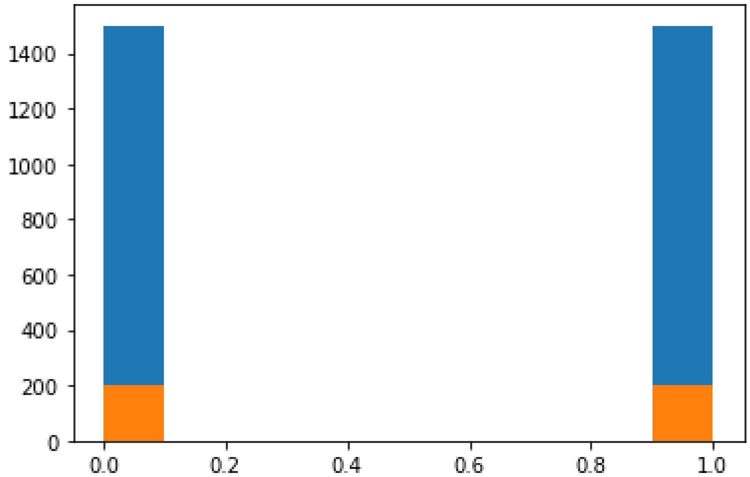
In [62]:
```
plt.hist(X_train_h_r[7,:])
plt.hist(X_train_k_r[7,:])
plt.show()
```



In [63]:
```
X_concept_h = np.concatenate((X_train_h_r, Y_train_h_r.reshape(-1,1)), a
xis=1)
plt.hist(X_concept_h[6,:])
X_concept_k = np.concatenate((X_train_k_r, Y_train_k_r.reshape(-1,1)), a
xis=1)
plt.hist(X_concept_k[6,:])
plt.show()
```



In [64]:
```
plt.hist(Y_train_h_r[:])
plt.hist(Y_train_k_r[:])
plt.show()
```



In [66]:
```
# warnings.filterwarnings("ignore")
# mean_s = means[0,:]
# cov_mat_s = cov_mat[0,:,:]

# mean_t = means[1,:]
# cov_mat_t = cov_mat[1,:,:]
# from sklearn.preprocessing import normalize

# X_uninon = normalize(X_uninon)

# w = multivariate_normal.pdf(X_uninon, mean=mean_t, cov=cov_mat_t) / mu
ltivariate_normal.pdf(X_uninon, mean=mean_s, cov=cov_mat_s)

# clf_unio = LinearSVC()
# clf_unio.fit(X_uninon, y_uninon, sample_weight = w)
# y_pred_test_unio = clf_unio.predict(X_test_h_r)
# acc_adab_t = accuracy_score(Y_test_h_r, y_pred_test_unio)
# print('The accuracy score of the source AdaBoostClassifier on target d
ata = ', acc_adab_t)
```

Semi-Suervised Learning

In [273]:
```
# trained Labeld and unlabeled
X_train_h_r, X_test_h_r, Y_train_h_r, Y_test_h_r = train_test_split(X_re
view_h, y_h, train_size = 1000,
                                                                    test_
size=300,random_state=200, stratify =y_h)

x_h_l = X_train_h_r[:8,:]
y_h_l = Y_train_h_r[:8]
y_h_labeled = np.where(y_h_l == 0 , -1, 1)

x_h_u = X_train_h_r[8:,:]
y_h_u = Y_train_h_r[8:]
y_h_un = np.where(y_h_u  == 0 , -1, 1)

# test Labeld
# X_test_h_r[]
y_h_test = np.where(Y_test_h_r  == 0 , -1, 1)
```

In [274]:
```
import warnings
warnings.filterwarnings('ignore', category=PendingDeprecationWarning)

# train and predict on labeled data
clf_3 = QN_S3VM(x_h_l.tolist(), y_h_labeled.tolist(), x_h_u.tolist(), la
m=1,
                kernel_type='Linear', random_generator=random.Random())
clf_3.train()
y_pred_test = clf_3.getPredictions(X_test_h_r)
acc_s3vm_l = classification_report(y_h_test, y_pred_test, output_dict=Fa
lse)
print('The accuracy score of QN_S3VM on test data home is:\n ', acc_s3vm
_l)
```

```
The accuracy score of QN_S3VM on test data home is:
                precision    recall  f1-score   support

          -1       0.60      0.66      0.63       150
           1       0.62      0.55      0.58       150

    accuracy                           0.61       300
   macro avg       0.61      0.61      0.61       300
weighted avg       0.61      0.61      0.61       300
```

In [92]:
```
# Another method LabelSpreading

x_uninon_train = np.concatenate((x_h_l,x_h_u), axis=0)
y_uninon_train = np.concatenate((y_h_l, y_h_u), axis=0)

label_prop_model = LabelSpreading()
label_prop_model.fit(x_uninon_train, y_uninon_train)
acc_score = label_prop_model.score(X_test_h_r, Y_test_h_r)
print(f'The accuracy score of semi_supervised by LabelSpreading on test
 data home is= {acc_score}')
```

```
The accuracy score of semi_supervised by LabelSpreading on test data ho
me is= 0.6833333333333333
```

In [93]:
```
acc_score = label_prop_model.score(x_uninon_train, y_uninon_train)
print(f'The accuracy score of semi_supervised by LabelSpreading on test
 data home is= {acc_score}')
```

```
The accuracy score of semi_supervised by LabelSpreading on test data ho
me is= 0.999
```

In [76]:
```
# Another method
L = np.arange(1,11,1)
acc_b = []
clf = LinearSVC()
for i in L:
    idx = 20*i
    clf.fit(X_train_h_r[:idx,:], Y_train_h_r[:idx])
    y_pred_test = clf.predict(X_test_h_r)
    acc_svm_l = accuracy_score(y_h_test, y_pred_test)
    acc_b.append(acc_svm_l)
    print(f'The accuracy score of SVM of {idx} samples = {acc_svm_l}')
```

```
The accuracy score of SVM of 20 samples = 0.4066666666666667
The accuracy score of SVM of 40 samples = 0.37333333333333335
The accuracy score of SVM of 60 samples = 0.43
The accuracy score of SVM of 80 samples = 0.395
The accuracy score of SVM of 100 samples = 0.36833333333333335
The accuracy score of SVM of 120 samples = 0.3616666666666667
The accuracy score of SVM of 140 samples = 0.37
The accuracy score of SVM of 160 samples = 0.37
The accuracy score of SVM of 180 samples = 0.36333333333333334
The accuracy score of SVM of 200 samples = 0.37166666666666665
```

In [77]:
```
L = np.arange(1,11,1)
acc_c = []
idx_lis = []
for i in L:
    idx = 2*i
    idx_lis.append(idx)
    y_h_r = np.where(Y_train_h_r[:idx] == 0 , -1, 1)
    clf_3 = QN_S3VM(X_train_h_r[:idx,:].tolist(), y_h_r.tolist(), X_trai
n_h_r[idx:,:].tolist(),
                    lam=1, kernel_type='Linear', random_generator=random
.Random())
    clf_3.train()
    y_pred_test = clf_3.getPredictions(X_test_h_r)
    acc_s3vm = accuracy_score(y_h_test, y_pred_test)
    acc_c.append(acc_s3vm_l)
    print(f'The accuracy score of QN_S3VM of {idx} samples = {acc_s3vm}'
)
```

```
The accuracy score of QN_S3VM of 2 samples = 0.495
The accuracy score of QN_S3VM of 4 samples = 0.5
The accuracy score of QN_S3VM of 6 samples = 0.5
The accuracy score of QN_S3VM of 8 samples = 0.5
The accuracy score of QN_S3VM of 10 samples = 0.5
The accuracy score of QN_S3VM of 12 samples = 0.5
The accuracy score of QN_S3VM of 14 samples = 0.5
The accuracy score of QN_S3VM of 16 samples = 0.5
The accuracy score of QN_S3VM of 18 samples = 0.5
The accuracy score of QN_S3VM of 20 samples = 0.5
```

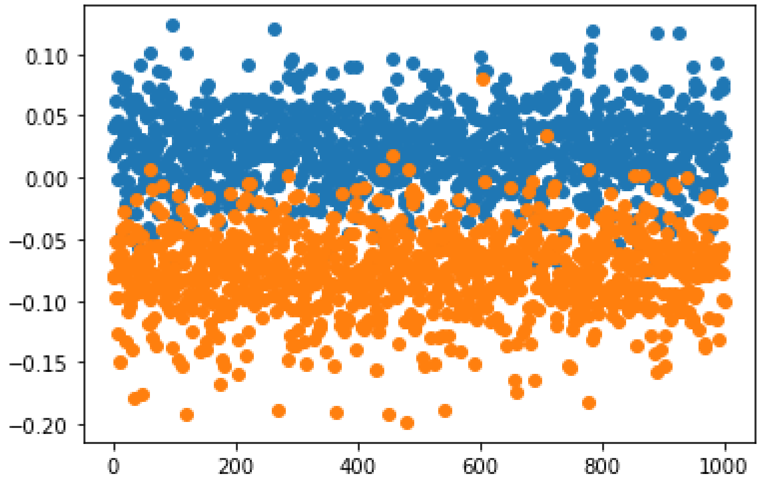
Unsupervised Learning

In [276]:
```
# EM Algorithm
X_train_h_r, X_test_h_r, Y_train_h_r, Y_test_h_r = train_test_split(X_re
view_h, y_h, train_size = 1000,
                                                                    test_
size=300,random_state=200, stratify =y_h)
gmm = GaussianMixture(n_components=2, random_state=28).fit(X_train_h_r)
y_pred = gmm.predict(X_test_h_r)
acc_sc = classification_report(Y_test_h_r, y_pred, output_dict=False)
print('The accuracy EM of USL =\n ', acc_sc)

means = gmm.means_
cov_mat = gmm.covariances_
conv = gmm.n_iter_
print('The means = \n', means.shape)
print('The covariance matrices = \n', cov_mat.shape)
print('The convergence steps = \n', conv)
```

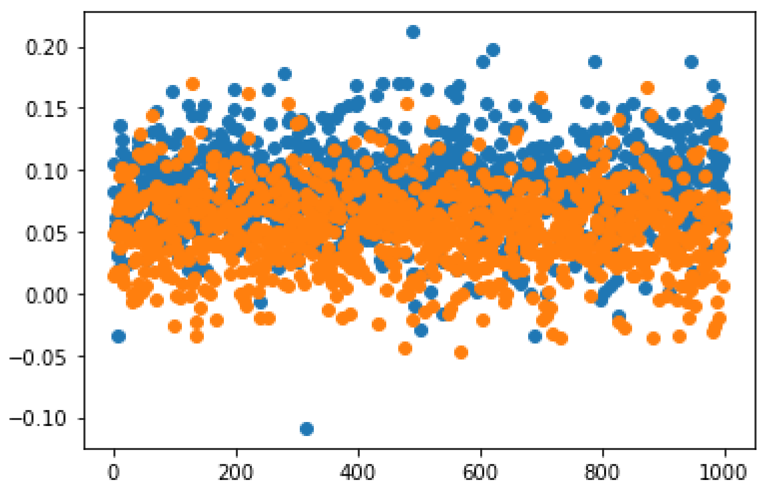
```
The accuracy EM of USL =
                precision    recall  f1-score   support

           0       0.56      0.73      0.64       150
           1       0.62      0.44      0.51       150

    accuracy                           0.58       300
   macro avg       0.59      0.58      0.57       300
weighted avg       0.59      0.58      0.57       300

The means =
 (2, 300)
The covariance matrices =
 (2, 300, 300)
The convergence steps =
 3
```

gmm = GaussianMixture(n_components=2).fit(X_train_h_r) y_pred = gmm.predict(X_train_h_r) acc_sc = classification_report(Y_train_h_r, y_pred, output_dict=False) print('The accuracy EM of USL =\n ', acc_sc)

In [278]:
```
x_axis = np.arange(1,1001,1)
plt.scatter(x_axis, X_train_h_r[:,0])
plt.scatter(x_axis, X_train_h_r[:,7])
plt.show()
```



In [279]:
```
x_axis = np.arange(1,1001,1)
plt.scatter(x_axis,X_train_h_r[:,8])
plt.scatter(x_axis,X_train_h_r[:,1])
plt.show()
```



In [334]:
```
# kmeans Algorithm
centroid, label = kmeans2(X_train_h_r, k = 2, seed=22)

y_pred1 = np.zeros((len(X_test_h_r)))

for i in range(len(X_test_h_r)):
    if distance.euclidean( X_test_h_r[i,:] , centroid[0,:] ) < distance.
euclidean( X_test_h_r[i,:] , centroid[1,:] ):
        y_pred1[i] = 0
        #y_e_r = y[label == i]
    elif distance.euclidean( X_test_h_r[i,:] , centroid[0,:] ) >= distan
ce.euclidean( X_test_h_r[i,:] ,centroid[1,:]):
        y_pred1[i] = 1
    else:
        y_pred1[i] = None


acc_scc = accuracy_score(Y_train_h_r, label)
print(f'The accuracy of kmeans for USL on train data home is = {acc_scc}
')


acc_scc = accuracy_score(Y_test_h_r, y_pred1)
print(f'The accuracy of kmeans for USL on test data home is = {acc_scc}'
)

cm = confusion_matrix(Y_test_h_r, y_pred1)
print(cm)
```

```
The accuracy of kmeans for USL on train data home is = 0.621
The accuracy of kmeans for USL on test data home is = 0.61
[[100  50]
 [ 67  83]]
```

In [ ]: