10 Amazon Fine Food Reviews Analysis_Clustering

April 2, 2019

1 Amazon Fine Food Reviews Analysis

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews

EDA: https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/

The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454 Number of users: 256,059 Number of products: 74,258 Timespan: Oct 1999 - Oct 2012 Number of Attributes/Columns in data: 10

Attribute Information:

1. Id
2. ProductId - unique identifier for the product
3. UserId - unqiue identifier for the user
4. ProfileName
5. HelpfulnessNumerator - number of users who found the review helpful
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
7. Score - rating between 1 and 5
8. Time - timestamp for the review
9. Summary - brief summary of the review
10. Text - text of the review

Objective: Given a review, determine whether the review is positive (rating of 4 or 5) or negative (rating of 1 or 2).

[Q] How to determine if a review is positive or negative? [Ans] We could use Score/Rating. A rating of 4 or 5 can be cosnidered as a positive review. A rating of 1 or 2 can be considered as negative one. A review of rating 3 is considered nuetral and such reviews are ignored from our analysis. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

2 [1]. Reading Data

2.1 [1.1] Loading the data

The dataset is available in two forms 1. .csv file 2. SQLite Database

In order to load the data, We have used the SQLITE dataset as it is easier to query the data and visualise the data efficiently.

Here as we only want to get the global sentiment of the recommendations (positive or negative), we will purposefully ignore all Scores equal to 3. If the score is above 3, then the recommendation wil be set to "positive". Otherwise, it will be set to "negative".

```
In [1]: %matplotlib inline
        import warnings
        warnings.filterwarnings("ignore")


        import sqlite3
        import pandas as pd
        import numpy as np
        import nltk
        import string
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.feature_extraction.text import TfidfTransformer
        from sklearn.feature_extraction.text import TfidfVectorizer

        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.metrics import confusion_matrix
        from sklearn import metrics
        from sklearn.metrics import roc_curve, auc
        from nltk.stem.porter import PorterStemmer

        import re
        # Tutorial about Python regular expressions: https://pymotw.com/2/re/
        import string
        from nltk.corpus import stopwords
        from nltk.stem import PorterStemmer
        from nltk.stem.wordnet import WordNetLemmatizer

        from gensim.models import Word2Vec
        from gensim.models import KeyedVectors
        import pickle

        from tqdm import tqdm
        import os
        import sys

        from sklearn.model_selection import train_test_split
        from sklearn.model_selection import cross_val_score
        from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
        from sklearn.model_selection import GridSearchCV
        from wordcloud import WordCloud
        from sklearn.cluster import KMeans
        from sklearn.cluster import AgglomerativeClustering
        from sklearn.cluster import DBSCAN
```

```
from sklearn.neighbors import NearestNeighbors
```

In [2]:
```
# using SQLite Table to read data.
con = sqlite3.connect(os.path.join( os.getcwd(), '..', 'database.sqlite' ))

# filtering only positive and negative reviews i.e.
# not taking into consideration those reviews with Score=3
# SELECT * FROM Reviews WHERE Score != 3 LIMIT 500000, will give top 500000 data point
# you can change the number to any other number based on your computing power

# filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 LIMIT 5
# for tsne assignment you can take 5k data points

filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3""", con)

# Give reviews with Score>3 a positive rating(1), and reviews with a score<3 a negativ
def partition(x):
    if x < 3:
        return 0
    return 1

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print("Number of data points in our data", filtered_data.shape)
filtered_data.head(3)
```

```
Number of data points in our data (5000, 10)
```

Out[2]:
```
   Id   ProductId          UserId                      ProfileName  \
0   1  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
1   2  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
2   3  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"

   HelpfulnessNumerator  HelpfulnessDenominator  Score        Time  \
0                     1                       1      1  1303862400
1                     0                       0      0  1346976000
2                     1                       1      1  1219017600

                 Summary                                                Text
0  Good Quality Dog Food  I have bought several of the Vitality canned d...
1      Not as Advertised  Product arrived labeled as Jumbo Salted Peanut...
2  "Delight" says it all  This is a confection that has been around a fe...
```

In [3]:
```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
```

```
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [4]:
```
print(display.shape)
display.head()
```

```
(80668, 7)
```

Out[4]:
```
                UserId   ProductId             ProfileName        Time  Score  \
0  #oc-R115TNMSPFT9I7  B005ZBZLT4                 Breyton  1331510400      2
1  #oc-R11D9D7SHXIJB9  B005HG9ESG  Louis E. Emory "hoppy"  1342396800      5
2  #oc-R11DNU2NBKQ23Z  B005ZBZLT4        Kim Cieszykowski  1348531200      1
3  #oc-R11O5J5ZVQE25C  B005HG9ESG           Penguin Chick  1346889600      5
4  #oc-R12KPBODL2B5ZD  B007OSBEV0   Christopher P. Presta  1348617600      1

                                                Text  COUNT(*)
0  Overall its just OK when considering the price...         2
1  My wife has recurring extreme muscle spasms, u...         3
2  This coffee is horrible and unfortunately not ...         2
3  This will be the bottle that you grab from the...         3
4  I didnt like this coffee. Instead of telling y...         2
```

In [5]:
```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[5]:
```
               UserId   ProductId                       ProfileName        Time  \
80638  AZY10LLTJ71NX  B001ATMQK2  undertheshrine "undertheshrine"  1296691200

       Score                                                Text  COUNT(*)
80638      5  I bought this 6 pack because for the price tha...         5
```

In [6]:
```
display['COUNT(*)'].sum()
```

Out[6]: 393063

3 [2] Exploratory Data Analysis

3.1 [2.1] Data Cleaning: Deduplication

It is observed (as shown in the table below) that the reviews data had many duplicate entries. Hence it was necessary to remove duplicates in order to get unbiased results for the analysis of the data. Following is an example:

In [7]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

```
Out[7]:        Id   ProductId         UserId      ProfileName  HelpfulnessNumerator  \
        0   78445  B000HDL1RQ  AR5J8UI46CURR  Geetha Krishnan                     2
        1  138317  B000HDOPYC  AR5J8UI46CURR  Geetha Krishnan                     2
        2  138277  B000HDOPYM  AR5J8UI46CURR  Geetha Krishnan                     2
        3   73791  B000HDOPZG  AR5J8UI46CURR  Geetha Krishnan                     2
        4  155049  B000PAQ75C  AR5J8UI46CURR  Geetha Krishnan                     2

           HelpfulnessDenominator  Score        Time  \
        0                       2      5  1199577600
        1                       2      5  1199577600
        2                       2      5  1199577600
        3                       2      5  1199577600
        4                       2      5  1199577600

                                   Summary  \
        0  LOACKER QUADRATINI VANILLA WAFERS
        1  LOACKER QUADRATINI VANILLA WAFERS
        2  LOACKER QUADRATINI VANILLA WAFERS
        3  LOACKER QUADRATINI VANILLA WAFERS
        4  LOACKER QUADRATINI VANILLA WAFERS

                                                     Text
        0  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
        1  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
        2  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
        3  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
        4  DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
```

As it can be seen above that same user has multiple reviews with same values for HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary and Text and on doing analysis it was found that ProductId=B000HDOPZG was Loacker Quadratini Vanilla Wafer Cookies, 8.82-Ounce Packages (Pack of 8) ProductId=B000HDL1RQ was Loacker Quadratini Lemon Wafer Cookies, 8.82-Ounce Packages (Pack of 8) and so on

It was inferred after analysis that reviews with same parameters other than ProductId belonged to the same product just having different flavour or quantity. Hence in order to reduce redundancy it was decided to eliminate the rows having same parameters.

The method used for the same was that we first sort the data according to ProductId and then just keep the first similar product review and delelte the others. for eg. in the above just the review for ProductId=B000HDL1RQ remains. This method ensures that there is only one representative for each product and deduplication without sorting would lead to possibility of different representatives still existing for the same product.

```
In [8]: #Sorting data according to ProductId in ascending order
        sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, inplace=Fals
```

```
In [9]: #Deduplication of entries
        final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep=
        final.shape
```

Out[9]: (4986, 10)

In [10]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[10]: 99.72

Observation:- It was also seen that in two rows given below the value of HelpfulnessNumerator is greater than HelpfulnessDenominator which is not practically possible hence these two rows too are removed from calcualtions

In [11]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[11]:
```
      Id   ProductId          UserId               ProfileName  \
0  64422  B000MIDROQ  A161DK06JJMCYF  J. E. Stephens "Jeanne"
1  44737  B001EQ55RW  A2V0I904FH7ABY                      Ram

   Helpfulness Numerator  Helpfulness Denominator  Score        Time  \
0                      3                        1      5  1224892800
1                      3                        2      4  1212883200

                                        Summary  \
0            Bought This for My Son at College
1  Pure cocoa taste with crunchy almonds inside

                                                Text
0  My son loves spaghetti so I didn't hesitate or...
1  It was almost a 'love at first bite' - the per...
```

In [12]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [13]:
```
#Before starting the next phase of preprocessing lets see the number of entries left
print(final.shape)

#How many positive and negative reviews are present in our dataset?
final['Score'].value_counts()
```

```
(4986, 10)
```

Out[13]:
```
1    4178
0     808
Name: Score, dtype: int64
```

4 [3] Preprocessing

4.1 [3.1]. Preprocessing Review Text

Now that we have finished deduplication our data requires some preprocessing before we go on further with analysis and making the prediction model.

Hence in the Preprocessing phase we do the following in the order below:-

1. Begin by removing the html tags
2. Remove any punctuations or limited set of special characters like , or . or # etc.
3. Check if the word is made up of english letters and is not alpha-numeric
4. Check to see if the length of the word is greater than 2 (as it was researched that there is no adjective in 2-letters)
5. Convert the word to lowercase
6. Remove Stopwords
7. Finally Snowball Stemming the word (it was obsereved to be better than Porter Stemming)

After which we collect the words used to describe positive and negative reviews

In [14]:
```
# printing some random reviews
sent_0 = final['Text'].values[0]
print(sent_0)
print("="*50)

sent_1000 = final['Text'].values[1000]
print(sent_1000)
print("="*50)

sent_1500 = final['Text'].values[1500]
print(sent_1500)
print("="*50)

sent_4900 = final['Text'].values[4900]
print(sent_4900)
print("="*50)
```

```
Why is this $[...] when the same product is available for $[...] here?<br />http://www.amazon.c
==================================================
I recently tried this flavor/brand and was surprised at how delicious these chips are.  The bes
==================================================
Wow.  So far, two two-star reviews.  One obviously had no idea what they were ordering; the otl
==================================================
love to order my coffee on amazon.  easy and shows up quickly.<br />This k cup is great coffee
==================================================
```

In [15]:
```
# remove urls from text python: https://stackoverflow.com/a/40823105/4084039
sent_0 = re.sub(r"http\S+", "", sent_0)
sent_1000 = re.sub(r"http\S+", "", sent_1000)
```

```
sent_150 = re.sub(r"http\S+", "", sent_1500)
sent_4900 = re.sub(r"http\S+", "", sent_4900)

print(sent_0)
```

```
Why is this $[...] when the same product is available for $[...] here?<br /> /><br />The Victo
```

In [16]:
```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how-to-remove-all
from bs4 import BeautifulSoup

soup = BeautifulSoup(sent_0, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1000, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1500, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_4900, 'lxml')
text = soup.get_text()
print(text)
```

```
Why is this $[...] when the same product is available for $[...] here? />The Victor M380 and M5
==================================================
I recently tried this flavor/brand and was surprised at how delicious these chips are.  The bes
==================================================
Wow.  So far, two two-star reviews.  One obviously had no idea what they were ordering; the ot
==================================================
love to order my coffee on amazon.  easy and shows up quickly.This k cup is great coffee.  dca
```

In [17]:
```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
```

```
        phrase = re.sub(r"\'re", " are", phrase)
        phrase = re.sub(r"\'s", " is", phrase)
        phrase = re.sub(r"\'d", " would", phrase)
        phrase = re.sub(r"\'ll", " will", phrase)
        phrase = re.sub(r"\'t", " not", phrase)
        phrase = re.sub(r"\'ve", " have", phrase)
        phrase = re.sub(r"\'m", " am", phrase)
        return phrase
```

In [18]:
```
sent_1500 = decontracted(sent_1500)
print(sent_1500)
print("="*50)
```

```
Wow.  So far, two two-star reviews.  One obviously had no idea what they were ordering; the oth
==================================================
```

In [19]:
```
#remove words with numbers python: https://stackoverflow.com/a/18082370/4084039
sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
print(sent_0)
```

```
Why is this $[...] when the same product is available for $[...] here?<br /> /><br />The Victor
```

In [20]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_1500 = re.sub('[^A-Za-z0-9]+', ' ', sent_1500)
print(sent_1500)
```

```
Wow So far two two star reviews One obviously had no idea what they were ordering the other war
```

In [21]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# <br /><br /> ==> after the above steps, we are getting "br br"
# we are including them into stop words list
# instead of <br /> if we have <br/> these tags would have revmoved in the 1st step

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselve
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him'
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "t
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'h
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'o
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'n
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't'
```

```
                    "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mi
                    "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
                    'won', "won't", 'wouldn', "wouldn't"])
```

[3.2] Preprocessing Review Text and Summary

```
In [22]: preprocessed_reviews = []
         preprocessed_summary = []
         # Sampling the data and preprocessing
         def data_sampling_preprocessing(final, no_of_samples):

             final = final.sample(n=no_of_samples, replace=True)

             # Combining all the above stundents
             from tqdm import tqdm
             preprocessed_reviews = []
             # tqdm is for printing the status bar
             for sentance in tqdm(final['Text'].values):
                 sentance = re.sub(r"http\S+", "", sentance)
                 sentance = BeautifulSoup(sentance, 'lxml').get_text()
                 sentance = decontracted(sentance)
                 sentance = re.sub("\S*\d\S*", "", sentance).strip()
                 sentance = re.sub('[^A-Za-z]+', ' ', sentance)
                 # https://gist.github.com/sebleier/554280
                 sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in st
                 preprocessed_reviews.append(sentance.strip())

             # Combining all the above stundents
             from tqdm import tqdm
             preprocessed_summary = []
             # tqdm is for printing the status bar
             for summary in tqdm(final['Summary'].values):
                 summary = re.sub(r"http\S+", "", summary)
                 summary = BeautifulSoup(summary, 'lxml').get_text()
                 summary = decontracted(summary)
                 summary = re.sub("\S*\d\S*", "", summary).strip()
                 summary = re.sub('[^A-Za-z]+', ' ', summary)
                 # https://gist.github.com/sebleier/554280
                 summary = ' '.join(e.lower() for e in summary.split() if e.lower() not in sto

                 preprocessed_summary.append(summary.strip())


             final['CleanedText'] = preprocessed_reviews #adding a column of CleanedText which
             final['CleanedText'] = final['CleanedText'].astype('str')

             final['CleanedSummary'] = preprocessed_summary #adding a column of CleanedSummary
             final['CleanedSummary'] = final['CleanedSummary'].astype('str')
```

```
final['Text_Summary'] = final['CleanedSummary'] + final['CleanedText']

# # store final table into an SQlLite table for future.
# conn = sqlite3.connect('final.sqlite')
# c=conn.cursor()
# conn.text_factory = str
# final.to_sql('Reviews', conn,  schema=None, if_exists='replace', \
#              index=True, index_label=None, chunksize=None, dtype=None)
# conn.close()

return final['Text_Summary']
```

5 [4] Featurization

5.1 [4.1] BAG OF WORDS

In [23]:
```
# #BoW
# count_vect = CountVectorizer() #in scikit-learn
# count_vect.fit(preprocessed_reviews)
# print("some feature names ", count_vect.get_feature_names()[:10])
# print('='*50)

# final_counts = count_vect.transform(preprocessed_reviews)
# print("the type of count vectorizer ",type(final_counts))
# print("the shape of out text BOW vectorizer ",final_counts.get_shape())
# print("the number of unique words ", final_counts.get_shape()[1])
```

5.2 [4.2] Bi-Grams and n-Grams.

In [24]:
```
# #bi-gram, tri-gram and n-gram

# #removing stop words like "not" should be avoided before building n-grams
# # count_vect = CountVectorizer(ngram_range=(1,2))
# # please do read the CountVectorizer documentation http://scikit-learn.org/stable/m

# # you can choose these numebrs min_df=10, max_features=5000, of your choice
# count_vect = CountVectorizer(ngram_range=(1,2), min_df=10, max_features=5000)
# final_bigram_counts = count_vect.fit_transform(preprocessed_reviews)
# print("the type of count vectorizer ",type(final_bigram_counts))
# print("the shape of out text BOW vectorizer ",final_bigram_counts.get_shape())
# print("the number of unique words including both unigrams and bigrams ", final_bigr
```

5.3 [4.3] TF-IDF

In [25]:
```
# tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
# tf_idf_vect.fit(preprocessed_reviews)
# print("some sample features(unique words in the corpus)",tf_idf_vect.get_feature_na
```

```
# print('='*50)

# final_tf_idf = tf_idf_vect.transform(preprocessed_reviews)
# print("the type of count vectorizer ",type(final_tf_idf))
# print("the shape of out text TFIDF vectorizer ",final_tf_idf.get_shape())
# print("the number of unique words including both unigrams and bigrams ", final_tf_i
```

5.4 [4.4] Word2Vec

In [26]:
```
# # Train your own Word2Vec model using your own text corpus
# i=0
# list_of_sentance=[]
# for sentance in preprocessed_reviews:
#     list_of_sentance.append(sentance.split())
```

In [27]:
```
# # Using Google News Word2Vectors

# # in this project we are using a pretrained model by google
# # its 3.3G file, once you load this into your memory
# # it occupies ~9Gb, so please do this step only if you have >12G of ram
# # we will provide a pickle file wich contains a dict ,
# # and it contains all our courpus words as keys and  model[word] as values
# # To use this code-snippet, download "GoogleNews-vectors-negative300.bin"
# # from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit
# # it's 1.9GB in size.


# # http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17SRFAzZPY
# # you can comment this whole cell
# # or change these varible according to your need

# is_your_ram_gt_16g=False
# want_to_use_google_w2v = False
# want_to_train_w2v = True

# if want_to_train_w2v:
#     # min_count = 5 considers only words that occured atleast 5 times
#     w2v_model=Word2Vec(list_of_sentance,min_count=5,size=50, workers=4)
#     print(w2v_model.wv.most_similar('great'))
#     print('='*50)
#     print(w2v_model.wv.most_similar('worst'))

# elif want_to_use_google_w2v and is_your_ram_gt_16g:
#     if os.path.isfile('GoogleNews-vectors-negative300.bin'):
#         w2v_model=KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300
#         print(w2v_model.wv.most_similar('great'))
#         print(w2v_model.wv.most_similar('worst'))
#     else:
```

```
#         print("you don't have gogole's word2vec file, keep want_to_train_w2v = True
```

In [28]:
```
# w2v_words = list(w2v_model.wv.vocab)
# print("number of words that occured minimum 5 times ",len(w2v_words))
# print("sample words ", w2v_words[0:50])
```

5.5 [4.4.1] Converting text into vectors using Avg W2V, TFIDF-W2V

[4.4.1.1] Avg W2v

In [29]:
```
# # average Word2Vec
# # compute average word2vec for each review.
# sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
# for sent in tqdm(list_of_sentance): # for each review/sentence
#     sent_vec = np.zeros(50) # as word vectors are of zero length 50, you might need
#     cnt_words =0; # num of words with a valid vector in the sentence/review
#     for word in sent: # for each word in a review/sentence
#         if word in w2v_words:
#             vec = w2v_model.wv[word]
#             sent_vec += vec
#             cnt_words += 1
#     if cnt_words != 0:
#         sent_vec /= cnt_words
#     sent_vectors.append(sent_vec)
# print(len(sent_vectors))
# print(len(sent_vectors[0]))
```

[4.4.1.2] TFIDF weighted W2v

In [30]:
```
# # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
# model = TfidfVectorizer()
# tf_idf_matrix = model.fit_transform(preprocessed_reviews)
# # we are converting a dictionary with word as a key, and the idf as a value
# dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))
```

In [31]:
```
# # TF-IDF weighted Word2Vec
# tfidf_feat = model.get_feature_names() # tfidf words/col-names
# # final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = tfi

# tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored in this
# row=0;
# for sent in tqdm(list_of_sentance): # for each review/sentence
#     sent_vec = np.zeros(50) # as word vectors are of zero length
#     weight_sum =0; # num of words with a valid vector in the sentence/review
#     for word in sent: # for each word in a review/sentence
#         if word in w2v_words and word in tfidf_feat:
#             vec = w2v_model.wv[word]
# #             tf_idf = tf_idf_matrix[row, tfidf_feat.index(word)]
#             # to reduce the computation we are
```

```
#             # dictionary[word] = idf value of word in whole courpus
#             # sent.count(word) = tf valeus of word in this review
#             tf_idf = dictionary[word]*(sent.count(word)/len(sent))
#             sent_vec += (vec * tf_idf)
#             weight_sum += tf_idf
#     if weight_sum != 0:
#         sent_vec /= weight_sum
#     tfidf_sent_vectors.append(sent_vec)
#     row += 1
```

6 [5] Assignment 10: K-Means, Agglomerative & DBSCAN Clustering

Apply K-means Clustering on these feature sets:

SET 1:Review text, preprocessed one converted into vectors using (BOW)

```
    <li><font color='red'>SET 2:</font>Review text, preprocessed one converted into vectors usi
        <li><font color='red'>SET 3:</font>Review text, preprocessed one converted into vectors
        <li><font color='red'>SET 4:</font>Review text, preprocessed one converted into vectors
<li>Find the best k using the elbow-knee method (plot k vs inertia_)</li>
<li>Once after you find the k clusters, plot the word cloud per each cluster so that at a singl
```

go we can analyze the words in a cluster.

```
</ul>
</li>
<br>
<li><strong>Apply Agglomerative Clustering on these feature sets:</strong>
    <ul>
        <li><font color='red'>SET 3:</font>Review text, preprocessed one converted into vectors
        <li><font color='red'>SET 4:</font>Review text, preprocessed one converted into vectors
<li>Apply agglomerative algorithm and try a different number of clusters like 2,5 etc.</li>
<li>Same as that of K-means, plot word clouds for each cluster and summarize in your own words
        <li>You can take around 5000 reviews or so(as this is very computationally expensive or
    </ul>
</li>
<br>
<br>
<li><strong>Apply DBSCAN Clustering on these feature sets:</strong>
    <ul>
        <li><font color='red'>SET 3:</font>Review text, preprocessed one converted into vectors
        <li><font color='red'>SET 4:</font>Review text, preprocessed one converted into vectors
<li>Find the best Eps using the <a href='https://stackoverflow.com/questions/12893492/choosing-
<li>Same as before, plot word clouds for each cluster and summarize in your own words what that
        <li>You can take around 5000 reviews for this as well.</li>
    </ul>
</li>
```

6.1 [5.1] K-Means Clustering

In [32]:
```
# Source: https://docs.python.org/3/library/pickle.html

# Saving data to pickle file
def topicklefile(obj, file_name):
    pickle.dump(obj,open(file_name+'.pkl', 'wb'))
```

In [33]:
```
# Data from pickle file
def frompicklefile(file_name):
    data = pickle.load(open(file_name+'.pkl', 'rb'))
    return data
```

In [34]:
```
# Sort 'Time' column
final = final.sort_values(by='Time', ascending=True)
```

In [35]:
```
def apply_avgw2v_train_test(X_train):

    # Training own Word2Vec model using your own text corpus
    list_of_sent_train = []
    for sent in X_train:#final['Text_Summary'].values:
        list_of_sent_train.append(sent.split())
#     list_of_sent_test = []
#     for sent in X_test:#final['Text_Summary'].values:
#         list_of_sent_test.append(sent.split())

    # min_count = 5 considers only words that occured atleast 5 times
    w2v_model=Word2Vec(list_of_sent_train,min_count=5,size=50, workers=8)

    w2v_words = list(w2v_model.wv.vocab)
#     print("number of words that occured minimum 5 times ",len(w2v_words))
#     print("sample words ", w2v_words[0:50])

    # compute average word2vec for each review for train data
    avgw2v_train = []; # the avg-w2v for each sentence/review is stored in this list
    for sent in tqdm(list_of_sent_train): # for each review/sentence
        sent_vec = np.zeros(50) # as word vectors are of zero length
        cnt_words =0; # num of words with a valid vector in the sentence/review
        for word in sent: # for each word in a review/sentence
            if word in w2v_words:
                vec = w2v_model.wv[word]
                sent_vec += vec
                cnt_words += 1
        if cnt_words != 0:
            sent_vec /= cnt_words
        avgw2v_train.append(sent_vec)
#     print(len(avgw2v_train))
#     print(len(avgw2v_train[0]))
```

```
#     # compute average word2vec for each review for test data
#     avgw2v_test = []; # the avg-w2v for each sentence/review is stored in this list
#     for sent in tqdm(list_of_sent_test): # for each review/sentence
#         sent_vec = np.zeros(50) # as word vectors are of zero length
#         cnt_words =0; # num of words with a valid vector in the sentence/review
#         for word in sent: # for each word in a review/sentence
#             if word in w2v_words:
#                 vec = w2v_model.wv[word]
#                 sent_vec += vec
#                 cnt_words += 1
#         if cnt_words != 0:
#             sent_vec /= cnt_words
#         avgw2v_test.append(sent_vec)
# #     print(len(avgw2v_test))
# #     print(len(avgw2v_test[0]))

    return avgw2v_train
```

In [36]:
```
def apply_tfidfw2v_train_test(X_train):

    # Training own Word2Vec model using your own text corpus
    list_of_sent_train = []
    for sent in X_train:#final['Text_Summary'].values:
        list_of_sent_train.append(sent.split())
#     list_of_sent_test = []
#     for sent in X_test:#final['Text_Summary'].values:
#         list_of_sent_test.append(sent.split())

    # min_count = 5 considers only words that occured atleast 5 times
    w2v_model=Word2Vec(list_of_sent_train,min_count=5,size=50, workers=16)

    w2v_words = list(w2v_model.wv.vocab)

    model = TfidfVectorizer()
    tf_idf_matrix = model.fit_transform(X_train)
    # we are converting a dictionary with word as a key, and the idf as a value
    dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))

    # TF-IDF weighted Word2Vec
    tfidf_feat = model.get_feature_names() # tfidf words/col-names
    # final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = t

    tfidfw2v_train = []; # the tfidf-w2v for each sentence/review is stored in this l
    row=0;
    for sent in tqdm(list_of_sent_train): # for each review/sentence
        sent_vec = np.zeros(50) # as word vectors are of zero length
        weight_sum =0; # num of words with a valid vector in the sentence/review
```

```
        for word in sent: # for each word in a review/sentence
            if word in w2v_words and word in tfidf_feat:
                vec = w2v_model.wv[word]
#               tf_idf = tf_idf_matrix[row, tfidf_feat.index(word)]
                # to reduce the computation we are
                # dictionary[word] = idf value of word in whole courpus
                # sent.count(word) = tf valeus of word in this review
                tf_idf = dictionary[word]*(sent.count(word)/len(sent))
                sent_vec += (vec * tf_idf)
                weight_sum += tf_idf
        if weight_sum != 0:
            sent_vec /= weight_sum
        tfidfw2v_train.append(sent_vec)
        row += 1


#     tf_idf_matrix = model.transform(X_test)
#     # we are converting a dictionary with word as a key, and the idf as a value
#     dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))

#     # TF-IDF weighted Word2Vec
#     tfidf_feat = model.get_feature_names() # tfidf words/col-names
#     # final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val =

#     tfidfw2v_test = []; # the tfidf-w2v for each sentence/review is stored in this
#     row=0;
#     for sent in tqdm(list_of_sent_test): # for each review/sentence
#         sent_vec = np.zeros(50) # as word vectors are of zero length
#         weight_sum =0; # num of words with a valid vector in the sentence/review
#         for word in sent: # for each word in a review/sentence
#             if word in w2v_words and word in tfidf_feat:
#                 vec = w2v_model.wv[word]
#     #             tf_idf = tf_idf_matrix[row, tfidf_feat.index(word)]
#                 # to reduce the computation we are
#                 # dictionary[word] = idf value of word in whole courpus
#                 # sent.count(word) = tf valeus of word in this review
#                 tf_idf = dictionary[word]*(sent.count(word)/len(sent))
#                 sent_vec += (vec * tf_idf)
#                 weight_sum += tf_idf
#         if weight_sum != 0:
#             sent_vec /= weight_sum
#         tfidfw2v_test.append(sent_vec)
#         row += 1


    return tfidfw2v_train
```

In [37]:
```
# Applying BOW on train and test data and creating the
```

```
from sklearn.preprocessing import StandardScaler
from scipy.sparse import hstack


def apply_vectorizers_train_test(cluster_algo, model_name):

    if cluster_algo == 'K-Means' and model_name == 'BOW':
        train_data = data_sampling_preprocessing(final, 50000)
        topicklefile(train_data,'train_data')
    elif (cluster_algo == 'Agglomerative' and model_name == 'AvgW2V') or (cluster_alg
        train_data = data_sampling_preprocessing(final, 5000)
        topicklefile(train_data,'train_data')
    else:
        train_data = frompicklefile('train_data')

    if model_name == 'BOW':
        #Applying BoW on Train data
        count_vect = CountVectorizer()

        #Applying BoW on Test data
        train_vect = count_vect.fit_transform(train_data)

        #Applying BoW on Test data similar to the bow_train data
#         test_vect = count_vect.transform(test_data)

        topicklefile(train_vect, 'train_vect')
#         topicklefile(test_vect, 'test_vect')

        print("'train_vect' and 'test_vect' are the pickle files.")
        return count_vect

    elif model_name == 'TF-IDF':
        #Applying TF-IDF on Train data
        count_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)

        #Applying BoW on Test data
        train_vect = count_vect.fit_transform(train_data)

        #Applying BoW on Test data similar to the bow_train data
#         test_vect = count_vect.transform(test_data)

        topicklefile(train_vect, 'train_vect')
#         topicklefile(test_vect, 'test_vect')

        print("'train_vect' and 'test_vect' are the pickle files.")
        return count_vect

    elif model_name == 'AvgW2V':
```

```
            train_vect = apply_avgw2v_train_test(train_data)

            topicklefile(train_vect, 'train_vect')
#             topicklefile(test_vect, 'test_vect')
            print("'train_vect' and 'test_vect' are the pickle files.")

        elif model_name == 'TF-IDF W2V':
            train_vect = apply_tfidfw2v_train_test(train_data)

            topicklefile(train_vect, 'train_vect')
#             topicklefile(test_vect, 'test_vect')
            print("'train_vect' and 'test_vect' are the pickle files.")

        else:
            #Error Message
            print('Model specified is not valid! Please check.')
```

In [38]:
```
#Source: https://www.datasciencecentral.com/profiles/blogs/python-implementing-a-k-me
# https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html
def apply_kmeans(data, clusters):
#     clusters = [i for in range(2,10,2)]
    inertias = []

    for cluster in clusters:
        kmeans = KMeans(n_clusters=cluster, n_jobs=-1)
        kmeans.fit(data)
        inertias.append(kmeans.inertia_)
    print('inertias: ',inertias)
    return inertias
```

In [39]:
```
def apply_agglomerative(cluster, train_data):
    agg_clustering = AgglomerativeClustering(n_clusters=cluster)
    agg_clustering.fit(train_data)
    labels = agg_clustering.labels_
    return labels
```

In [52]:
```
#Source: https://stackoverflow.com/questions/48010276/how-to-estimate-eps-using-knn-d
def k_dist_plot(neighbors, data, noOfPoints):
    nn = NearestNeighbors(n_neighbors=neighbors).fit(data)
    distances, indices = nn.kneighbors(data)
    distanceDec = sorted(distances[:,neighbors-1], reverse=False)
    plt.xlabel('No. of points')
    plt.ylabel('100-NN distnace')
    plt.plot(list(range(1,noOfPoints+1)), distanceDec)
```

In [41]:
```
def apply_dbscan(eps_given, min_samples_given, train_data):
    dbscan = DBSCAN(eps=eps_given, min_samples=min_samples_given, n_jobs=-1)
    dbscan.fit(train_data)
```

```
        labels = dbscan.labels_
        return labels
```

In [42]:
```
def plot_scores_vs_clusters(clusters, inertias):
    plt.plot(clusters, inertias)
    plt.xlabel('n_clusters')
    plt.ylabel('Scores')
    plt.title('Elbow Plot')
    plt.show()
```

In [43]:
```
def retrain_kmeans(cluster, data):
    kmeans = KMeans(n_clusters=cluster, n_jobs=-1)
    kmeans.fit(data)
    labels = kmeans.labels_
    return labels
```

In [44]:
```
#https://andrew47.github.io/scikitlearn-cluster.html
from collections import Counter
def build_wordclouds(X, labels):
    review_labels = pd.DataFrame({'Reviews':X, 'labels':labels})

    for cluster in review_labels['labels'].unique():
        review_labels1 = review_labels[review_labels['labels']==cluster]

        if len(review_labels1) != 0:
            text = ' '.join(review_labels1['Reviews'])
            text = text.split()

            counts = Counter(text)
            counts = dict(counts)

            wordcloud = WordCloud(background_color = 'white')
            kMeansWordCloud = wordcloud.generate_from_frequencies(counts)
            plt.figure()
            plt.imshow(kMeansWordCloud)
            plt.axis("off")
```

6.1.1 [5.1.1] Applying K-Means Clustering on BOW, SET 1

In [164]:
```
# Please write all the code with proper documentation
```

In [165]:
```
count_vect = apply_vectorizers_train_test('K-Means','BOW')
```

```
100%|| 50000/50000 [00:26<00:00, 1875.90it/s]
100%|| 50000/50000 [00:17<00:00, 2895.39it/s]


'train_vect' and 'test_vect' are the pickle files.
```

```
In [166]: train_vect = frompicklefile('train_vect')
          print(train_vect.shape)
```

```
(50000, 17532)
```

```
In [167]: clusters = [i for i in range(3,30,3)]
          inertias = apply_kmeans(train_vect, clusters)
```

```
inertias:  [2783171.565628858, 2698353.9100002875, 2645691.5378675917, 2605413.098695551, 2566
```

```
In [168]: plot_scores_vs_clusters(clusters, inertias)
```

```
In [173]: labels = retrain_kmeans(21, train_vect)
          # cluster_centers.columns = count_vect.get_feature_names()
```

6.1.2 [5.1.2] Wordclouds of clusters obtained after applying k-means on BOW SET 1

```
In [174]: # Please write all the code with proper documentation
```

```
In [175]: X = frompicklefile('train_data')
          build_wordclouds(X, labels)
```

product
love
not
good
taste
amazon
would
like
best
flavor
one
time
price
find
coffee
get
buy
no
really
delicious
excellent
much
tried
tea
food
use
order
make
ever
well
little
also
recommend
chocolate
easy
better
quality
first
sweet
chips
box
hot
water
try
without
even
made
healthy
gluten
salt
bought
found
could
favorite
free
eat
used
years
dog
bag
makes
fresh
small
perfect
sugar
flavors
stuff
shipping
wonderful
cookies
regular
tasty
pack
cups
family
highly
organic
cage
cost
acid
bit
ice
cooking
brand
hard
go
snack
way
always
nice
cancel
day
tastes
cake
dry
give
nik
loves
cheap
taste
great
amazon
little
would
find
coffee
one
price
use
buy
good
not
like
time
also
food
favorite
really
best
love
product
flavor
tea
much
free
make
snack
recommend
better
hot
no
nice
many
loves
makes
found
easy
well
even
eat
could
delicious
chips
always
perfect
tried
way
water
gluten
excellent
healthy
tastes
chocolatestuff
brand
highly

not
like
would
one
taste
product
good
flavor
great
water
could
best
much
also
get
buy
price
really
no
try
love
make
eat
use
order
coconut
still
first
even
little
sugar
box
time
tried
found
better
tastes
since
way
drink

food
dog
newman
would
bag
dogs
foods
amazon
not
like
also
great
one
good
well
love
cat
eat
no
since
much
year
years
healthy
two
cats
quality
eating
loves
really
time
get
best
old
chicken
formula
tried
diet
brand
products

chips
flavor
not
like
salt
great
taste
bag
good
potato
kettle
bags
chip
love
much
spicy
snack
amazon
would
better
eat
vinegar
best
flavors
delicious
pack
salty
really
no
low
little
find
price
get
thai
favorite
crunchy
buy
product
first
popchips
brand
calories
recommend
tox

tea
green
like
drink
water
one
teas
black
white
use
also
sugar
coffee
little
would
taste
good
bag
well
get
hot
iced
not
flavor
much
sweet
tried
great
bags
love
find
loose
makes
made
found
leaves
drinking
no
product

chocolate
hot
taste
like
best
cup
not
dark
cocoa
good
cups
flavor
milk
cookies
would
keurig
square
great
coffee
product
tried
water
better
much
one
brand
recommend
cookie
try
sugar
make
love
found
even
find
tastes
use
peck
perfect
bought
delicious
little
really
bad
favorite
price
drink

mix
pancakes
free
gluten
not
make
pancake
waffles
like
bisquick
cake
good
great
one
use
also
taste
chicken
add
would
recipe
product
gf
box
tried
made
baking
waffle
try
time
fluffy
biscuits
butter
easy
could
best
even
recipes
whole
regular
store
flavor
makes
found
using
no
milk
fruit
wheat
syrup
think
well
used
amazon
little
better
know
family
years

gluten
free
mix
not
bisquick
products
biscuits
many
would
best
make
regular
little
crocker
making
good
use
one
pancakes
like
baking
product
flour
mixes
found
find
tried
recipes
made
betty
taste
sure
chicken
dairy
milk
texture
xanthan
substitute
powder
shortening
sugar
well
even
better
local
recipe
using
amazon
really
since
used
time
let
years
store
disease
gluten
ingredients
price
cost
buy
pound
every
pasta
tried
also
no
trying
bread

chips
brand
potato
kettle
not
like
salt
chip
taste
time
great
crisp
product
would
bag
flavors
bags
also
cream
know
sea
buy
jalapeno
cheddar
used
vinegar
purchase
might
mouth
high
get
quality
flavor
sour
no
spicy
price
uncle
carrel
best
well
natural
first
bottom
soft
crunchy
onion
try
recommend
disappointed
ordered
calorie
good
fantastic
definitely
much
packaging
company
tasty
seasoning
plain
awful
broken
crunch
explain
overall
makp
cz
fact
texture
rancid
probably
another
pepper
pop
pack
true
dairy
sticky
popcorn
popcornx

coffee
not
decaf
taste
good
like
flavor
cup
would
drink
great
water
one
strong
get
really
better
smooth
much
dark
melitta
bag
rich
blend
love
even
starbucks
little
black
aroma
bitter
beans
find
grind
roast
mountain
delicious

organic
dog
not
food
best
one
ingredients
water
product
pet
meat
baby
earth
chicken
products
rice
pets
much
would
brand
vitamin
newman
better
anything
healthy
value
waste
buy
like
blue
meals
brands
make
bpa
natural
good
cheap
brown
still
no
really
corn

not
like
good
rice
would
raw
make
use
also
amazon
product
taste
mix
box
alive
milk
get
cacao
powder
different
much
aware
flour
coffee
recipe
nice
know
enough
really
even
one
time
bag
bit
price
biscuits
still
chocolate
without
made
around
butter
food
fence

egg
calcium
allergy
baby
dha
formula
formulas
however
calories
bone
children
ratio
white
phosphorus
usually
child
also
foods
organic
yolk
not
infant
supplement
proteins
milligrams
per
good
eggs
meet
age
reply
please

kettle
chips
brand
flavour
potato
also
flavors
brands
solomon
bags
wine
not
sweet
chip
honey
flavor
would
great
much
one
pairing
backyard
dijon
good
spicy
crunchy
beer
salty
vegetable
unmistakable
ounce
major
kosher
friends
natural
certified
organic
product
trans
amazing
strong
sticks
company
healthier
annette
balance
state
gold
wonderful
lime
will
missing
crunch
moderate
artificial
bring
light
color
tell
amount
recommend
lemony

coconut
water
pineapple
splash
juice
natural
drink
not
sweet
cane
winter
snack
summer
packaging
could
would
green
no
vitamin
little
beverage
paired
refreshing
contains
source
sugar
flavor
almost
overpowering
salt
standalone
cheese
throw
today
according
good
potassium
cholesterol
tasty
package
type
another
personally
really
light
think
needed
savory
sweetn
however
left
completely
banana
palatable
prior
crackers
young
says
became
mace
heard
also
round
electrolytes
appetizer
like
experience
go
bag
weather
fruit
invigorating
essential
five
store

food
product
meat
dog
corn
eat
dogs
beneful
purchasing
ingredients
chicken
thing
disgusting
first
veggies
products
advertising
recommend
poop
bag
meal
would
full
awful
filler
even
not
three
feel
terrible
yellow
thought
digest
false
chunks
feedbought
feeding
retain
ground

difference
bottle
food
not
could
glass
store
versus
local
make
really
tried
happy
best
makes
box
stuff
amazon
taste
boxed
health
big
sell
shipping
would
something
product
city
served
tester
version
tiny
home
bubbles
drink
brand
blink
strolling
case
wine
brands
always
want
year
tasted
buck
card
chilled
ready
price
get
guess
far
find
take
visiting
bummed
bottled
allergy
packaged
high
know
world
bang
bigger
compared
write
reduce
love
lite
scratched
age
racks
getting
fake

food
not
eating
really
cat
whatever
little
always
day
would
three
got
no
around
hungry
real
much
science
morning
bean
things
feeding
made
back
eat
diet
cup
four
way
feed

lays
sodium
total
would
potassium
bbq
bag
chips
taste
not
fiber
much
popchips
baked
vitamin
carbohydrate
calories
slightly
protein
calcium
flavor
fat
dietary
benefit
nutrition
sugars
amazon
favorite
ordered
serving
packaging
information

6.1.3 [5.1.3] Applying K-Means Clustering on TFIDF, SET 2

```
In [176]: # Please write all the code with proper documentation
```

```
In [177]: count_vect = apply_vectorizers_train_test('K-Means','TF-IDF')
```

```
'train_vect' and 'test_vect' are the pickle files.
```

```
In [178]: train_vect = frompicklefile('train_vect')
          print(train_vect.shape)
```

```
(50000, 94974)
```

```
In [179]: clusters = [i for i in range(3,30,3)]
          inertias = apply_kmeans(train_vect, clusters)
```

```
inertias:  [49316.776665936064, 48977.8197000542, 48778.11200163094, 48691.98896329182, 48583.8
```

```
In [180]: plot_scores_vs_clusters(clusters, inertias)
```

```
In [181]: labels = retrain_kmeans(24, train_vect)
          # cluster_centers.columns = count_vect.get_feature_names()
```

6.1.4 [5.1.4] Wordclouds of clusters obtained after applying k-means on TFIDF SET 2

```
In [182]: # Please write all the code with proper documentation
```

```
In [183]: X = frompicklefile('train_data')
          build_wordclouds(X, labels)
```

cookies
not
like
taste
great
flavor
chocolate
would
cookie
box
store
good
buy
delicious
bought
no
make
two
time
one
soft
really
eat
drinks
butter
water
sweet
chips
much
found
still
try
better
find
chip
love
also
oz
even
recipe
something
boxes
recommend
tune
dripped
fortune
a'rived

not
bread
would
like
good
gift
time
try
christmas
make
even
use
also
easy
taste
since
came
well
one
think
amazon
buy
keep
product
best
get
find
love
makes
white
nice
delicious
much
fresh
butter
rice
potato
quality
really
mace
loved
excellent
everything
enough
cakes
tried
price
machine
mixer
baking
eating
hodgson
perfect
without

flavor
like
not
taste
good
oil
powder
best
use
gum
lime
one
chowder
milk
tastes
add
product
better
great
syrup
amazon
sugar
no
corn
much
clam
ingredients
make
also
natural
would
pie
organic
price
really
stevia
pepper
real
salt
chocolate

great
like
not
good
taste
flavor
much
make
find
one
add
use
food
little
price
love
also
would
tastes
best
time
stuff
sweet
eat
buy
chicken
snack
really
product
well
tried
better
hot
sauce
even
try
no
favorite
get
bit
lasty
pasta
recommend
fresh
amazon
rice
perfect
hard
spicy
first
flavors
brand
always
think
healthy
chocolate

not
one
delicious
great
cheese
try
could
bag
favorite
pop
eat
good
would
chips
sweet
go
little
taste
find
snack
love
flavor
flavors
like
calorie
best
healthy
popchips
crunchy
perfect
different
better
get
make
calories
first
think
time
product
variety
weight
easy
worth
size
yummy
cries
found
fantastic
loved
caviar
purchased
single
absolutely
whole
tried
snacks
even
enjoy
way
box
day
free
stale
yet
table
mount
food
dry
right
propane
fizzy
taffy
also
give
bit
small
still
work
bad
price
rice
nice
well
add
bags
packs
product

product
find
buy
store
cake
try
excellent
recommend
even
made
local
little
best
taste
mix
would
could
great
amazon
love
also
one
much
using
tried
icicle
order
use
make
not
like
way
found
pancakes
bought
box
good
price
no
really
many
time
get
better
well
food
used
ordered
drink
quality
still
bag
batch
worth
different
enough
since
years
fresh
work
sugar
refer
long
hard
think
works
stores
easy
buying
anything
family
purchase
excellent
first
flavor
sure
something
small
put
aid
good
bad
mix
tasty
two
year
day
size
bag
going
delicious
sophisticated
makes
usually
though
fresh

chips
salt
not
flavor
vinegar
like
love
pepper
bag
great
taste
flavors
good
potato
popchips
would
favorite
sea
crunchy
better
sour
eat
bbq
variety
salty
much
snack
pack
bags
really
fat
best
little
time
try
chip
low
even
regular
onion
calories

food
dog
newman
like
would
love
not
dogs
great
one
bag
loves
good
eat
organic
product
no
old
quality
ingredients
foods
healthy
brand
natural
better
well
much
also
price
since
two
find
little
never
best
amazon
years
feed
products
started
always

cocoa
hot
cup
not
chocolate
great
cups
taste
tried
good
like
flavor
keurig
grove
square
coffee
best
dark
product
milk
sugar
would
much
brand
water
give
recommend
juice
brewer
box
try
rich
one
delicious
brands
products
however
small
better
pretty
different
definitely
rendering
time
three
aftertastemakes
little
find
far
tasted
actually
size
make
sweetener
pack
highly
tastes
really
well
even
perfect
luve
ingredients
peppermint
order
k

gluten
free
not
mix
product
baking
great
many
pancakes
taste
waffles
husband
know
since
even
making
flour
pancake
gf
tried
recipes
using
flavor
really
milk
whole
products
like
good
recommend
chicken
recipe
sugar
best
found
once
tell
store
try
used
years
find
also
regular
would
wheat
box
made
use
local
family
make
love
makes
biscuits
bisquick
time
better
baked
cake
cracker
bread
tastes
panela
dairy
disease
mixes
eating
foods
wonderful
tiny
dietary
lovely
dist
gp

chips
potato
great
not
good
like
bag
bags
kettle
chip
taste
flavor
flavors
brand
best
amazon
muchamazon
no
salt
baked
better
love
snack
would
get
one
spicy
salty
crunch
crunchy
buy
also
pack
perfect
calories
recommend
really
regular
product
delicious
enough
sweet
fried
healthy
first
ever
try
time
pop
bought
price
ordered
lays
without
know
could
enjoy
pretty
right
nice
bit
$1.20
variety
butter
cape
fat
used
fried
boxed
oatmeal
favorite
bit
sonar
trying
made
taste
tasty
little
amount
poochips
many
tries
bitter
weather
healthier
spicy
onion
never
trial
nuts
sour
want
crisps
cooking
bites
bland
cheap
cooked
dried
bland
cooked
cooking
bites
bland
cheap
cooked
dried
bland
cooked
cooking
bites
bland
cheap
cooked
dried
bland
cooked

tea
noticed
taste
water
like
green
flavor
drink
tried
use
teas
love
would
one
great
good
sweet
bags
bag
get
no
black
product
tastes
loose
best
time
quality
recommend
day
much
hot
powder
little
delicious
natural
strong
lemon
pomegranate
leaves
buy
better
drinking
amazon
first
also
years
quality
nice
made
make
since
many
still
coffee
wonderful
whitebitter
price
cup
leaf
favorite
makes
very
ever
could
well
want
really
look
know
taste
flavors
used
several
never
tasty
pretty
sweet
sugar
fat
treat
box
pack
water
calorie
crispy
mini
cream
eat
cookies
product
calories
cookie
thin
really
crackers
crisps
quite
find
thing
nabisco
low
flavor
great
little
even
chocolate
would
like
not
good
one
no
love
healthy
favorite
snacks
products
work
made
small
tried
think
amazon
bag
breakfast
two
usually
tasty
less
enough
regular
crisp
snack
cookies
since
however
eating
texture
packages
taste
tracker
delicious
still
buy
box
tiny
water
pack
flavors
much
better
try
three
free
see
quantity
family
many
time
store
order
amount
years
also
nice
long
high
ordered
large
listing
different
enjoy
ever
way
want
really
could
well
look
know
parker
leaf
cup
price
wonderful
whitebitter
coffee
makes
very
ever
made
make
since
many
still
powder
little
delicious
natural
strong
lemon
pomegranate

chocolate
hot
not
cup
flavor
great
like
one
milk
taste
good
dark
best
would
keurig
tried
little
try
love
product
coffee
better
cups
find
k
much
price
favorite
peppermint
chocolates
sugar
cocoa
get
miss
bad
pack
use
made
brand
water
swiss
order
tastes
pretty
square
artificialrecommend
could
also

mix
not
pancake
pancakes
waffle
waffles
biscuits
bisquick
great
like
tried
make
add
recipe
good
best
one
product
taste
baking
price
even
time
using
use
butter
milk
flour
box
easy
really
delicious
never
flavor
buttermilk
syrup
gf
light
belgian
fluffy
reviews
texture
recipes
makes
also
used
water
much
fresh
years
gluten
ingredients
stonewall
family
amazon
without
regular
mixes
kitchen
wall
cup
made

price
not
amazon
shipping
product
good
cans
ordereduse
like
one
get
find
best
many
arrived
free
much
items
cheaper
canned
grocery
store
local
really
certified
box
item
time
buy
taste
great
order
tomatoes
no
would
love
fresh
better
received
however
excellent
since
shipped
case
new
came
way
stores

grass
not
food
cat
cats
felidae
one
eat
much
no
would
like
really
grain
good
old
great
love
product
time
also
free
grow
seeds
eating
get
diet
little
healthy
feed
water
quality
store
shipping

sinus
not
tea
great
product
really
taste
smell
wonderful
helps
would
back
good
almost
enjoyed
much
like
reviews
use
stuff
try
one
reading
relief
drinking
pain
difference
since
could
time
flavor
take
buying
find
kind
day
no
tests
ingredients
easily
orange
water
bath
upset
cup
hours
morning
worried
put
gct
goth
act
need
given
nice
cocy
well
sure
shower
buster
asparagus
based
nasal
told
family
worth
said
real
though
wash
little
issues
using
days
feel
congestion

licorice
sugar
not
great
good
black
product
flavor
taste
love
like
candy
panda
best
one
box
eat
cubes
sweet
soft
also
free
would
ever
much
no
candies
delicious
little
small
fruit
jelly
better
bought
liquid
aftertaste
make
buy
get
brown
excellent
real
really
highly
flavors
tasted
order
their
recommended
food
actually
blueberry
favorite
amazon
enjoy
buying
company
medical
natural
several
every
price
pretty
tasty
strong
found
many
use
gpt
cost
ordered
perfect
product
store
silent
people
gathered
had
made
locally
tried
first
recommend

water
not
coconut
like
product
splash
good
really
much
flavor
would
drink
sweet
no
natural
taste
pineapple
juice
one
sugar
refreshing
added

popcorn
shipping
ever
amish
taste
best
really
time
like
white
baby
well
flavor
good
popper
corn
much
product
country
get
every
not
pop
popping
hulls
great
kernels
small
little
popped
tender
delicious

great
water
not
coco
taste
love
use
coconut
good
much
pineapple
flavor
free
also
better
vita
drink
like
best
workout
content
hot
get mix
even
potassium
product
long
time
brand
sugar
little
really
buy
perfect
nutritional
convenience
plastic
regular
nutrients
vanilla
less
spray
feel
used
record
waffles
cheese
nothing
cup
found
natural
gift
keep
tagalong
hooked
bottles
banana
always
since
magnesium
ingredients
taste
bottle
thirst
drinks
drinking
recommend
enough
amount
work
others
tried
taste
brands
would
love
flavors
friends
pack
tired
value
pulp
when
sweet
sweetener
ice
sodium
water
sweetener
ice
sodium
water
sweetener
ice
sodium
water

```
In [189]: labels = retrain_kmeans(24, np.array(train_vect))
          # cluster_centers.columns = count_vect.get_feature_names()
```

6.1.6 [5.1.6] Wordclouds of clusters obtained after applying k-means on AVG W2V SET 3

```
In [190]: # Please write all the code with proper documentation
```

```
In [191]: build_wordclouds(X, labels)
```

not
box
one
like
would
great
order
good
time
product
bag
get
taste
arrived
much
buy
no
well
even
popcorn
could
two
little
fresh
find
ordered
bought
really
eat
price
used
small
flavor
amazon
open
gift
make
bags
came
candy
use
cookies
broken
best
back
think
first
since
plastic
shipping
days
got
found
received
size
put
enough
love
package
boxes
store
purchase
hard
around
several
air
alt
pack
long
day

product
great
excellent
good
gift
quality
time
well
arrived
love
delicious
fresh
tasty
delivery
price
recommend
amazon
service
came
shipping
quick
thank
perfect
order
buy
deal
family
not
fast
best
easy
like
prompt
customer
exactly
expected
tasting
use
looking
packed
timely
highly
taste
satisfied
purchase
packaging
shipped
definitely
chocolate
happy
ordered
cookies
received
would
always

great
oil
taste
good
not
use
little
best
flavor
make
like
add
rice
milk
one
also
better
butter
potatoes
product
used
olive
water
cream
time
no
much
would
even
easy
get
ingredients
tried
added
first
bit
mix
organic
sugar
well
love

sweet
taste
not
like
flavor
good
love
little
really
bit
spicy
hot
would
mint
key
tastes
flavors
gum
lime
best
great
pie
product
time
licorice
hard
tried
nice
delicious
real
strong
chocolate
juice
enough
find
perfect
ginger
artificial
flavored
tart
cicse
had
eve
even
one
enjoy
sweetness
chips
long
much
though
cherry
mustard
food
candy
apple
unique
yummy
lasly
cnap
butter
cream
right
first
actually
absolutely
strawberry
make
tasted
balance
buy
dessert
ingredients
nouth
nice
tried
hard
licorice
time
product
treat
flavorful
price
ear
artificial
sauce
salsa
ketchup
mustard
ice
tea
try
favorite
chardonnay
sweet
taste
not
like
flavor
good
love
little
great
best
lime
gum
enjoy
one
key
mint
would
hot
spicy
bit
really
tastes
flavors
pie
product
licorice
hard
tried
nice
time
first
find
enough
yummy
lasly
unique
chocolate
ginger
artificial
perfect
had
eve
tart
cicse
cnap
butter
cream
right
actually
absolutely
strawberry
make
tasted
balance
buy
dessert
ingredients
nouth
food
candy
apple
mustard
long
much
though
cherry
chips
juice
sauce
salsa
ketchup
ice
tea
try
favorite
chardonnay
delicious
real
strong
flavored
treat

not
like
good
would
taste
one
really
product
great
flavor
time
buy
something
sugar
bad
make
no
much
get
eat
first
price
also
even
love
try
food
better
little
know
tried
chips
flavors
reviews
sure
enough
could
chocolate
best
different
harm
still
anything
thought
bit
box
people
make
give
found
go
bought
cake
many
order
right
real
mix
since
aids
take
tried
never
amazon
product
order
like
certified
also
made
quality
great
no
know
good
buy
price
received
reviews
aware
organic
tomatoes
review
food
would
one
ordered
different
ingredients
products
shipping
company
item
not
cacao
label
arrived
service
customer
even
description
oil
bag
raw
box
best
chips
place
shipped
cans
local
coconut
flavor
back
taste
purchase
package
case
olive
people
well
still
return
read
going
container
refund
another
recommend
reviewer
powder
really

sugar
taste
not
like
water
drink
coconut
tea
product
good
would
use
better
natural
flavor
pineapple
much
best
sweet
coffee
energy
stevia
calories
drinks
great
well
little
also
tried
one
stuff
still
bottle
since
sweetener
powder
drinking
no
cup
packets
without
artificial
even
hot
find
price
juice
make
song
really
feel
brand
add
think
buy
milk
trap
cola
used
tastes
diet
green
makes
using
soda
refreshing
chocolate
cocoa
day
time
flavors

dog
not
food
dogs
cat
one
like
good
great
love
no
would
product
organic
cats
eat
formula
time
healthy
ingredients
still
brand
first
grass
corn
well
tried
get
price
since
high
find
found
free
best
meal
feeding
bag
baby
twice
foods
much
also
amazon
quality
happy
always
back
give
products
starched
grain
north
years
little
think
diet
vet
pet
buy
year
egg
chicken
something
made
really
put
lot
eating
store
brands
know
food
felidae
small
better
problems
loves
recommend
ingredient
calcium
protein
delicious

gluten
free
not
bisquick
good
great
pancakes
product
mix
taste
found
make
flour
biscuits
products
best
tried
wheat
baking
pancake
makes
use
years
family
find
one
many
gf
like
would
made
even
waffles
regular
celiac
dairy
recipe
recipes
eat
also
store
amazon

find
not
stores
product
good
great
could
best
order
taste
available
like
store
buy
hard
years
grocery
happy
time
amazon
local
love
flavor
found
price
delicious
much
chips
get
bought
buying
purchased
ordered
market
made
many
thanks
cinnamon
cereal
snack
favorite
better
locally
glad
ever
even
pack
try
flavors
brand
long
way
always
regular
variety
right
purchase
candy
food
used
first
family
also
able
would
one
no
have
cannot
online

chips
great
not
flavors
bag
good
popchips
bags
snack
love
variety
flavor
taste
salt
calories
pop
get
like
pack
calorie
chip
vinegar
delicious
tasty
try
order
would
amazon
baked
favorite
size
salty
enjoy
healthy
bbc
cnc
single
potato
better
weight
crunch
serving
perfect
kids
eat
crunchy
regular
rinc

coffee
not
good
taste
decaf
cup
one
smooth
flavor
like
great
strong
bitter
blend
best
water
rich
roast
dark
would
get
nice
cups
bold
drink
melitta
even
love
k
no
little
used
much
mountain
french
beans
bag
black
try
aftertaste
full
regular
better
delicious
medium
fine
grind
starbucks
tasted
tried
quite
lime
burnt
price
open
definitely
well
product
coffees
use
buy
also
taste
found
recommend
riviera
grand
warm
day
aroma
pretty
cooked
fresh
sodas
amazing
enjoyed
far
more
make
flavors
decaffeinated
medium
finc

cocoa
good
cups
hot
chocolate
not
taste
great
cup
like
best
keurig
coffee
price
flavor
would
one
product
square
tried
milk
dark
grove
brand
love
water
much
better
watery
add
delicious
make
purchase
far
try
peppermint
gel
really
little
found
swiss
use
pack
buy
sugar
order
perfect
brands
recommend
miss
favorite
find
tastes
setting
creamy
aftertaste

great
not
good
fat
snack
low
calorie
healthy
like
little
diet
eat
taste
salt
calories
flavor
free
chips
snacks
delicious
sugar
tasty
love
high
crunch
crackers
one
product
light
also
excellent
sodium
gluten
weight
foods
fiber
even
carb
crispy
kids
eating
fiber

sauce
not
good
best
great
chicken
hot
flavor
taste
like
product
love
sweet
favorite
soup
beef
seasoning
perfect
one
many
use
made
make
bottle
spicy
jerky
wonderful
recommend
fish
buy
years
try
ever
time
really
find
rice
amazon
salad
cheese
unique
meat
tried
excellent
authentic
mustard
tasted
brand
price
go
much
nice
everything
jerky

chips
potato
kettle
salt
flavor
taste
bag
great
good
flavors
chip
would
like
vinegar
brand
also
love
best
sweet
snack
salty
onion
favorite
pepper
texture
spicy
crunchy
bags
crunch
cheddar
food
recommend
brands
natural
baked
healthier
healthy
better
tasty
really
without
time
much
cream
purchase
eat
try
no
bbq
oil
tried
cooked
could
nice
hard
eating
sweet
size
plain
visit
well
definitely
product
mary
little
greasy
right
want
enough
used
delicious
gel
enjoy
even
sea
thai
ona
soul
ever
honey
cought
strong
find bit
recipe

tea
not
great
green
like
good
flavor
taste
black
teas
sweet
loose
iced
water
much
use
best
would
bags
one
drink
tried
coffee
love
no
hot
bag
time
get
find
leaves
drinking
quality
recommend
white
little
product
tastes
day
cup
since
well
wonderful
powder
used
strong
first
better
try
sugar
also
makes
bought
years
perfect
high
organic
price

coffee
not
good
one
like
product
flavor
taste
great
price
best
cup
cups
tried
amazon
buy
get
brands
love
pods
coffees
much
different
ordered
well
every
flavors
order
blend
excellent
would
better
chocolate
purchase
tasted
dark
many
could
use
back
ever
rich
tea
recommend
tasting
highly
give
store
favorite
brand
find
also
grt
park
decaf
bad
delicious
new
house
smooth
really
box
round
gonna
since
however
bought
roast
purchased
local
drink
years
energy
day
flavored
coffee

chocolate
cookies
not
like
cookie
taste
oreo
great
good
one
peanut
butter
small
would
really
love
flavor
best
mini
dark
chip
oreos
popcorn
white
think
snack
cakesters
eating
try
eat
even
thin
better
want
bars
nabisco
delicious
calories

price
product
good
amazon
store
not
great
really
shipping
would
find
local
box
free
buy
get
taste
much
like
food
grocery
order
chips
deal
save
stuff
subscribe
time
bag
love
item
better
regular
found
cheaper
way
quality
cost
bags
high
use
online
less
expensive
even
pay
still
per
reasonable
purchase
excellent

food
dog
great
not
loves
love
good
newman
dogs
would
old
natural
healthy
organic
foods
quality
amazon
years
like
cat
bag
one
much
also
best
brand
dry
happy
no
year
product
price
cats
eat
stomach
harmony
treats
recommendations
formula
since
problems
issues
allergies
chicken
always
tried
found
started
feed
ingredients
sensitive
expensive
different
decided
balance
senior
bought
better
farms
brands
highly
switched
buy
mix
pancakes
pancake
waffles
waffle
make
made
bisquick
bread
gluten
free
recipe
great
delicious
family
makes
cake
use
box
would
one
flour
time
texture
using
milk
recipes
regular
tried
gf
add
making
oven
syrup
much
found
wheat
butter
fluffy
flavor
like
biscuits
taste
used
little
really
product
best
better
mixes
never
chicken
could
pizza
easy
unbaked
wonderful
years
time
buy
light
sweet
cake
bake
love
pretty
ingredients
reviews
products
equipment
tasty
butter
milk
water
time
not

6.1.7 [5.1.7] Applying K-Means Clustering on TFIDF W2V, SET 4

```
In [192]: # Please write all the code with proper documentation
```

```
In [193]: count_vect = apply_vectorizers_train_test('K-Means','TF-IDF W2V')
```

```
100%|| 50000/50000 [10:28<00:00, 79.56it/s]
```

```
'train_vect' and 'test_vect' are the pickle files.
```

```
In [194]: train_vect = frompicklefile('train_vect')
          print(np.array(train_vect).shape)
```

```
(50000, 50)
```

```
In [195]: clusters = [i for i in range(3,30,3)]
          inertias = apply_kmeans(np.array(train_vect), clusters)
```

```
inertias:  [802692.1832587925, 685241.5240394964, 621654.1069478836, 588293.6762042007, 566068
```

```
In [196]: plot_scores_vs_clusters(clusters, inertias)
```

```
In [197]: labels = retrain_kmeans(24, np.array(train_vect))
          # cluster_centers.columns = count_vect.get_feature_names()
```

6.1.8 [5.1.8] Wordclouds of clusters obtained after applying k-means on TFIDF W2V SET 4

```
In [198]: # Please write all the code with proper documentation
```

```
In [199]: build_wordclouds(X, labels)
```

not
cookie
like
cookies
chocolate
oreo
cakesters
flavor
great
thin
good
taste
one
butter
delicious
snack
oreos
nabiscocalories
chips
would
mini
calories
peanut
chip
favorite
try
sweet
fat
get
ate
really
crisps
chewy
box
make
even
wafer
found
bought
back
find
breakfast
amazon
hard
cream
store
money
love
bars
know
give
better
since
still
free
whole
best
time
small
much
little
two
packs

flavor
hot
pie
love
best
great
taste
like
not
good
sweet
gum
little
tried
time
vanilla
sweetness
chocolate
product
mint
nice
flavors
bit
buy
try
one
limes
really
spicy
tastes
chips
tasted
favorite
hard
real
key
much
ever
yummy
delicious
enjoy
would
ginger
pepper
salt
run
long
strawberry
first
strong
chip
without
honey
though
make
use
cream
better
wonderful
spirit
ketchup
apple
tasty
also
cooker
cyan
find
get
made
could
actually
amazing
quite
found
thing
tic

great
best
not
like
product
good
taste
flavor
would
also
love
time
one
get
no
little
licorice
tried
use
really
amazon
fresh
better
quality
made
even
buy
recommend
make
many
chocolate
well
find
different
several
delicious
enough
always
company
ordered
found
excellent
coffee
tasted
real
favorite
could
back
think
looking
etc
products
price
years
since
flavors
purchase
sugar
organic
nuts
much
food
bought
enjoy
perfect
stuff
things
used
anything
way
keep
first
wonderful
try
old
c11
without
order
small
board
highly
still
bar
taste

not
great
good
like
eat
taste
love
snack
tasty
one
flavor
healthy
little
product
kids
also
no
really
would
find
free
easy
make
delicious
perfect
whole
gluten
calories
calorie
snacks
treat
fat
fruit
crackers
tried
oatmeal
add
think
food
go
low
pasta
cheese
chocolate
recommend
best
much
even
bars
bit
butter
sweet
something
buy
nuts
sugar
price
years
flavors
purchase
organic
chocolate
perfect
stuff
things
used
anything
way
keep
first
wonderful
try
old
c11
without
order
small
board
highly
still
bar
taste

great
good
loved
not
gift
ordered
product
christmas
would
well
time
perfect
fresh
box
candy
arrived
liked
get
order
love
one
came
price
loves
really
enjoyed
excellent
happy
amazon
family
flavor
delicious
friend
find
day
year
first
child
two
buy
got
hot
nice
cute
right
party
chocolate
daughter
little
always
bought
try

not
like
taste
good
great
coffee
flavor
drink
one
better
energy
water
would
product
sugar
tea
much
little
tastes
use
price
buy
give
best
really
get
make
also
cup
sweet
hot
tried
love
even
bad
time
try
no
well
without
stuff
nice
pretty
regular
different
drinking
bitter
first
still
using
found
chocolate
natural
since
think

amazon
product
price
not
good
order
shipped
store
service
best
tomatoes
received
customer
time
item
certified
aware
arrived
free
would
like
box
great
shipping
buy
get
ordered
excellent
cacao
quality
fast
purchase
bag
local
fresh
company
another
different
said
recommend
reviews
seller
better
way
however
package
presents
buying
shipment
delivery
thank
food
stuff
alive
thanks
label
save
bought
love
also
really
organic
brand
take
still
many
high
even
much
deal
drop
came
less
case
case

sugar
corn
like
taste
great
product
not
amazon
would
syrup
powder
oil
no
low
fat
one
food
use
good
free
also
ingredients
high
flavor
even
artificial
products
chocolate
fiber
natural
salt
ingredient
make
ginger
health
carb
little
dist
milk
hydrogenated
sodium
flour
much
better
protein
taking
actually
calcium
people
using
lis.
thing
sweetener
brown
buy
found
msg
price
per
water
agave
could
add
lot
time
calories
fructose
get
used
put
please
need
bac
tried
way
package
read
trans
sweet
sweetener
artificial

would
product
not
like
good
one
box
great
time
bag
taste
even
price
order
use
make
flavor
also
bags
open
amazon
package
plastic
two
try
first
no
get
buy
really
little
could
much
way
think
well
money
fresh
made
enough
put
find
store
better
bought
easy
small
had
lake
tried
food
used
love
sale
size
packaging
boxes
opened
different
received
however
ordered
nappy
water
need
along
thing
going
came
nicolas:
pack
back
thoughe
say
still
know
got
round
size
want
less
best
size
local
quality
variety
buy
local
product
taste
price
store
good
one
like
flavor
no
order
hard
would
chips
time
best
coffee
really
much
could
get
found
grocery
available
delicious
market
cheaper
regular
case
reasonable
online
pack
free
flavors
locally
years
use
ordered
every
way
try
last
since
well
name
per
shipping
happy
cereal
make
expensive
tasteless
worth
brands
box
tried
brand
better
long
purchase
lot
case
buying
favorite
nice
bite
bottle
bulk
sure
candy
amazing
reviews
water
hard
food
used
love
amazon
package
sure
tried
lake
had
small
plastic
two
easy
still
better
bought
store
thoughe
say
know
got
round
size
want
less
best
size

chips
not
great
flavor
salt
love
bag
flavors
good
potato
snack
taste
like
bags
variety
popchips
calorics
vinegar
pepper
would
eat
low
chip
fat
really
better
pack
get
amazon
buy
order
one
try
delicious
tasty
crispy
pop
size
product
kettle
little
serving
without
healthy
eating
crunch
hhq
tried
enjoy

coffee
decaf
cup
strong
k
would
not
taste
great
smooth
flavor
water
starbucks
really
tried
drink
rich
best
one
good
like
roast
bitter
blend
dark
bold
aroma
full
cups
bag
no
make
ground
price
even
tound
house
melitta
tastes
delicious
favorite
regular
buy
without
caribou
trench
better
love
definitely
much
little
pods
mountain
made
get
green
time
coffees
grinc
black
tona
find
try
use
bears
think
day
warning
easy
know
also
enjoy

great
cocoa
good
milk
water
cup
taste
like
cups
best
sugar
hot
not
keurig
square
tried
flavor
chocolate
dark
product
make
coffee
better
would
tastes
brand

gluten
regular
love
flour
dict
mix
make
bisquick
free
like
many
amazon
goodo
waffles
found
dairy
best
taste
would
baking
pancakes
made
products
product
great
wheat
box

tea
not
like
good
great
flavor
taste
green
water
drink
love
teas
iced
sweet
one
tried
best
find
use
would
product
loose
bags
cup
get
hot
black
tastes
licorice
well
really
delicious
recommend

tea
green
not
like
flavor
coffee
drink
iced
much
teas
water
one
leaves
bags
sugar
bag
black
taste
good
would
great
get
tried
loose
powder
strong
use
white
drinking
high
sweet
time
no
made
years
day
find
make
little
hot
makes
try
quality

chips
not
salt
kettle
potato
flavor
brand
bag
like
vinegar
good
love
chip
bags
great
taste
would
no
also
much
spicy
flavors
best
baked
crunch
crunchy
sea
sally
delicious
popchips
product
amazon
recommend
onion
natural
sweet
sour
get
pop
variety
enjoy
cheddar
pepper
plain
ordered
snack
brands
bought
price
pack
lays
made
healthy
trying
regular
thai
bit
enough
want
size
cream
buy
tangy
better
quality
hard
found
case
easy
make
log
opened
big
sun
flavour
well
cheese
jalapeño
spring
cajun
chili
little
try
tortilla
cracker
cheesy
flavor
ile
pu
chase
box
ordered
plain
coating
healthier
lite
dough
price
healthy
trying
made
pack
lays
pop
food
enjoy
enough
want
size
variety
delicious
sour
get
sweet
bit
thai
regular
popchips
best
flavors
spicy
much
vinegar
onion
natural
amazon
recommend
product
also
ordered
plain
good
love
chip
great
taste
would
no
bags
sea
sally
like
baked
crunch
crunchy
kettle
potato
flavor
brand
bag
chips
not
salt

dog
food
not
newman
dogs
greats
good
like
cats
would
no
much
cat
love
old
foods
bag
loves
best
organic
natural
amazon
eat
also
one
healthy
year
formula
better
feed
products
still
issues
product
ingredients
corn
chicken
three
quality
lot
nix
years
every
first
tried
brands
problems
enough
happy
really
well
buy
make
dry
expensive
small
since
treats
time
stomach
farms
store
allergy
recommend
brand
get
try
play
bonus
balance
found
always
buying
purchasing
switched
harmony
order
meat
find
know
two
easy
feeding
never
almost
go
meat
find
size
aaring

not
pancakes
mix
gluten
free
good
make
great
like
pancake
bisquick
made
waffles
box
product
cake
gf
recipe
biscuits
bread
waffle
time
would
tasty
also
mixed
store
butter
pizza
chicken
recipes
easy
tried
best
one
try
makes
recommend
whole
light
wheat
flavor
amazon
celiac
using
could
much
taste
use
price
years
little
well
add
find
never
water
flour
really
even
buy
may
product
family
baking
delicious
love
milk
think
tutty
wonderful
mixes
used
make
making
texture
cake
time
frosting
found
tasty
nasty
perfect
worth
mix
flavor
amazon
wheat
butter
water
never
well
add
find
little
years
price
even
really
product
use
would
tried
best
one
wheat
flavor
amazon
celiac
using
could
much
taste
butter
pizza
chicken
recipes
love
easy

sauce
best
great
chicken
good
not
taste
flavor
like
use
rice
soup
make
one
love
hot
sweet
add
salad
also
pasta
oil
chowder
seasoning
canned
product
potatoes
meat
brand
easy
clams
find
cheese
better
amazon
first
would
bit
favorite
fish
clam
garlic
tasty
meal
mix
ever
tried
fresh
wonderful
delicious
soups
makes
pasta
little
heat
water
spicy
years
rice
tomato
cook
cooking
sausage
out
found
beef

pineapple
not
juice
water
coconut
great
better
oil
coco
also
would
taste
product
flavor
best
olive
good
like
drink
no
milk
makes
natural
sweet
little
add
drinking
buy
stuff
tastes
bottle
much
splash
tea
still
well
sugar
one
oz
gel
use
refreshing
really
drinks
brands
found
price
want
another
healthy
potassium

6.2 [5.2] Agglomerative Clustering

6.2.1 [5.2.1] Applying Agglomerative Clustering on AVG W2V, SET 3

```
In [200]: # Please write all the code with proper documentation
```

```
In [241]: count_vect = apply_vectorizers_train_test('Agglomerative','AvgW2V')
```

```
100%|| 5000/5000 [00:03<00:00, 1580.67it/s]
100%|| 5000/5000 [00:01<00:00, 2515.56it/s]
100%|| 5000/5000 [00:09<00:00, 563.62it/s]

'train_vect' and 'test_vect' are the pickle files.
```

```
In [242]: train_vect = frompicklefile('train_vect')
          print(np.array(train_vect).shape)
```

```
(5000, 50)
```

```
In [243]: labels = apply_agglomerative(9, np.array(train_vect))
```

6.2.2 [5.2.2] Wordclouds of clusters obtained after applying Agglomerative Clustering on AVG W2V SET 3

```
In [244]: # Please write all the code with proper documentation
```

```
In [245]: X = frompicklefile('train_data')
          build_wordclouds(X, labels)
```

taste
really
love
also
great
coffee
tea
chocolate
better
flavor
like
not
good
drink
product
hot
one
no
get
little
try
use
would
much
best
water
sweet
price
tried
nice
green
cup
add
ever
strong
make
decaf
dark
buy
first
excellent
delicious
find
well
sugar
bit
full
wonderful
think
cookies
without
tastes
lime
bitter
favorite
used
pretty
recommend
tasting
milk
coconut
hard
smooth
sauce
stevia
taste
tasted
bland
enough
different
definitely
tasty
version
thanksgiving

gluten
free
bisquick
pancakes
mix
not
good
great
tried
flour
make
would
biscuits
like
products
amazon
love
made
find
found
product
regular
best
wheat
even
baking
box
celiac
disease
makes
family
waffles
pancake
loved
gf
easy
time
table
recipes
expensive
eating
yeast
things
first
recipe
diagnosed
thank
diet
chicken
fluffy
buy
one
could
local
bread
foods
better
store
version
thanksgiving
recommend
little
many
way
old
purchase
flavor
really
ingredients
loves
kids
eat
use
tastes
milk
dairy
wheat
gluten
version

grocery
amazon
good
find
buy
love
not
great
store
product
price
local
would
could
shipping
order
chips
stores
taste
item
pack
hard
fresh
food
time
case
less
bag
bags
per
no
much
found
bough
way
buying
best
regular
available
cereal
excellent
flavors
small
tastes
one
get
cheaper
perfect
first
like
really
try
right
loves
candy
diarrhea
shipped
variety
cookies
ordered
online
different

not
one
good
great
like
taste
would
product
no
use
could
cookies
without
fat
even
love
tried
time
better
try
buy
really
bread
many
years
well
fresh
flavor
favorite
first
need
mix
go
keep
nice
bit
get
way
add
easy
water
best
used
made
make
find
little
also
tea
much
two
chocolate
calories
half
enough
big
sugar
store
luv
milk
price
delicious
recommend
almond
found
back
coffee
candy

not
great
mix
pancakes
like
sugar
free
good
product
gluten
waffles
love
make
made
would
use
pancake
taste
amazon
gf
much
one
best
waffle
flour
really
easy
makes
delicious
syrup
bisquick
ingredients
price
better
family
using

great
not
chips
bag
flavor
like
salt
potato
kettle
chip
bags
love
good
taste
would
snack
much
vinegar
popchips
salty
flavors
really
get
buy
best
delicious
better
spicy
brand
baked
healthy
crunchy
amazon
variety
pepper

6.2.3 [5.2.3] Applying Agglomerative Clustering on TFIDF W2V, SET 4

```
In [246]: # Please write all the code with proper documentation
```

```
In [247]: count_vect = apply_vectorizers_train_test('Agglomerative','TF-IDF W2V')
```

```
100%|| 5000/5000 [00:42<00:00, 111.90it/s]
```

```
'train_vect' and 'test_vect' are the pickle files.
```

```
In [248]: train_vect = frompicklefile('train_vect')
          print(np.array(train_vect).shape)
```

```
(5000, 50)
```

```
In [249]: labels = apply_agglomerative(9, np.array(train_vect))
```

6.2.4 [5.2.4] Wordclouds of clusters obtained after applying Agglomerative Clustering on TFIDF W2V SET 4

```
In [250]: # Please write all the code with proper documentation
```

```
In [251]: X = frompicklefile('train_data')
          build_wordclouds(X, labels)
```

one
amazon
food
not
like
good
product
great
find
price
would
store
also
love
could
time
buy
box
eat
taste
no
dog
well
much
get
bag
little
best
ordered
shipping
small
use
old
think
easy
hard
dogs
organic
order
make
local
found
better
first
however

food
dog
great
cat
amazon
organic
foods
much
old
good
not
dogs
like
love
eating
really
no
years
product
cats
one
best
loves
newman
bag
would
eat
quality
little
healthy
always
price
time
find
feeding
brand
well
first
day
tried
shipping
ingredients
year
diet
baby
chicken
dryscience
weeks
feed
give
take
know
without
starter
getting

great
hot
chocolate
water
best
like
not
good
coffee
taste
flavor
tried
love
would
drink
product
one
cup
buy
get
little
found
tea
sweet
try
use
really
cookies
tastes
strong
definitely
brand

make
bisquick
product
good
mix
gluten
flour
buy
wheat
amazon
biscuits
like
waffle
not
free
sugar
makes
one
great
use
pancakes
waffles
pancake
made
would
best
box

coffee
tea
not
hot
taste
like
chocolate
cup
flavor
great
good
cocoa
decaf
green
milk
water
dark
love
sweet
better
drink
tried
one
use
get
little
k
best
would
cups
recommend
product
find
try
kcurig
hark
found
sugar
much
even
first
price
highly
brand
make
smooth
pomegranate
hag
nikas
tar
tasty
favorite
superfun
price
get
enjoy
want
taste
strong
taste
milk
even
since
could
years
store
cream
n'
thour
cuality
many
regular
drinking
nice
flavors
definitely
excellent
wonde
full
yummy
leaves
tags
bitter
time
actually
tasting
something
bitter
well
water
ingredients
bcx
since
treat
used
say
order
variety
hage
brand
food
calorie
popcorn
makes
sweet
snack
pack
regular
tasty
quite
light
eating
thing
bag
mix
use
find
get
also
even
no
snack
make
buy
product
little
bag
could
really
time
eat
price
sugar
fat
tried
much
not
great
taste
good
flavor
like
best
better
favorite
delicious
well
excellent
cookies
flavors
amazon
calories
tasty
use
mix

not
taste
great
like
good
flavor
love
price
drink
also
tea
time
no
use
tried
could
chocolate
sweet
much
used
make
best
would
little
coffee
buy
nice
one
better
way
well
get
really
product
delicious
found
without
ever
find
cookies

potato
chips
salt
brand
like
good
taste
bag
bags
flavor
great
not
kettle
love
flavors
would
regular
spicy
chip
vinegar
healthy
bought
little

6.3 [5.3] DBSCAN Clustering

6.3.1 [5.3.1] Applying DBSCAN on AVG W2V, SET 3

```
In [212]: # Please write all the code with proper documentation
```

```
In [45]: count_vect = apply_vectorizers_train_test('DBSCAN','AvgW2V')
```

```
100%|| 5000/5000 [00:02<00:00, 1799.36it/s]
100%|| 5000/5000 [00:01<00:00, 2900.54it/s]
100%|| 5000/5000 [00:08<00:00, 580.04it/s]

'train_vect' and 'test_vect' are the pickle files.
```

```
In [46]: train_vect = frompicklefile('train_vect')
         print(np.array(train_vect).shape)
```

```
(5000, 50)
```

```
In [53]: k_dist_plot(100,train_vect,np.array(train_vect).shape[0])
```

```
In [55]: labels = apply_dbscan(0.8,2*(np.array(train_vect).shape[1]),np.array(train_vect))

         noisy_pts = np.count_nonzero(labels == -1)
         print('% noisy points: ', str((noisy_pts/len(labels))*100)+'%')
```

```
% noisy points:  0.24%
```

6.3.2 [5.3.2] Wordclouds of clusters obtained after applying DBSCAN on AVG W2V SET 3

```
In [56]: # Please write all the code with proper documentation
```

```
In [57]: X = frompicklefile('train_data')
         build_wordclouds(X, labels)
```

6.3.3 [5.3.3] Applying DBSCAN on TFIDF W2V, SET 4

```
In [58]: # Please write all the code with proper documentation
```

```
In [60]: count_vect = apply_vectorizers_train_test('DBSCAN','TF-IDF W2V')
```

```
100%|| 5000/5000 [00:41<00:00, 121.48it/s]
```

```
'train_vect' and 'test_vect' are the pickle files.
```

```
In [61]: train_vect = frompicklefile('train_vect')
         print(np.array(train_vect).shape)
```

```
(5000, 50)
```

```
In [62]: k_dist_plot(100,train_vect,np.array(train_vect).shape[0])
```

```
In [70]: labels3 = apply_dbscan(0.8,2*(np.array(train_vect).shape[1]),np.array(train_vect))

         noisy_pts = np.count_nonzero(labels3 == -1)
         print('% noisy points: ', str((noisy_pts/len(labels3))*100)+'%')
```

```
% noisy points:  1.2%
```

6.3.4 [5.3.4] Wordclouds of clusters obtained after applying DBSCAN on TFIDF W2V SET 4

```
In [71]: # Please write all the code with proper documentation
```

```
In [72]: X = frompicklefile('train_data')
         build_wordclouds(X, labels3)
```

7 [6] Conclusions

```
In [69]: # Please compare all your models using Prettytable library.
         # You can have 3 tables, one each for kmeans, agllomerative and dbscan
```

7.1 [6.1] Observations:

K-Means:

1. Data points: 50,000 points were considered for K-Means.

2. Hyperparameter in K-Means:

```
a) Number of Clusters: The numbers ranging from 3 to 30 with interval of 3 were considere
b) Optimal cluster number obtained in K-Means is around 20-25 for different models.
```

3. Wordclouds:

```
a) WC1: This cluster contains words related to dog/cat food. Ex: dog, cat, food, love, fe

b) WC2: This cluster contains words related to oreo/cookies or chocolates, chips. Ex: coo

c) WC3: This cluster contains similar words related to the reviews of the products relate

d) WC4: This cluster is similar to the WC3

f) WC5: This cluster is related to the buying products/gifting them for holiday season in

g) WC6: This cluster is also similar to amazon products and shipmet related words.

h) WC7: This cluster is related to the ingredients in the food products. Ex: Sigar, corn,

i) WC8: This cluster is similar to the WC3

j) WC9: This cluster contains similar words related to Chips. Ex: chips, salty, potato, b

h) WC10: This cluster contains similar words related to beverages to tea/coffee. Ex: tea,

i) WC11: This cluster is similar to the WC10

j) WC12: This cluster contains similar words related to pancakes, waffles, biscuits and e

k) WC13: This cluster is similar to the WC10

l) WC14: This cluster contains similar words related to Chips. Ex: chips, salty, potato,

m) WC15: This cluster is similar to the WC1

n) WC16: This cluster is similar to the WC12

0) WC17: This cluster contains similar words related to sauce/soup. Ex: tea, sauce, soup,
```

Agglomerative Clustering:

1. Data points: 5,00 points were considered for Agglomerative Clustering.
2. No. of Clusters: 9 clusters is considered for n_clusters as the data points are low in number compared to K-Means.

3. Wordclouds:

```
a) WC1: This cluster contains similar tastes of food products on Amazon. Ex: amazon, fooc

b) WC2: This cluster contains words related to dog/cat food. Ex: dog, cat, food, love, fe

c) WC3: This cluster contains similar words related to beverages to tea/coffee. Ex: tea,

d) WC4: This cluster contains similar words related to pancakes, waffles, biscuits and et

e) WC5: This cluster is very similar like WC3.

f) WC6: This cluster contains similar words related to the reviews of the products relate

g) WC7: This cluster is very similar to WC4.

h) WC8: This cluster contains similar words related to Chips. Ex: chips, salty, potato, b

i) his cluster is very similar like WC3.
```

DBSCAN:

1) Hyperparameter in DBSCAN:

```
a) Number of Clusters: 2* dimensionality of the vector is chosen as a rule of thumb
b) eps: Values like 0.1, 0.5, 1.0 were considered for the dataset. % noisy points decreas
```

2) Wordclouds:

```
Cluster1: This cluster groups the similar words related products like Coffee, tea, sugar

Cluster2: This cluster groups the words related to food ingredients(cocoa, milk, chocolat
```