Amazon review

Objective :

Given a review determine wheather the review is positive (4 or 5) or negative (1 or 2) ?

we can achieve this by using score(an atribute in dataset). A score rating of 4 or 5 indicates a positive review and a score of 1 or 2 indicates a negative review. rating of 3 is neutral and is ignored.

The dataset contains two files 1 review.csv(sample of 10 points that repliates the database) 2 database.sqliteThe database contains columns as id,productId,UserId,ProfileName,HelpfulnessNumerator,HelpfulnessDenominator,Score,Time(UNIX/epoch),Summary,Text.1. we will process the data i.e., deduplication and text processing and sort the data according to productId, Time 2. Now we will take the random sample of 10k points , and will use simle cross validation to find the optimal k- value. 3. Now , we apply different text processing techniques like BOW , TF-IDF , W2V , TF-IDF W2V . and for each of the technique we will aplly the k-nn

In [62]:

```
#......loading the data...........
import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sbn
from sklearn import metrics
from sklearn.metrics import f1_score
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.metrics import roc_curve,auc
from nltk.stem.porter import PorterStemmer
#....for string
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
import re

#.....using sqlite for connecting and reading the data.......
con = sqlite3.connect('../input/amazon-fine-food-reviews/database.sqlite')

#.....filtering only positive reviews and negative reviews and ignoring reviews with rating =
3......
filtered_data = pd.read_sql_query('''Select * From Reviews Where Score!=3''',con)

#.....labeling reviews with rating>3 a positive reviews and reivews with rating<3 as negative u
sing the method "partition".......
def partition(x):
    if x<3:
        return'negative'
    return'positive'

#.....now changing the numerical values into positive or negative ratings....
actualscore = filtered_data['Score']
positivenegative = actualscore.map(partition)
filtered_data['Score'] = positivenegative
filtered_data.shape
```

Out[62]:

```
(525814, 10)
```

DATA DEDUPLICATION:

In [63]:

```
#...sorting the data according to productId in Ascending order
sorted_data = filtered_data.sort_values('ProductId',axis=0,ascending=True,inplace=False,kind='q
uicksort',na_position='last')
```

In [64]:

```
#...deduplication of entries
final = sorted_data.drop_duplicates(subset ={"UserId","ProfileName","Time","Text"},keep='first'
,inplace=False)
final.shape
```

Out[64]:

```
(364173, 10)
```

In [65]:

```
#...makin sure that HelpfulnessDenominator is greater than HelpfulnessNumerator
final=final[final.HelpfulnessNumerator <= final.HelpfulnessDenominator]
final['Score'].value_counts()
```

Out[65]:

```
positive    307061
negative     57110
Name: Score, dtype: int64
```

TEXT Preprocessing :

In [66]:

```
#...for removing stopwords and initializing snowball stemmer

stop = set(stopwords.words('english'))
snow = nltk.stem.SnowballStemmer('english')
```

In [67]:

```
#.... defining funtions for cleaning the documents from html tags,punctuations and replacing th
em with white spaces

def cleanhtml(sentence):
    cleanr = re.compile('<.*?')
    cleantext = re.sub(cleanr,"",sentence)
    return cleantext

def cleanpunc(sentence):
    cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
    cleaned = re.sub(r'[.|,|)|(|\|/]',r'',cleaned)
    return cleaned
print(stop)
print('*******')
print(snow.stem('tasty'))
```

```
{"you'd", "hadn't", 'under', 'having', 'those', 'her', 'these', 'yourselves', "is
n't", 'whom', 'does', 'only', 'few', 'is', 'was', 'my', 'through', 'you', 'do', 'o
r', 'on', 'our', 'such', 'itself', 'ma', "mustn't", 'in', 'further', "shan't", 'hi
s', 'd', "don't", 'so', "didn't", 'off', 'yours', 'its', "that'll", 'than', 'are',
"aren't", 'being', 'won', 'am', 'ain', 'nor', 'isn', 'themselves', 'most', 'too',
'me', 'if', 'shouldn', 've', 'to', 'no', 'has', 'did', 'some', 'didn', 'own', 'you
r', 'been', "you'll", 'and', 'doesn', 'but', 'while', 'other', 'very', 'he', 'do
n', "haven't", 'any', 'who', 'needn', 'a', 'of', 'were', 'weren', 'this', 'befor
e', 'over', "doesn't", "shouldn't", 'ourselves', 'shan', 'haven', 'then', 'that',
'it', "you're", 're', 'him', 'both', 'm', 'mightn', 'from', 'out', 'them', 'hers',
'up', 'will', 'during', 'same', 'i', "it's", 'aren', 'at', 'about', 'had', 'all',
'they', 'against', 'doing', 'because', 'more', 'where', "wouldn't", 'after', 'thei
rs', 'for', 'which', "she's", 'as', 'between', 'can', 'wouldn', "weren't", 'into',
'the', 'there', 'each', 'o', 'wasn', 'herself', 'should', 'not', "should've", 'b
y', 'be', "couldn't", 'why', 'couldn', 'yourself', 'below', 'once', 'himself', 'ag
ain', 'we', "you've", 'have', 'mustn', 't', 'ours', 'when', 'hasn', "won't", 'wha
t', 'above', 'an', 'hadn', 's', 'she', 'll', 'now', 'just', "wasn't", 'their',
'y', 'myself', 'with', "hasn't", 'until', 'down', 'here', "needn't", 'how', "might
n't"}
*******
tasti
```

In [68]:

```
#....now we implement the code according to the steps in the preprocessing

i = 0
str1 = ' '
final_string = []
all_positive_words = []
all_negative_words = []
s = ''

#....with use of sent we are  processing each review

for sent in final['Text'].values:
    filtered_sentence = []
    sent = cleanhtml(sent)
    for w in sent.split():
        for cleaned_words in cleanpunc(w).split(): #....directly processing after cleanpun(w)
            if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                if(cleaned_words.lower() not in stop):
                    s=(snow.stem(cleaned_words.lower())).encode('utf8')
                    filtered_sentence.append(s)
                    if(final['Score'].values)[i] == 'positive':
                        all_positive_words.append(s)
                    if(final['Score'].values)[i] == 'negative':
                        all_negative_words.append(s)
                else:
                    continue
            else:
                continue

    str1 = b" ".join(filtered_sentence)

    final_string.append(str1)

    i+=1
```

In [69]:

```
final['CleanedText'] = final_string
final['CleanedText'] = final['CleanedText'].str.decode('utf-8')
final.head()
type(final)
```

Out[69]:

```
pandas.core.frame.DataFrame
```

In [70]:

```
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)
```

In [71]:

```
#....dataframe "refined_data" is our refined dataset
con = sqlite3.connect('final.sqlite')
refined_data = pd.read_sql_query("select * from Reviews" ,con)
refined_data.head(3)
```

Out[71]:

	index	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Help
0	138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0
1	138688	150506	0006641040	A2IW4PEEKO2R0U	Tracy	1	1
2	138689	150507	0006641040	A1S4A3IQ2MU7V4	sally sue "sally sue"	1	1

In [72]:

```
#....sorting the refined_data according to time,first we will convrt time into normal form
refined_data["Time"] = pd.to_datetime(refined_data["Time"] , unit = "s")
#refined_data["Time"][2]
#.....now, we will sort the data according to time.
refined_data = refined_data.sort_values('Time')
refined_data.head(2)
```

Out[72]:

	index	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Help
0	138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0
30	138683	150501	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	2	2

In [73]:

```
#....now , we will define our x and y values
X = refined_data['CleanedText'].sample(10000)
Y = refined_data['Score'].sample(10000)
print(X.head(2))
print(Y.head(2))
```

```
329201    think puroast pretti good job coffe coffe smal...
351061    tri coco samba first time friend love huge fan...
Name: CleanedText, dtype: object
196459    positive
125974    positive
Name: Score, dtype: object
```

BOW :

(Brute Force)

In [74]:

```
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
from sklearn import model_selection
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

In [75]:

```
from sklearn.feature_extraction.text import CountVectorizer

bow = CountVectorizer()
brute_X = bow.fit_transform(X)
```

In [76]:

```
brute_X.shape
```

Out[76]:

```
(10000, 17196)
```

In [142]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(brute_X, Y, test_size=0.3, random_s
tate=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i)

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.715238
f1 score for i= 3 is 0.793810
f1 score for i= 5 is 0.808571
f1 score for i= 7 is 0.821429
f1 score for i= 9 is 0.828095
f1 score for i= 11 is 0.829048
f1 score for i= 13 is 0.829524
f1 score for i= 15 is 0.830000
f1 score for i= 17 is 0.831429
f1 score for i= 19 is 0.831429
f1 score for i= 21 is 0.831429
f1 score for i= 23 is 0.831429
f1 score for i= 25 is 0.831429
f1 score for i= 27 is 0.831429
f1 score for i= 29 is 0.831429
optimal fscore is 0.831429
```

In [78]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 85%
```

observation :

1. The f1 scores for k-d tree are increasing along with the value of k till k=17 and from then on the value isnt changing.
2. The max f score is obtained when k = [17,19,21,23,25,27,29]

(K-D Tree) :

In [79]:

```
kdtree_bow = CountVectorizer(max_features = 2000,min_df = 50)
kd_X = kdtree_bow.fit_transform(X)
```

In [80]:

```
kd_X = kd_X.toarray()
```

In [146]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(kd_X, Y, test_size=0.3, random_stat
e=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i,algorithm = "kd_tree")

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.683810
f1 score for i= 3 is 0.769524
f1 score for i= 5 is 0.808095
f1 score for i= 7 is 0.819048
f1 score for i= 9 is 0.826667
f1 score for i= 11 is 0.834762
f1 score for i= 13 is 0.835238
f1 score for i= 15 is 0.839048
f1 score for i= 17 is 0.838571
f1 score for i= 19 is 0.840476
f1 score for i= 21 is 0.840476
f1 score for i= 23 is 0.840476
f1 score for i= 25 is 0.840952
f1 score for i= 27 is 0.840952
f1 score for i= 29 is 0.840952
optimal fscore is 0.840952
```

In [82]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 85%
```

observation :

1. The f1 scores for k-d tree are increasing along with the value of k till k=19 and from then on the value isnt changing.
2. The max f score is obtained when k = [19,21,23,25,27,29]

TF-IDF

(Brute Force)

In [83]:

```
#.....these will give final tf-idf tokenized matrix shape for the raw text not the cleaned text
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2))
tf_idf_X = tf_idf_vect.fit_transform(X)
tf_idf_X.get_shape()
```

Out[83]:

```
(10000, 245990)
```

In [143]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(tf_idf_X, Y, test_size=0.3, random_
state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i)

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.740000
f1 score for i= 3 is 0.795714
f1 score for i= 5 is 0.821905
f1 score for i= 7 is 0.835714
f1 score for i= 9 is 0.841905
f1 score for i= 11 is 0.842857
f1 score for i= 13 is 0.843810
f1 score for i= 15 is 0.844286
f1 score for i= 17 is 0.844286
f1 score for i= 19 is 0.844286
f1 score for i= 21 is 0.844286
f1 score for i= 23 is 0.844286
f1 score for i= 25 is 0.844286
f1 score for i= 27 is 0.844286
f1 score for i= 29 is 0.844286
optimal fscore is 0.844286
```

In [144]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 85%
```

observation :

1. The f1 scores for k-d tree are increasing along with the value of k till k=15 and from then on the value isnt changing.
2. The max f score is obtained when k = [15,17,19,21,23,25,27,29]

KD-Tree

In [86]:

```
kd_tree_tf_idf_vect = TfidfVectorizer(ngram_range=(1,2),max_features = 2000,min_df = 50)
kd_tree_tf_idf_X = kd_tree_tf_idf_vect.fit_transform(X)
```

In [87]:

```
kd_tree_tf_idf_X = kd_tree_tf_idf_X.toarray()
```

In [88]:

```
type(kd_tree_tf_idf_X)
```

Out[88]:

```
numpy.ndarray
```

In [147]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(kd_tree_tf_idf_X, Y, test_size=0.3,
 random_state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i,algorithm = "kd_tree")

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.720952
f1 score for i= 3 is 0.772381
f1 score for i= 5 is 0.807619
f1 score for i= 7 is 0.815714
f1 score for i= 9 is 0.819048
f1 score for i= 11 is 0.819524
f1 score for i= 13 is 0.820952
f1 score for i= 15 is 0.822381
f1 score for i= 17 is 0.822857
f1 score for i= 19 is 0.822857
f1 score for i= 21 is 0.822857
f1 score for i= 23 is 0.822857
f1 score for i= 25 is 0.822857
f1 score for i= 27 is 0.822857
f1 score for i= 29 is 0.822857
optimal fscore is 0.822857
```

In [90]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 85%
```

observation :

1. The f1 scores for k-d tree are increasing along with the value of k till k=17 and from then on the value isnt changing.
2. The max f score is obtained when k = [17,19,21,23,25,27,29]

Word 2 Vec :

In [91]:

```
import os
is_your_ram_gt_16g=False
want_to_read_sub_set_of_google_w2v = True
want_to_read_whole_google_w2v = True
if not is_your_ram_gt_16g:
    if want_to_read_sub_set_of_google_w2v and  os.path.isfile('google_w2v_for_amazon.pkl'):
        with open('google_w2v_for_amazon.pkl', 'rb') as f:
            # model is dict object, you can directly access any word vector using model[word]
            model = pickle.load(f)
else:
    if want_to_read_whole_google_w2v and os.path.isfile('GoogleNews-vectors-negative300.bin'):
        model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=
True)
```

In [92]:

```
i=0
list_of_sent=[]
for sent in X.values:
    list_of_sent.append(sent.split())
```

In [94]:

```
# min_count = 5 considers only words that occured atleast 5 times
w2v_model=Word2Vec(list_of_sent,min_count=5,size=50, workers=4)
```

In [160]:

```
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])
```

In [97]:

```
w2v = w2v_model[w2v_model.wv.vocab]
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationWarnin
g: Call to deprecated `__getitem__` (Method will be removed in 4.0.0, use self.wv.
__getitem__() instead).
  """Entry point for launching an IPython kernel.
```

In [161]:

```
w2v.shape
```

Out[161]:

```
(4731, 50)
```

since w2v of repeating words with min count of 5 are 4731 , so we are reducing y items to 4731.

In [149]:

```
w2v_brute_y = Y.sample(4731)
```

In [150]:

```
w2v_brute_y.shape
```

Out[150]:

```
(4731,)
```

Brute Force:

In [152]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(w2v,w2v_brute_y, test_size=0.3, ran
dom_state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i)

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.747485
f1 score for i= 3 is 0.813883
f1 score for i= 5 is 0.824950
f1 score for i= 7 is 0.847082
f1 score for i= 9 is 0.852113
f1 score for i= 11 is 0.852113
f1 score for i= 13 is 0.853119
f1 score for i= 15 is 0.854125
f1 score for i= 17 is 0.854125
f1 score for i= 19 is 0.853119
f1 score for i= 21 is 0.854125
f1 score for i= 23 is 0.854125
f1 score for i= 25 is 0.854125
f1 score for i= 27 is 0.854125
f1 score for i= 29 is 0.854125
optimal fscore is 0.854125
```

In [154]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 84%
```

observation :

1. The f1 scores for k-d tree are random and not following any order .
2. The max f score is obtained when k = [15,17,21,23,25,27,29]

K-D Tree :

In [155]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(w2v, w2v_brute_y, test_size=0.3, ra
ndom_state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i,algorithm = "kd_tree")

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.722334
f1 score for i= 3 is 0.804829
f1 score for i= 5 is 0.818913
f1 score for i= 7 is 0.825956
f1 score for i= 9 is 0.837022
f1 score for i= 11 is 0.839034
f1 score for i= 13 is 0.843058
f1 score for i= 15 is 0.845070
f1 score for i= 17 is 0.846076
f1 score for i= 19 is 0.846076
f1 score for i= 21 is 0.846076
f1 score for i= 23 is 0.846076
f1 score for i= 25 is 0.846076
f1 score for i= 27 is 0.846076
f1 score for i= 29 is 0.846076
optimal fscore is 0.846076
```

In [156]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 84%
```

observation :

1. The f1 scores for k-d tree are increasing along with the value of k till k=17 and from then on the value isnt changing.
2. The max f score is obtained when k = [17,19,21,23,25,27,29]

Avg Word2Vec :

In [109]:

```
#....computing the avaerage of each word 2 vec.
# average Word2Vec
# compute average word2vec for each review.
from tqdm import tqdm
sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sent in tqdm(list_of_sent): # 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
    sent_vectors.append(sent_vec)
print(len(sent_vectors))
print(len(sent_vectors[0]))
```

```
100%|██████████| 10000/10000 [00:14<00:00, 705.50it/s]

10000
50
```

In [170]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(sent_vectors, Y, test_size=0.3, ran
dom_state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i)

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.725714
f1 score for i= 3 is 0.788571
f1 score for i= 5 is 0.812381
f1 score for i= 7 is 0.831429
f1 score for i= 9 is 0.834286
f1 score for i= 11 is 0.837143
f1 score for i= 13 is 0.839048
f1 score for i= 15 is 0.840000
f1 score for i= 17 is 0.840476
f1 score for i= 19 is 0.841429
f1 score for i= 21 is 0.841429
f1 score for i= 23 is 0.841905
f1 score for i= 25 is 0.841905
f1 score for i= 27 is 0.841905
f1 score for i= 29 is 0.841905
optimal fscore is 0.841905
```

In [111]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 85%
```

observation :

1. The f1 scores for brute force are increasing along with the value of k till k=23 and from then on the value isnt changing.
2. The max f score is obtained when k = [23,25,27,29]

(KD-Tree)

In [157]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(sent_vectors, Y, test_size=0.3, ran
dom_state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i,algorithm = "kd_tree")

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.718571
f1 score for i= 3 is 0.782381
f1 score for i= 5 is 0.806667
f1 score for i= 7 is 0.820476
f1 score for i= 9 is 0.827619
f1 score for i= 11 is 0.827619
f1 score for i= 13 is 0.827619
f1 score for i= 15 is 0.828571
f1 score for i= 17 is 0.829048
f1 score for i= 19 is 0.829048
f1 score for i= 21 is 0.829048
f1 score for i= 23 is 0.829048
f1 score for i= 25 is 0.829048
f1 score for i= 27 is 0.829048
f1 score for i= 29 is 0.829048
optimal fscore is 0.829048
```

In [159]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 85%
```

observation :

1. The f1 scores for k-d tree are increasing along with the value of k till k=17 and from then on the value isnt changing.
2. The max f score is obtained when k = [17,19,21,23,25,27,29]

TF-IDF(W2V)

In [114]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
model = TfidfVectorizer()
tf_idf_matrix = model.fit_transform(X.values)
# 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 [115]:

```
# 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 = tfidf

tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored in this list
row=0;
for sent in tqdm(list_of_sent): # 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:
            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
```

```
100%|██████████| 10000/10000 [00:24<00:00, 415.37it/s]
```

(Brute Force) :

In [162]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(tfidf_sent_vectors, Y, test_size=0.
3, random_state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i)

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f scorefor each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.709048
f1 score for i= 3 is 0.776667
f1 score for i= 5 is 0.806667
f1 score for i= 7 is 0.820000
f1 score for i= 9 is 0.828571
f1 score for i= 11 is 0.829524
f1 score for i= 13 is 0.828571
f1 score for i= 15 is 0.829524
f1 score for i= 17 is 0.830952
f1 score for i= 19 is 0.830952
f1 score for i= 21 is 0.830952
f1 score for i= 23 is 0.830952
f1 score for i= 25 is 0.830952
f1 score for i= 27 is 0.830952
f1 score for i= 29 is 0.830952
optimal fscore is 0.830952
```

In [163]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 85%
```

K-D Tree

In [164]:

```
# split the data set into train and test
X_1, X_test, y_1, y_test = model_selection.train_test_split(tfidf_sent_vectors, Y, test_size=0.
3, random_state=0)

# split the train data set into cross validation train and cross validation test
X_tr, X_cv, y_tr, y_cv = model_selection.train_test_split(X_1, y_1, test_size=0.3)

for i in range(1,30,2):
    # instantiate learning model (k = 30)
    knn = KNeighborsClassifier(n_neighbors = i,algorithm = "kd_tree")

    # fitting the model on crossvalidation train
    knn.fit(X_tr, y_tr)

    # predict the response on the crossvalidation train
    pred_classifier = knn.predict(X_cv)

    #calculating f score for each i
    f1 = f1_score(y_cv,pred_classifier,average='micro')
    print("f1 score for i= %i is %2.6f" % (i , f1))


f = max([f1])
print("optimal fscore is %2.6f " %f)
```

```
f1 score for i= 1 is 0.733810
f1 score for i= 3 is 0.782381
f1 score for i= 5 is 0.808095
f1 score for i= 7 is 0.827143
f1 score for i= 9 is 0.827619
f1 score for i= 11 is 0.831429
f1 score for i= 13 is 0.833810
f1 score for i= 15 is 0.834762
f1 score for i= 17 is 0.835238
f1 score for i= 19 is 0.835238
f1 score for i= 21 is 0.835238
f1 score for i= 23 is 0.835238
f1 score for i= 25 is 0.835238
f1 score for i= 27 is 0.835238
f1 score for i= 29 is 0.835238
optimal fscore is 0.835238
```

In [169]:

```
knn = KNeighborsClassifier(29)
knn.fit(X_tr,y_tr)
pred = knn.predict(X_test)
acc = accuracy_score(y_test, pred, normalize=True) * float(100)
print('\n****Test accuracy for k = 29 is %d%%' % (acc))
```

```
****Test accuracy for k = 29 is 85%
```

observation :

1. The f1 scores for both k-d tree are increasing along with the value of k till k=17 and from then on the value isnt changing.