Amazon_Food_Review-k-NN

September 25, 2018

1 K-NN on Different models-(BagOfWords,TfIDF,AvgWord2Vec,TfIDF Weighted Word2Vec)

```
In [1]: %matplotlib inline

        #import all the modules
        import sqlite3
        import numpy as np
        import pandas as pd
        import nltk
        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,accuracy_score
        from sklearn import metrics
        from sklearn.metrics import roc_curve,auc
        from sklearn.manifold import TSNE
        from nltk.corpus import stopwords
        from nltk.stem.porter import PorterStemmer
        from nltk.stem import SnowballStemmer
        from nltk.stem.wordnet import WordNetLemmatizer
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.model_selection import train_test_split
        from sklearn.cross_validation import cross_val_score
        from collections import Counter
        from sklearn import cross_validation
```

```
D:\Anaconda\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module w
  "This module will be removed in 0.20.", DeprecationWarning)
```

```
In [2]: conn=sqlite3.connect('final2.sqlite')
        conn.cursor()
        conn.commit()
        conn.text_factory=str
        #final_data.to_sql('Reviews',conn,schema=None,if_exists='replace')
```

```
In [3]: fd=pd.read_sql_query("""SELECT * FROM REVIEWS""",conn)
```

```
In [4]: fd.head(5)
```

```
Out[4]:     index      Id   ProductId          UserId                    ProfileName  \
        0  138706  150524  0006641040   ACITT7DI6IDDL                shari zychinski   
        1  138688  150506  0006641040  A2IW4PEEKO2R0U                          Tracy   
        2  138689  150507  0006641040  A1S4A3IQ2MU7V4          sally sue "sally sue"   
        3  138690  150508  0006641040     AZGXZ2UUK6X  Catherine Hallberg "(Kate)"   
        4  138691  150509  0006641040  A3CMRKGE0P909G                         Teresa   

           HelpfulnessNumerator  HelpfulnessDenominator        Time  \
        0                     0                       0   939340800   
        1                     1                       1  1194739200   
        2                     1                       1  1191456000   
        3                     1                       1  1076025600   
        4                     3                       4  1018396800   

                                              Summary  \
        0                       EVERY book is educational   
        1  Love the book, miss the hard cover version   
        2                 chicken soup with rice months   
        3      a good swingy rhythm for reading aloud   
        4             A great way to learn the months   

                                                        Text  \
        0  this witty little book makes my son laugh at l...   
        1  I grew up reading these Sendak books, and watc...   
        2  This is a fun way for children to learn their ...   
        3  This is a great little book to read aloud- it ...   
        4  This is a book of poetry about the months of t...   

                                                 CleanedText  
        0  witti littl book make son laugh loud recit car...  
        1  grew read sendak book watch realli rosi movi i...  
        2  fun way children learn month year learn poem t...  
        3  great littl book read nice rhythm well good re...  
        4  book poetri month year goe month cute littl po...  
```

```
In [5]: conn2=sqlite3.connect('final.sqlite')
```

```
In [6]: label_df=pd.read_sql_query("""SELECT * FROM REVIEWS""",conn2)
```

```
In [7]: label_df.head(4)
```

```
Out[7]:     index      Id   ProductId          UserId                    ProfileName  \
        0  138706  150524  0006641040   ACITT7DI6IDDL                shari zychinski   
        1  138688  150506  0006641040  A2IW4PEEKO2R0U                          Tracy   
        2  138689  150507  0006641040  A1S4A3IQ2MU7V4          sally sue "sally sue"   
```

```
3  138690  150508  0006641040     AZGXZ2UUK6X  Catherine Hallberg "(Kate)"

   HelpfulnessNumerator  HelpfulnessDenominator     Score        Time  \
0                     0                       0  Positive   939340800
1                     1                       1  Positive  1194739200
2                     1                       1  Positive  1191456000
3                     1                       1  Positive  1076025600

                                    Summary  \
0                  EVERY book is educational
1  Love the book, miss the hard cover version
2              chicken soup with rice months
3      a good swingy rhythm for reading aloud

                                                Text  \
0  this witty little book makes my son laugh at l...
1  I grew up reading these Sendak books, and watc...
2  This is a fun way for children to learn their ...
3  This is a great little book to read aloud- it ...

                                         CleanedText
0  witti littl book make son laugh loud recit car...
1  grew read sendak book watch realli rosi movi i...
2  fun way children learn month year learn poem t...
3  great littl book read nice rhythm well good re...
```

```
In [8]: label_df=label_df.sort_values('Time',axis=0,inplace=False,kind='quicksort')

In [9]: label_df.shape

Out[9]: (364173, 12)

In [10]: fd=fd.sort_values('Time',axis=0,inplace=False,kind='quicksort')

In [11]: fd.head(3)

Out[11]:       index      Id   ProductId          UserId         ProfileName  \
         0    138706  150524  0006641040  ACITT7DI6IDDL     shari zychinski
         30   138683  150501  0006641040  AJ46FKXOVC7NR  Nicholas A Mesiano
         424  417839  451856  B00004CXX9  AIUWLEQ1ADEG5    Elizabeth Medina

              HelpfulnessNumerator  HelpfulnessDenominator       Time  \
         0                       0                       0  939340800
         30                      2                       2  940809600
         424                     0                       0  944092800

                                                 Summary  \
         0                     EVERY book is educational
         30   This whole series is great way to spend time w...
```

```
424                                  Entertainingl Funny!

                                                  Text  \
0    this witty little book makes my son laugh at l...
30   I can remember seeing the show when it aired o...
424  Beetlejuice is a well written movie ... ever...

                                           CleanedText
0    witti littl book make son laugh loud recit car...
30   rememb see show air televis year ago child sis...
424  beetlejuic well written movi everyth excel act...
```

2 Since we have not much of RAM we are working on select set of samples

```
In [12]: #work on selected samples

         n_samples=2000
         test_data=fd.sample(n_samples)   ##using pd.sample() function
         label_data=label_df['Score'][0:2000]
```

```
In [13]: test_data.shape
```

```
Out[13]: (2000, 11)
```

```
In [14]: label_data.shape
```

```
Out[14]: (2000,)
```

3 Bag Of Words KNN

```
In [15]: #Bag of Words
         count_vect=CountVectorizer()
         final_count=count_vect.fit_transform(test_data['CleanedText'].values)
         print("the type of count vectorizer is:",type(final_count))
         final_count.get_shape()
```

```
the type of count vectorizer is: <class 'scipy.sparse.csr.csr_matrix'>
```

```
Out[15]: (2000, 6910)
```

```
In [16]: #split the data into train and test fo bag of words

         X_train,X_test,Y_train,Y_test=cross_validation.train_test_split(final_count,label_data
         #split train into cross val train and cross val test
         X_t,X_cv,Y_t,Y_cv=cross_validation.train_test_split(X_train,Y_train,test_size=0.3)
```

```
In [17]: #shape of train and test,cv
         X_t.shape

Out[17]: (980, 6910)

In [18]: X_cv.shape

Out[18]: (420, 6910)

In [19]: X_test.shape

Out[19]: (600, 6910)

In [20]: Y_test.shape

Out[20]: (600,)

In [21]: X_train.shape

Out[21]: (1400, 6910)

In [22]: Y_train.shape

Out[22]: (1400,)

In [23]: #find the best k based on cv accuracy for bow

         for i in range(1,10,2):
             # instantiate learning model (k = 10)
             knn = KNeighborsClassifier(n_neighbors=i)

             # fitting the model on crossvalidation train
             knn.fit(X_t, Y_t)

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

             # evaluate CV accuracy
             acc = accuracy_score(Y_cv, pred, normalize=True) * float(100)
             print('\nCV accuracy for k = %d is %d%%' % (i, acc))
         #test accuracy
         knn = KNeighborsClassifier(7)
         knn.fit(X_t,Y_t)
         pred = knn.predict(X_test)
         acc = accuracy_score(Y_test, pred, normalize=True) * float(100)
         print('\n****Test accuracy for k = 7 is %d%%' % (acc))
```

```
CV accuracy for k = 1 is 77%
CV accuracy for k = 3 is 85%

CV accuracy for k = 5 is 87%

CV accuracy for k = 7 is 88%

CV accuracy for k = 9 is 88%

****Test accuracy for k = 7 is 91%
```

```
In [24]: #with either k=8 or 9 we get best accuracy for test to be 91%,now with 10 fold cross

         n_list=list(range(0,30))
         neighb=list(filter(lambda x: x % 2 != 0, n_list))
         #create a list of cross-val scores
         scores_cv=[]
         for k in neighb:
             knn=KNeighborsClassifier(n_neighbors=k)
             scores=cross_val_score(knn,X_train,Y_train,cv=10,scoring='accuracy')
             scores_cv.append(scores.mean())
```

```
In [25]: #Find misclassification error(i.e)how much data is misclassified

         MSE=[1-x for x in scores_cv]

         #find optimal k

         optimal_k=neighb[MSE.index(min(MSE))]
         print('\n the optimal k is %d.' % optimal_k)
```

```
 the optimal k is 9.
```

```
In [26]: #KNN with optimal k and test accuracy for bag of words

         knn_opt=KNeighborsClassifier(n_neighbors=optimal_k)
         #fit the model
         knn_opt.fit(X_train,Y_train)
         #predict the model
         prediction=knn_opt.predict(X_test)

         #the accuracy score
         acc_score=accuracy_score(Y_test,pred)* 100
         print('\n the accuracy score for bag of words model with optimal k=%d is %f%%' %(opti
```

```
 the accuracy score for bag of words model with optimal k=9 is 91.500000%
```

4 TF-IDF KNN

```
In [27]: tf_idf_vect=TfidfVectorizer()
         final_tf_idf_vect=tf_idf_vect.fit_transform(test_data["CleanedText"].values)
         final_tf_idf_vect.get_shape()

Out[27]: (2000, 6910)
```

```
In [28]: #split the data into train and test fo tf-idf

         X_train,X_test,Y_train,Y_test=cross_validation.train_test_split(final_tf_idf_vect,labe
         #split train into cross val train and cross val test
         X_t,X_cv,Y_t,Y_cv=cross_validation.train_test_split(X_train,Y_train,test_size=0.3)
```

```
In [29]: X_train.shape

Out[29]: (1400, 6910)
```

```
In [30]: X_test.shape

Out[30]: (600, 6910)
```

```
In [31]: X_t.shape

Out[31]: (980, 6910)
```

```
In [32]: Y_t.shape

Out[32]: (980,)
```

```
In [33]: X_cv.shape

Out[33]: (420, 6910)
```

```
In [34]: #find the best k based on cv accuracy for bow

         for i in range(1,10,2):
             # instantiate learning model (k = 10)
             knn = KNeighborsClassifier(n_neighbors=i)

             # fitting the model on crossvalidation train
             knn.fit(X_t, Y_t)

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

             # evaluate CV accuracy
             acc = accuracy_score(Y_cv, pred, normalize=True) * float(100)
             print('\nCV accuracy for k = %d is %d%%' % (i, acc))
         #test accuracy
         knn = KNeighborsClassifier(9)
         knn.fit(X_t,Y_t)
         pred = knn.predict(X_test)
         acc = accuracy_score(Y_test, pred, normalize=True) * float(100)
         print('\n****Test accuracy for k = 9 is %d%%' % (acc))
```

```
CV accuracy for k = 1 is 79%

CV accuracy for k = 3 is 83%

CV accuracy for k = 5 is 84%

CV accuracy for k = 7 is 86%

CV accuracy for k = 9 is 86%

****Test accuracy for k = 9 is 91%
```

```
In [35]: #for finding optimal k with odd list for neighbors
         n_list=list(range(0,30))
         neighb=list(filter(lambda x: x % 2 != 0, n_list))
         #create a list of cross-val scores
         scores_cv=[]
         for k in neighb:
             knn=KNeighborsClassifier(n_neighbors=k)
             scores=cross_val_score(knn,X_train,Y_train,cv=10,scoring='accuracy')
             scores_cv.append(scores.mean())
```

```
In [36]: #Find misclassification error(i.e)how much data is misclassified

         MSE=[1-x for x in scores_cv]

         #find optimal k

         optimal_k=neighb[MSE.index(min(MSE))]
         print('\n the optimal k is %d.' % optimal_k)
```

```
 the optimal k is 15.
```

```
In [37]: #KNN with optimal k and test accuracy for tf_idf model

         knn_opt=KNeighborsClassifier(n_neighbors=optimal_k)
         #fit the model
         knn_opt.fit(X_train,Y_train)
         #predict the model
```

```
        prediction=knn_opt.predict(X_test)

        #the accuracy score
        acc_score=accuracy_score(Y_test,pred)* 100
        print('\n the accuracy score for bag of words model with optimal k=%d is %f%%' %(opti
```

```
 the accuracy score for bag of words model with optimal k=15 is 91.166667%
```

5 AvgWord2Vec KNN

```
In [38]: import gensim
         from gensim.models import word2vec,KeyedVectors
```

```
D:\Anaconda\lib\site-packages\gensim\utils.py:1209: UserWarning: detected Windows; aliasing chu
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

```
In [39]: model=word2vec.load('w2vmodel.model')
```

```
        ---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call last)

        <ipython-input-39-ff5ebc47b102> in <module>()
    ----> 1 model=word2vec.load('w2vmodel.model')


        AttributeError: module 'gensim.models.word2vec' has no attribute 'load'
```

```
In [40]: #word2vec own model
         i=0
         list_of_sentence=[]
         for sent in test_data['CleanedText'].values:
             list_of_sentence.append(sent.split())
             #sent=cleanhtml(sent)
             #for w in sent.split():
                 #for cleaned in cleanpunc(w).split():
                     #if(cleaned.isalpha()):
                         #filtered_sentence.append(cleaned.lower())
                     #else:
                         #continue
                 #list_of_sentence.append(filtered_sentence)
         print(test_data['CleanedText'].values[0])
         print('###########')
         print(list_of_sentence[0])
         w2v_model=gensim.models.Word2Vec(list_of_sentence,min_count=5,size=50,workers=4)

         words=list(w2v_model.wv.vocab)
         print(len(words))
```

```
sorri food snob allergi dont spit mouth finish chew mean tast good certain doesnt make graham
###########
['sorri', 'food', 'snob', 'allergi', 'dont', 'spit', 'mouth', 'finish', 'chew', 'mean', 'tast'
1990
```

```
In [41]: sent_vectors = []
         for sent in list_of_sentence: # 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 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]))

         vec_avg=np.array(sent_vectors)
```

```
2000
50
```

```
In [42]: #split the data into train and test fo tf-idf

         X_train,X_test,Y_train,Y_test=cross_validation.train_test_split(vec_avg,label_data,tes
         #split train into cross val train and cross val test
         X_t,X_cv,Y_t,Y_cv=cross_validation.train_test_split(X_train,Y_train,test_size=0.3)
```

```
In [44]: X_train.shape
```

```
Out[44]: (1400, 50)
```

```
In [45]: Y_cv.shape
```

```
Out[45]: (420,)
```

```
In [46]: Y_train.shape
```

```
Out[46]: (1400,)
```

```
In [47]: X_test.shape
```

```
Out[47]: (600, 50)
```

```
In [48]: X_cv.shape
```

```
Out[48]: (420, 50)
```

```
In [49]: X_t.shape
```

```
Out[49]: (980, 50)
```

```
In [53]: #find the best k based on cv accuracy for AvgWord2Vec

         for i in range(1,10,2):
             # instantiate learning model (k = 10)
             knn = KNeighborsClassifier(n_neighbors=i)

             # fitting the model on crossvalidation train
             knn.fit(X_t, Y_t)

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

             # evaluate CV accuracy
             acc = accuracy_score(Y_cv, pred, normalize=True) * float(100)
             print('\nCV accuracy for k = %d is %d%%' % (i, acc))
         #test accuracy
         knn = KNeighborsClassifier(9)
         knn.fit(X_t,Y_t)
         pred = knn.predict(X_test)
         acc = accuracy_score(Y_test, pred, normalize=True) * float(100)
         print('\n****Test accuracy for k = 9 is %d%%' % (acc))
```

```

CV accuracy for k = 1 is 80%

CV accuracy for k = 3 is 85%

CV accuracy for k = 5 is 87%

CV accuracy for k = 7 is 88%

CV accuracy for k = 9 is 89%

****Test accuracy for k = 9 is 91%
```

```
In [54]: #for finding optimal k with odd list for neighbors
         n_list=list(range(0,30))
```

```
        neighb=list(filter(lambda x: x % 2 != 0, n_list))
        #create a list of cross-val scores
        scores_cv=[]
        for k in neighb:
            knn=KNeighborsClassifier(n_neighbors=k)
            scores=cross_val_score(knn,X_train,Y_train,cv=10,scoring='accuracy')
            scores_cv.append(scores.mean())
```

```
In [55]: #Find misclassification error(i.e)how much data is misclassified

         MSE=[1-x for x in scores_cv]

         #find optimal k

         optimal_k=neighb[MSE.index(min(MSE))]
         print('\n the optimal k is %d.' % optimal_k)
```

```

 the optimal k is 9.
```

```
In [57]: #KNN with optimal k and test accuracy for AvgWord2Vec model

         knn_opt=KNeighborsClassifier(n_neighbors=optimal_k)
         #fit the model
         knn_opt.fit(X_train,Y_train)
         #predict the model
         prediction=knn_opt.predict(X_test)

         #the accuracy score
         acc_score=accuracy_score(Y_test,pred)* 100
         print('\n the accuracy score for AvgWord2Vec model with optimal k=%d is %f%%' %(optima
```

```

 the accuracy score for AvgWord2Vec model with optimal k=9 is 91.166667%
```

6 Weighted Tf-Idf Word2Vec KNN

```
In [58]: # TF-IDF weighted Word2Vec
         tf_idf_features = tf_idf_vect.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 li
         row=0
         for sent in list_of_sentence: # 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 words:
                    vec = w2v_model.wv[word]
                    # obtain the tf_idfidf of a word in a sentence/review
                    tf_idf = final_tf_idf_vect[row, tf_idf_features.index(word)]
                    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
```

```
In [59]: tf_vec_avg=np.array(tfidf_sent_vectors)
```

```
In [60]: #split the data into train and test fo tf-idf

         X_train,X_test,Y_train,Y_test=cross_validation.train_test_split(tf_vec_avg,label_data
         #split train into cross val train and cross val test
         X_t,X_cv,Y_t,Y_cv=cross_validation.train_test_split(X_train,Y_train,test_size=0.3)
```

```
In [62]: X_train.shape
```

```
Out[62]: (1400, 50)
```

```
In [63]: X_t.shape
```

```
Out[63]: (980, 50)
```

```
In [64]: X_cv.shape
```

```
Out[64]: (420, 50)
```

```
In [65]: X_test.shape
```

```
Out[65]: (600, 50)
```

```
In [66]: #find the best k based on cv accuracy for TfIDF Weighted Word2Vec

         for i in range(1,10,2):
             # instantiate learning model (k = 10)
             knn = KNeighborsClassifier(n_neighbors=i)

             # fitting the model on crossvalidation train
             knn.fit(X_t, Y_t)

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

             # evaluate CV accuracy
             acc = accuracy_score(Y_cv, pred, normalize=True) * float(100)
```

```
                print('\nCV accuracy for k = %d is %d%%' % (i, acc))
         #test accuracy
         knn = KNeighborsClassifier(9)
         knn.fit(X_t,Y_t)
         pred = knn.predict(X_test)
         acc = accuracy_score(Y_test, pred, normalize=True) * float(100)
         print('\n****Test accuracy for k = 9 is %d%%' % (acc))
```

```

CV accuracy for k = 1 is 81%

CV accuracy for k = 3 is 84%

CV accuracy for k = 5 is 85%

CV accuracy for k = 7 is 85%

CV accuracy for k = 9 is 85%

****Test accuracy for k = 9 is 91%
```

```
In [67]: #for finding optimal k with odd list for neighbors
         n_list=list(range(0,30))
         neighb=list(filter(lambda x: x % 2 != 0, n_list))
         #create a list of cross-val scores
         scores_cv=[]
         for k in neighb:
             knn=KNeighborsClassifier(n_neighbors=k)
             scores=cross_val_score(knn,X_train,Y_train,cv=10,scoring='accuracy')
             scores_cv.append(scores.mean())
```

```
In [68]: #Find misclassification error(i.e)how much data is misclassified

         MSE=[1-x for x in scores_cv]

         #find optimal k

         optimal_k=neighb[MSE.index(min(MSE))]
         print('\n the optimal k is %d.' % optimal_k)
```

```

 the optimal k is 15.
```

```
In [69]: #KNN with optimal k and test accuracy for Weighted TfIDF Word2Vec model

         knn_opt=KNeighborsClassifier(n_neighbors=optimal_k)
         #fit the model
```

```
knn_opt.fit(X_train,Y_train)
#predict the model
prediction=knn_opt.predict(X_test)

#the accuracy score
acc_score=accuracy_score(Y_test,pred)* 100
print('\n the accuracy score for TfIdf Word2Vec model with optimal k=%d is %f%%' %(op
```

```
the accuracy score for TfIdf Word2Vec model with optimal k=15 is 91.166667%
```

Conclusion/Observation
With optimal k being 9 or 15 we get a accuracy of 91% and 91.1666%