sentiment Analysis for Medicical reviews

July 27, 2019

1 importing required library

```
In [24]: import numpy as numpy
         import pandas as pd
         import matplotlib
         import matplotlib.pyplot as plt
         %matplotlib inline

         import seaborn as sns
```

```
In [25]: import sklearn
         from sklearn.utils import shuffle
         from sklearn.feature_extraction.text import TfidfVectorizer
```

2 NLP librararies

```
In [26]: import nltk
         from nltk.corpus import stopwords
         from nltk.tokenize import word_tokenize
```

```
In [27]: import re
         import random
```

```
In [28]: from collections import Counter
         import unicodedata as udata
         import string
```

```
In [29]: #checking the versions
         print(sklearn.__version__)
         print(matplotlib.__version__)
         print(numpy.__version__)
         print(pd.__version__)
         print(nltk.__version__)
```

```
0.18.1
2.0.0
1.11.3
```

```
0.19.2
3.2.2
```

```
In [30]: df=pd.read_csv("data/train_F3WbcTw1.csv")              #reading train data
         df_test=pd.read_csv("data/test_tOlRoBf.csv")           #reading test data
```

```
/opt/conda/lib/python3.6/site-packages/IPython/core/interactiveshell.py:2717: DtypeWarning: Columns (3) have mixed types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

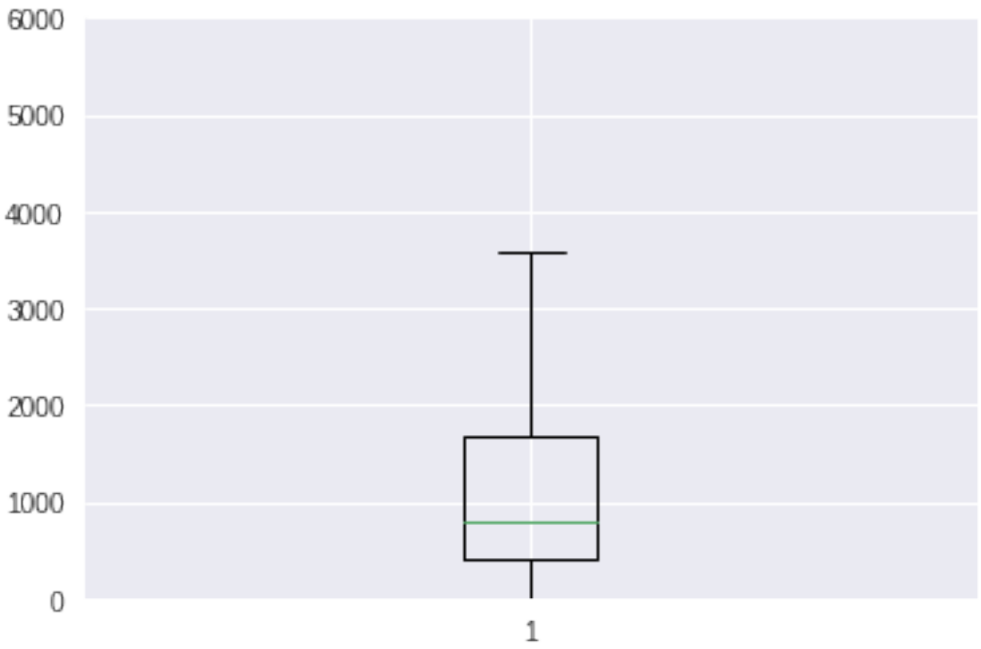
```
In [32]: df=df[["unique_hash","text","drug","sentiment"]]
         df_test=df_test[["unique_hash","text","drug"]]

In [33]: df_test.head(4)

Out[33]:                                unique_hash  \
         0  9e9a8166b84114aca147bf409f6f956635034c08
         1  e747e6822c867571afe7b907b51f0f2ca67b0e1a
         2  50b6d851bcff4f35afe354937949e9948975adf7
         3  7f82ec2176ae6ab0b5d20b5ffc767ac829f384ae

                                                 text        drug
         0  256 (previously stable on natalizumab), with 5...  fingolimod
         1  On fingolimod and have been since December 201...  fingolimod
         2  Apparently it's shingles! :-/ I do have a few ...      humira
         3  If the Docetaxel doing once a week x3 weeks th...    tagrisso

In [34]: df.columns

Out[34]: Index(['unique_hash', 'text', 'drug', 'sentiment'], dtype='object')

In [35]: df_test.columns

Out[35]: Index(['unique_hash', 'text', 'drug'], dtype='object')
```

3 cleaning data

```
In [36]: df.isnull().sum()

Out[36]: unique_hash     0
         text            0
         drug           36
         sentiment      36
         dtype: int64

In [37]: df_test.isnull().sum()
```

```
Out[37]: unique_hash    0
         text           0
         drug           0
         dtype: int64

In [38]: df.duplicated().sum()

Out[38]: 9

In [40]: df = df.drop_duplicates()

In [41]: df.duplicated().sum()

Out[41]: 0

In [42]: df_test.duplicated().sum()

Out[42]: 0

In [43]: df = df.drop(['unique_hash','drug'], axis = 1)

In [44]: df_test = df_test.drop(['unique_hash','drug'], axis = 1)

In [46]: df.dtypes

Out[46]: text         object
         sentiment    object
         dtype: object

In [47]: df_test.dtypes

Out[47]: text    object
         dtype: object

In [49]: df['sentiment'] = pd.to_numeric(df['sentiment'], errors='coerce')    #conve

In [50]: df.isnull().sum()

Out[50]: text          0
         sentiment    61
         dtype: int64

In [51]: df = df.dropna(how='any',axis=0)      #drops NaN rows

In [52]: df.isnull().sum()

Out[52]: text         0
         sentiment    0
         dtype: int64

In [53]: df.shape
```

```
Out[53]: (5245, 2)

In [54]: df.sentiment.value_counts()

Out[54]: 2.0    3796
         1.0     837
         0.0     612
         Name: sentiment, dtype: int64

In [55]: df['pre_clean_len'] = [len(t) for t in df.text]

/opt/conda/lib/python3.6/site-packages/ipykernel/__main__.py:1: SettingWithCopyWarn
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/i
  if __name__ == '__main__':


In [56]: ax=plt.gca()
         ax.set_ylim(0,6000)
         plt.boxplot(df.pre_clean_len)
         plt.show()                                          #graph for oultier detectio
```




```
In [57]: df[df.pre_clean_len > 140].head(10)
```

Out[57]:

	text	sentiment	pre_clean_len
0	Autoimmune diseases tend to come in clusters. ...	2.0	404
1	I can completely understand why you'd want to ...	2.0	1184
2	Interesting that it only targets S1P-1/5 recep...	2.0	780
4	Hi everybody, My latest MRI results for Brain ...	1.0	612
5	I can't give you advice about Lemtrada because...	2.0	285
6	Reply posted for JessZidek. Hi Jess Sorry to r...	0.0	755
7	Well as expected my Neurologist wants me to st...	2.0	723
8	Why do you think that FIngolimod was such a mi...	1.0	296
9	Thank you so much...I'm learning a lot here at G...	2.0	927
10	I have no vision in one eye, unrelated to my e...	1.0	1807

In [58]:
```
df_test.head(4)
```

Out[58]:

	text
0	256 (previously stable on natalizumab), with 5...
1	On fingolimod and have been since December 201...
2	Apparently it's shingles! :-/ I do have a few ...
3	If the Docetaxel doing once a week x3 weeks th...

In [59]:
```
df.reset_index(inplace = True)      #we are reindexing the train data as th
```

4 Cleaning operations

5 Importing beautiful soup

6 remove @ mentions from reviews

7 remove URLs from reviews

8 converting words like isn't to is not

9 get only text from the reviews

10 remove utf-8-sig code

11 converting all into lower case

12 will replace non-alphabetic characters by space

13 Word Punct Tokenize and only consider words whose length is greater than 1

14 join the words

```
In [60]: import re
         from bs4 import BeautifulSoup
         from nltk.tokenize import WordPunctTokenizer
         tok = WordPunctTokenizer()

         pat1 = r'@[A-Za-z0-9_]+'         # remove @ mentions from reviews
         pat2 = r'https?://[^ ]+'         # remove URLs from reviews
         combined_pat = r'|'.join((pat1, pat2)) #addition of pat1 and pat2
         www_pat = r'www.[^ ]+'          # remove URLs from reviews
         negations_dic = {"isn't":"is not", "aren't":"are not", "wasn't":"was not",
                          "haven't":"have not","hasn't":"has not","hadn't":"had not'
                          "wouldn't":"would not", "don't":"do not", "doesn't":"does
                          "can't":"can not","couldn't":"could not","shouldn't":"shou
                          "mustn't":"must not"}
         neg_pattern = re.compile(r'\b(' + '|'.join(negations_dic.keys()) + r')\b')

         def review_cleaner(text):  # define review_cleaner function to clean the r
             soup = BeautifulSoup(text, 'lxml')    # create beautiful soup object
             souped = soup.get_text()    # get only text from the reviews
             try:
                 bom_removed = souped.decode("utf-8-sig").replace(u"\ufffd", "?")
             except:
```

```
            bom_removed = souped
        stripped = re.sub(combined_pat, '', bom_removed) # calling combined_pa
        stripped = re.sub(www_pat, '', stripped) #remove URLs
        lower_case = stripped.lower()       # converting all into lower case
        neg_handled = neg_pattern.sub(lambda x: negations_dic[x.group()], lowe
        letters_only = re.sub("[^a-zA-Z]", " ", neg_handled)       # will repl
        words = [x for x  in tok.tokenize(letters_only) if len(x) > 1] # Word
        return (" ".join(words)).strip() # join the words
```

```
In [93]: limit=len(df.index)
         import time;
         ms = time.time()
         clean_reviews_texts = [] # initialize list
         for i in range(0,limit):
             if i % 10000==0:
                 print(i, time.time()-ms)
             clean_reviews_texts.append(review_cleaner(df['text'][i]))  # call revi
```

```
0 0.0019366741180419922
```

```
In [65]: len(df_test.index)
```

```
Out[65]: 2924
```

```
In [125]: Test_limit=len(df_test.index)
          import time;
          ms = time.time()
          Test_clean_reviews_texts = [] # initialize list
          for i in range(0,Test_limit):
              if i % 10000==0:
                  print(i, time.time()-ms)
              Test_clean_reviews_texts.append(review_cleaner(df_test['text'][i]))
```

```
0 0.0011870861053466797
```

```
In [70]: df.tail(5)
```

```
Out[70]:       index                                               text  sentiment
         5240   5310  Hi Bee, Thanks for the update and the good new...        0.0
         5241   5311  Have you had blood testing done to check your ...        2.0
         5242   5312           All the best to your husband and family.        2.0
         5243   5313  Hi bazza, luckily my eyes aren't so badly affe...        2.0
         5244   5314  Well, my MS appeared to be very mild for a num...        0.0

               pre_clean_len
         5240           1129
         5241            826
```

```
5242             41
5243            608
5244            527
```

```
In [73]: nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /home/jovyan/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
Out[73]: True
```

```
In [76]: word_tokens = [] # initialize list for tokens
         for word in clean_reviews_texts:  # for each word in clean_review_texts
             word_tokens.append(word_tokenize(word)) #tokenize word in clean_reviev
```

```
In [78]: Test_word_tokens = [] # initialize list for tokens
         for word in Test_clean_reviews_texts:  # for each word in clean_review_tex
             Test_word_tokens.append(word_tokenize(word)) #tokenize word in clean_r
```

```
In [79]: nltk.download('wordnet')   #lametization
```

```
[nltk_data] Downloading package wordnet to /home/jovyan/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
Out[79]: True
```

```
In [80]: df1 = [] # initialize list df1 to store words after lemmatization
         from nltk.stem import WordNetLemmatizer # import WordNetLemmatizer from nl
         lemmatizer = WordNetLemmatizer() # create an object of WordNetLemmatizer
         for l in word_tokens: # for loop for every tokens in word_token
             b = [lemmatizer.lemmatize(q) for q in l] #for every tokens in word_tok
             df1.append(b) #append b to list df1
```

```
In [81]: Test_df1 = [] # initialize list df1 to store words after lemmatization
         from nltk.stem import WordNetLemmatizer # import WordNetLemmatizer from nl
         lemmatizer = WordNetLemmatizer() # create an object of WordNetLemmatizer
         for l in Test_word_tokens: # for loop for every tokens in word_token
             b = [lemmatizer.lemmatize(q) for q in l] #for every tokens in word_tok
             Test_df1.append(b) #append b to list Test_df1
```

```
In [82]: clean_df1 =[] # initialize list clean_df1 to join word tokens after lemmat
         for c in df1:  # for loop for each list in df1
             a = " ".join(c) # join words in list with space in between and give it
             clean_df1.append(a) # append a to clean_df1
```

```
In [86]: Test_clean_df1 =[] # initialize list clean_df1 to join word tokens after l
         for c in Test_df1:  # for loop for each list in df1
             a = " ".join(c) # join words in list with space in between and give it
             Test_clean_df1.append(a) # append a to clean_df1
```

In [87]:
```
clean_df = pd.DataFrame(clean_df1,columns=['text'])
Test_clean_df = pd.DataFrame(Test_clean_df1,columns=['text'])
```

In [90]:
```
clean_df['clean_len'] = [len(t) for t in clean_df.text]
Test_clean_df['clean_len'] = [len(t) for t in Test_clean_df.text]
```

In [91]:
```
clean_df[clean_df.clean_len > 140].head(10)
```

Out[91]:

	text	clean_len
0	autoimmune disease tend to come in cluster a f...	361
1	can completely understand why you want to try ...	1107
2	interesting that it only target receptor rathe...	646
4	hi everybody my latest mri result for brain an...	557
5	can give you advice about lemtrada because cho...	178
6	reply posted for jesszidek hi jess sorry to re...	611
7	well a expected my neurologist want me to star...	666
8	why do you think that fingolimod wa such miser...	280
9	thank you so much learning lot here at grace s...	852
10	have no vision in one eye unrelated to my eye ...	1681

In [94]:
```
target2 = [] # initialize list
for i in range(0,limit): #
    target2.append(df['sentiment'][i])
clean_df['target']=target2
df.head()
```

Out[94]:

	index	text	sentiment	\
0	0	Autoimmune diseases tend to come in clusters. ...	2.0	
1	1	I can completely understand why you'd want to ...	2.0	
2	2	Interesting that it only targets S1P-1/5 recep...	2.0	
3	3	Very interesting, grand merci. Now I wonder wh...	2.0	
4	4	Hi everybody, My latest MRI results for Brain ...	1.0	

	pre_clean_len
0	404
1	1184
2	780
3	124
4	612

In [95]:
```
X = clean_df.text # get all the text in x variable
y = clean_df.target # get all the sentiments into y variable
print(X.shape) #print shape of x
print(y.shape) # print shape of y
from collections import Counter
print(set(y)) # equals to list(set(words))
print(Counter(y).values()) #
```

```
(5245,)
(5245,)
```

```
{0.0, 1.0, 2.0}
dict_values([3796, 837, 612])
```

```
In [130]: Test_X=Test_clean_df.text
          print(Test_X.shape)
          Test_X.head(5)

(2924,)
```

```
Out[130]: 0    previously stable on natalizumab with switchin...
          1    on fingolimod and have been since december the...
          2    apparently it shingle do have few red spot jus...
          3    if the docetaxel doing once week week then wee...
          4    cc stelara worked in matter of day for me if y...
          Name: text, dtype: object
```

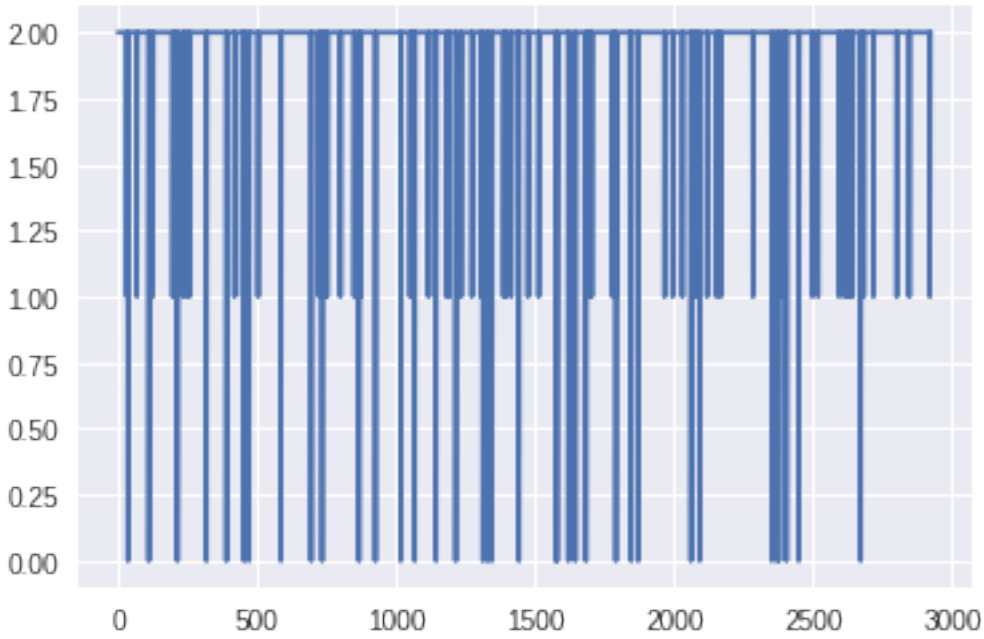
15 perform train and test split

```
In [96]: from sklearn.model_selection  import train_test_split #from sklearn.cross_
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20

In [97]: vect = TfidfVectorizer(analyzer = "word", ngram_range=(1,3))

In [133]: vect.fit(X_train)
          X_train_dtm = vect.transform(X_train)

In [99]: X_test_dtm = vect.transform(X_test)

In [120]: X_test_dtm

Out[120]: <1049x1121273 sparse matrix of type '<class 'numpy.float64'>'
                  with 416895 stored elements in Compressed Sparse Row format>

In [134]: TEST=vect.transform(Test_X)

In [135]: TEST

Out[135]: <2924x1121273 sparse matrix of type '<class 'numpy.float64'>'
                  with 1636563 stored elements in Compressed Sparse Row format>
```

16 SVC

```
In [109]: from sklearn.svm import LinearSVC # import SVC model from sklearn.svm
          svm_clf = LinearSVC(random_state=0) # get object of SVC model with rando

In [110]: svm_clf.fit(X_train_dtm, y_train)
```

```
Out[110]: LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
               intercept_scaling=1, loss='squared_hinge', max_iter=1000,
               multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
               verbose=0)

In [111]: from sklearn.model_selection import cross_val_score  # import cross_val_s
          accuracies = cross_val_score(estimator = svm_clf, X = X_train_dtm, y = y_
          accuracies.mean() # measure the mean accuray of 10 fold cross validation

Out[111]: 0.73069084194213807

In [115]: y_pred_svm = svm_clf.predict(X_test_dtm)  # predict the sentiments of tes

In [116]: y_pred_svm

Out[116]: array([ 2.,  2.,  2., ...,  2.,  0.,  2.])

In [117]: from sklearn import metrics  # import metrics from sklearn
          metrics.accuracy_score(y_test, y_pred_svm)  # measure the accuracy of our

Out[117]: 0.71877979027645378

In [118]: from sklearn.metrics import confusion_matrix # import confusion matrix fr
          confusion_matrix(y_test, y_pred_svm)# plot the confusion matrix between c

Out[118]: array([[  4,   3, 119],
                 [  0,  23, 161],
                 [  3,   9, 727]])

In [136]: y_predicted = svm_clf.predict(TEST)

In [137]: y_predicted

Out[137]: array([ 2.,  2.,  2., ...,  2.,  2.,  2.])

In [149]: plt.plot(y_predicted)
          plt.show()
```




```
In [155]: df_pred=pd.DataFrame({'Y_predict':y_predicted[:]})

In [158]: df_test_predicted=pd.read_csv("data/test_tOlRoBf.csv")

In [157]: df_test_pred=df_test[["unique_hash","text","drug"]].head(5)

Out[157]:                                                   text
          0  256 (previously stable on natalizumab), with 5...
          1  On fingolimod and have been since December 201...
          2  Apparently it's shingles! :-/ I do have a few ...
          3  If the Docetaxel doing once a week x3 weeks th...
          4  CC, Stelara worked in a matter of days for me...

In [165]: df_pred.Y_predict.value_counts()

Out[165]: 2.0    2823
          1.0      65
          0.0      36
          Name: Y_predict, dtype: int64

In [177]: output=pd.concat([df_test_pred,df_pred])
          df_test_pred[['predicted']]=df_pred[['Y_predict']]

In [178]: df_test_pred.head(4)

Out[178]:                                  unique_hash  \
          0  9e9a8166b84114aca147bf409f6f956635034c08
```

```
1  e747e6822c867571afe7b907b51f0f2ca67b0e1a
2  50b6d851bcff4f35afe354937949e9948975adf7
3  7f82ec2176ae6ab0b5d20b5ffc767ac829f384ae

                                                text        drug  out  \
0  256 (previously stable on natalizumab), with 5...  fingolimod  2.0
1  On fingolimod and have been since December 201...  fingolimod  2.0
2  Apparently it's shingles! :-/ I do have a few ...      humira  2.0
3  If the Docetaxel doing once a week x3 weeks th...    tagrisso  2.0

   predicted
0        2.0
1        2.0
2        2.0
3        2.0
```

In [180]:
```
df_test_pred.to_csv("data/Final_preducted_output.csv")
```

In [ ]: