```
import nltk
import pandas as pd
from nltk.corpus import stopwords
import numpy as np
from nltk.tokenize import word_tokenize
nltk.download('punkt')
from textblob import TextBlob
from string import punctuation
import gensim
from gensim.models.coherencemodel import CoherenceModel
from gensim.models.ldamodel import LdaModel
import pyLDAvis.gensim

df = pd.read_csv("K8 Reviews v0.2.csv")     # reading csv file
df.head()

text = df['review']

norm = [rev.lower() for rev in text]          #normalizing
norm[0:2]

norm_word = [word_tokenize(i) for i in norm]                #tokenizing
for i in norm_word:
   print(i)
norm_word[0:2]
```

```
pos_text = []                                   #applying parts of speech
#nltk.pos_tag(norm_word[5])
for sent in norm_word:
   pos_text.append(nltk.pos_tag(sent))

nn_words = []

for ter in pos_text:                            #filtering all Nouns from the list of words
   res=[sent for sent,pos in ter if pos.startswith ("N")]
   nn_words.append(res)

nn_words[1:10]

lemma=nltk.stem.WordNetLemmatizer()             #lemmatization
lemma_word = []

for sent in nn_words:
   lemma_word.append([lemma.lemmatize(words) for words in sent])
lemma_word[1:3]

stop_words = stopwords.words('english')
print(stop_words)
stop_punc=list(punctuation)
print(stop_punc)

stoppings = stop_words+stop_punc
print(stoppings)

filt_sent = []
```

```
for word in lemma_word:                    #removing stop words and punctuations from the list
    res=[trm for trm in word if trm not in stoppings]
    filt_sent.append(res)
filt_sent[1:3]


dictionary=gensim.corpora.Dictionary(filt_sent)
count=0
for a,b in dictionary.iteritems():
    print(a,b)
    count+=1
    if count>12:
        break
data=[dictionary.doc2bow(word) for word in filt_sent]
data[0:3]


[[(dictionary[id],freq) for id,freq in dt] for dt in data[0:3]]


goodLdamodel=LdaModel(corpus=data,id2word=dictionary,iterations=50,num_topics=2)
badLdamodel=LdaModel(corpus=data,id2word=dictionary,iterations=1,num_topics=2)


pyLDAvis.enable_notebook()
pyLDAvis.gensim.prepare(goodLdamodel,data,dictionary)
pyLDAvis.gensim.prepare(badLdamodel,data,dictionary)


goodcm = CoherenceModel(model=goodLdamodel, texts=filt_sent, dictionary=dictionary,
coherence='c_v')
badcm = CoherenceModel(model=badLdamodel, texts=filt_sent, dictionary=dictionary, coherence='c_v')
print(goodcm.get_coherence())              #0.4879
```

```
print(badcm.get_coherence())            #0.4745

print(goodLdamodel.show_topics(formatted=False))
print(badLdamodel.show_topics(formatted=False))

def sentence_format(ldamodel=goodLdamodel, corpus=data, texts=filt_sent):
    # Init output
    df1= pd.DataFrame()

    # Get main topic in each document
    for i, row in enumerate(ldamodel[corpus]):
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # ---> dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                df1 = df1.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]),
ignore_index=True)
            else:
                break
    df1.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']

    # Add original text to the end of the output
    contents = pd.Series(texts)
    sent_topics_df = pd.concat([df1, contents], axis=1)
    return(sent_topics_df)
```

```
df_topic_sents_keywords = sentence_format(ldamodel=goodLdamodel, corpus=data, texts=filt_sent)


# Format

df_dominant_topic = df_topic_sents_keywords.reset_index()

df_dominant_topic.columns = ['Document_No', 'Dominant_Topic', 'Topic_Perc_Contrib', 'Keywords',
'Text']


# Show

df_dominant_topic.head(10)
```