

# Import libraries and load data

In [1]: *#Importing Libraries*

```
import pickle
import pandas as pd
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import chi2
import numpy as np
```

In [2]: *#Accessing document uploaded*

```
path_df = "News_dataset.pickle"

with open(path_df, 'rb') as data:
    df = pickle.load(data)
```

In [3]: *#checking data*

```
df.head()
```

Out[3]:

	File_Name	Content	Category	Complete_Filename	id	News_length
0	001.txt	Ad sales boost Time Warner profit\r\n\r\nQuart...	business	001.txt-business	1	2569
1	002.txt	Dollar gains on Greenspan speech\r\n\r\nThe do...	business	002.txt-business	1	2257
2	003.txt	Yukos unit buyer faces loan claim\r\n\r\nThe o...	business	003.txt-business	1	1557
3	004.txt	High fuel prices hit BA's profits\r\n\r\nBriti...	business	004.txt-business	1	2421
4	005.txt	Pernod takeover talk lifts Domecq\r\n\r\nShare...	business	005.txt-business	1	1575

In [4]: *#Chcking article*

```
df.loc[1]['Content']
```

Out[4]: 'Dollar gains on Greenspan speech\r\n\r\nThe dollar has hit its highest level against the euro in almost three months after the Federal Reserve head said the US trade deficit is set to stabilise.\r\n\r\nAnd Alan Greenspan highlighted the US government's willingness to curb spending and rising household savings as factors which may help to reduce it. In late trading in New York, the dollar reached \$1.2871 against the euro, from \$1.2974 on Thursday. Market concerns about the deficit has hit the greenback in recent months. On Friday, Federal Reserve chairman Mr Greenspan's speech in London ahead of the meeting of G7 finance ministers sent the dollar higher after it had earlier tumbled on the back of worse-than-expected US jobs data. "I think the chairman's taking a much more sanguine view on the current account deficit than he's taken for some time," said Robert Sinche, head of currency strategy at Bank of America in New York. "He's taking a longer-term view, laying out a set of conditions under which the current account deficit can improve this year and next."\r\n\r\nWorries about the deficit concerns about China do, however, remain. China's currency remains pegged to the dollar and the US currency's sharp falls in recent months have therefore made Chinese export prices highly competitive. But calls for a shift in Beijing's policy have fallen on deaf ears, despite recent comments in a major Chinese newspaper that the "time is ripe" for a loosening of the peg. The G7 meeting is thought unlikely to produce any meaningful movement in Chinese policy. In the meantime, the US Federal Reserve's decision on 2 February to boost interest rates by a quarter of a point - the sixth such move in as many months - has opened up a differential with European rates. The half-point window, some believe, could be enough to keep US assets looking more attractive, and could help prop up the dollar. The recent falls have partly been the result of big budget deficits, as well as the US's yawning current account gap, both of which need to be funded by the buying of US bonds and assets by foreign firms and governments. The White House will announce its budget on Monday, and many commentators believe the deficit will remain at close to half a trillion dollars.'

# 1. Text cleaning and preparation

In [5]: *#Text cleaning*

```
df['Content_Parsed_1'] = df['Content'].str.replace("\r", " ")
df['Content_Parsed_1'] = df['Content_Parsed_1'].str.replace("\n", " ")
df['Content_Parsed_1'] = df['Content_Parsed_1'].str.replace(" ", " ")
df['Content_Parsed_1'] = df['Content_Parsed_1'].str.replace("'", '')
```

In [6]: *#Text preparation*

```
df['Content_Parsed_2'] = df['Content_Parsed_1'].str.lower() #all to lower case

punctuation_signs = list("?!.,;") #remove punctuation
df['Content_Parsed_3'] = df['Content_Parsed_2']

for punct_sign in punctuation_signs:
    df['Content_Parsed_3'] = df['Content_Parsed_3'].str.replace(punct_sign, '')

df['Content_Parsed_4'] = df['Content_Parsed_3'].str.replace("'s", "") #remove possessive
```

C:\Users\SARVESH\AppData\Local\Temp\ipykernel\_16900\3467828116.py:9: FutureWarning: The default value of regex will change from True to False in a future version. In addition, single character regular expressions will \*not\* be treated as literal strings when regex=True.

```
df['Content_Parsed_3'] = df['Content_Parsed_3'].str.replace(punct_sign, '')
```

a) Use any 1 method for Lemmatization

In [7]: *#Stemming and Lemmatization*

```
nltk.download('punkt')
```

```

nltk.download('wordnet')

nltk.download('averaged_perceptron_tagger')
from nltk.corpus import wordnet

```

```

[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\SARVESH\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\SARVESH\AppData\Roaming\nltk_data...
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data] C:\Users\SARVESH\AppData\Roaming\nltk_data...
[nltk_data] Unzipping taggers\averaged_perceptron_tagger.zip.

```

```

In [9]: import nltk
nltk.download('omw-1.4')

```

```

[nltk_data] Downloading package omw-1.4 to
[nltk_data] C:\Users\SARVESH\AppData\Roaming\nltk_data...

```

```

Out[9]: True

```

## 1st method for lemmatization

```

In [10]: #Stemming and Lemmatization

wordnet_lemmatizer = WordNetLemmatizer()
nrows = len(df)
lemmatized_text_list = []

for row in range(0, nrows):

    # Create an empty list containing lemmatized words
    lemmatized_list = []

    # Save the text and its words into an object
    text = df.loc[row]['Content_Parsed_4']
    text_words = text.split(" ")

    # Iterate through every word to lemmatize
    for word in text_words:
        lemmatized_list.append(wordnet_lemmatizer.lemmatize(word, pos="v"))

    # Join the list
    lemmatized_text = " ".join(lemmatized_list)

    # Append to the list containing the texts
    lemmatized_text_list.append(lemmatized_text)

df['Content_Parsed_5'] = lemmatized_text_list

```

```

In [11]: df['Content_Parsed_5']

```

```

Out[11]: 0      ad sales boost time warner profit quarterly pr...
          1      dollar gain on greenspan speech the dollar hav...
          2      yukos unit buyer face loan claim the owners of...
          3      high fuel price hit ba profit british airways ...
          4      pernod takeover talk lift domecq share in uk d...
          ...
          2220     bt program to beat dialler scam bt be introduc...
          2221     spam e-mail tempt net shoppers computer users ...
          2222     be careful how you code a new european directi...
          2223     us cyber security chief resign the man make su...
          2224     lose yourself in online game online role play ...
Name: Content_Parsed_5, Length: 2225, dtype: object

```

## 2nd method for lemmatization

```

In [12]: lemmatizer = WordNetLemmatizer()

# function to convert nltk tag to wordnet tag
def nltk_tag_to_wordnet_tag(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None

def lemmatize_sentence(sentence):
    #tokenize the sentence and find the POS tag for each token
    nltk_tagged = nltk.pos_tag(nltk.word_tokenize(sentence))
    #tuple of (token, wordnet_tag)
    wordnet_tagged = map(lambda x: (x[0], nltk_tag_to_wordnet_tag(x[1])), nltk_tagged)
    lemmatized_sentence = []
    for word, tag in wordnet_tagged:
        if tag is None:
            #if there is no available tag, append the token as is
            lemmatized_sentence.append(word)
        else:
            #else use the tag to lemmatize the token
            lemmatized_sentence.append(lemmatizer.lemmatize(word, tag))
    return " ".join(lemmatized_sentence)

nrows = len(df)
lemmatized_text_list = []

for row in range(0, nrows):
    lemmatized_text = lemmatize_sentence(df.loc[row]['Content_Parsed_4'])
    lemmatized_text_list.append(lemmatized_text)

df['Content_Parsed_5'] = lemmatized_text_list

In [13]: df['Content_Parsed_5']

```

```
Out[13]: 0      ad sale boost time warner profit quarterly pro...
        1      dollar gain on greenspan speech the dollar hav...
        2      yukos unit buyer face loan claim the owner of ...
        3      high fuel price hit ba profit british airway h...
        4      pernod takeover talk lift domecq share in uk d...
        ...
        2220     bt program to beat dialler scam bt be introduc...
        2221     spam e-mails tempt net shopper computer user a...
        2222     be careful how you code a new european directi...
        2223     us cyber security chief resign the man make su...
        2224     lose yourself in online gaming online role pla...
Name: Content_Parsed_5, Length: 2225, dtype: object
```

b) Use any 1 method for stop word

```
In [14]: #Downloading
```

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\SARVESH\AppData\Roaming\nltk_data...
[nltk_data] Unzipping corpora\stopwords.zip.
```

```
Out[14]: True
```

```
In [15]: #Removing stop words
```

```
stop_words = list(stopwords.words('english'))
```

## 1st Method

```
In [16]: df['Content_Parsed_6'] = df['Content_Parsed_5']
```

```
for stop_word in stop_words:
```

```
    regex_stopword = r"\b" + stop_word + r"\b"
    df['Content_Parsed_6'] = df['Content_Parsed_6'].str.replace(regex_stopword, '')
```

```
C:\Users\SARVESH\AppData\Local\Temp\ipykernel_16900\2995545048.py:6: FutureWarning: The default value of regex will change from True to False in a future version.
df['Content_Parsed_6'] = df['Content_Parsed_6'].str.replace(regex_stopword, '')
```

```
In [17]: df.loc[5]['Content_Parsed_6']
```

```
Out[17]: 'japan narrowly escape recession japan economy teeter brink technical recessio
n three month september figure show revised figure indicate growth 01 % - s
imilar-sized contraction previous quarter annual basis data suggest annual gr
owth 02 % suggest much hesitant recovery previously think common technical
definition recession two successive quarter negative growth government keen
play worrying implication data maintain view japan economy remain minor a
djustment phase upward climb monitor development carefully say economy minist
er heizo takenaka face strengthen yen make export less competitive indicatio
n weaken economic condition ahead observer less sanguine paint picture recov
ery much patchy previously think say paul sheard economist lehman brother tokyo
improvement job market apparently yet fee domestic demand private consumpti
on 02 % third quarter'
```

## 2nd Method

```
In [18]: stop_list_final=[]
```

```
nrows = len(df)
```

```
stopwords_english = stopwords.words('english')
```

```

for row in range(0, nrow):

    # Create an empty list containing no stop words
    stop_list = []

    # Save the text and its words into an object
    text = df.loc[row]['Content_Parsed_5']
    text_words = text.split(" ")

    # Iterate through every word to remove stopwords
    for word in text_words:
        if (word not in stopwords_english):
            stop_list.append(word)

    # Join the list
    stop_text = " ".join(stop_list)

    # Append to the list containing the texts
    stop_list_final.append(stop_text)

df['Content_Parsed_6'] = stop_list_final

```

In [19]: `df.loc[5]['Content_Parsed_6']`

Out[19]: 'japan narrowly escape recession japan economy teeter brink technical recession th  
ree month september figure show revised figure indicate growth 01 % - similar-size  
d contraction previous quarter annual basis data suggest annual growth 02 % sugges  
t much hesitant recovery previously think common technical definition recession tw  
o successive quarter negative growth government keen play worrying implication dat  
a maintain view japan economy remain minor adjustment phase upward climb monitor d  
evelopment carefully say economy minister heizo takenaka face strengthen yen make  
export less competitive indication weaken economic condition ahead observer less s  
anguine paint picture recovery much patchy previously think say paul sheard econom  
ist lehman brother tokyo improvement job market apparently yet fee domestic demand  
private consumption 02 % third quarter'

In [20]: `#Checking data`

```
df.head(1)
```

Out[20]:

	File_Name	Content	Category	Complete_Filename	id	News_length	Content_Parsed
0	001.txt	Ad sales boost Time Warner profit\r\n\r\nQuart...	business	001.txt-business	1	2569	Ad sales boc Time Warn profit Quarte p

In [21]: `#Removing the old content_parsed columns`

```

list_columns = ["File_Name", "Category", "Complete_Filename", "Content", "Content_Parsed_5"]
df = df[list_columns]

df = df.rename(columns={'Content_Parsed_6': 'Content_Parsed'})

```

In [22]: `df.head()`



Out[22]:

	File_Name	Category	Complete_Filename	Content	Content_Parsed
0	001.txt	business	001.txt-business	Ad sales boost Time Warner profit\r\n\r\nQuart...	ad sale boost time warner profit quarterly pro...
1	002.txt	business	002.txt-business	Dollar gains on Greenspan speech\r\n\r\nThe do...	dollar gain greenspan speech dollar hit high l...
2	003.txt	business	003.txt-business	Yukos unit buyer faces loan claim\r\n\r\nThe o...	yukos unit buyer face loan claim owner embattl...
3	004.txt	business	004.txt-business	High fuel prices hit BA's profits\r\n\r\nBriti...	high fuel price hit ba profit british airway b...
4	005.txt	business	005.txt-business	Pernod takeover talk lifts Domecq\r\n\r\nShare...	pernod takeover talk lift domecq share uk drin...

## 2. Label coding

In [23]: *#Generating new column for Category codes*

```
category_codes = {
    'business': 0,
    'entertainment': 1,
    'politics': 2,
    'sport': 3,
    'tech': 4
}

# Category mapping
df['Category_Code'] = df['Category']
df = df.replace({'Category_Code':category_codes})
```

In [24]: df.head()

Out[24]:

	File_Name	Category	Complete_Filename	Content	Content_Parsed	Category_Code
0	001.txt	business	001.txt-business	Ad sales boost Time Warner profit\r\n\r\nQuart...	ad sale boost time warner profit quarterly pro...	
1	002.txt	business	002.txt-business	Dollar gains on Greenspan speech\r\n\r\nThe do...	dollar gain greenspan speech dollar hit high l...	
2	003.txt	business	003.txt-business	Yukos unit buyer faces loan claim\r\n\r\nThe o...	yukos unit buyer face loan claim owner embattl...	
3	004.txt	business	004.txt-business	High fuel prices hit BA's profits\r\n\r\nBriti...	high fuel price hit ba profit british airway b...	
4	005.txt	business	005.txt-business	Pernod takeover talk lifts Domecq\r\n\r\nShare...	pernod takeover talk lift domecq share uk drin...	

### 3. Train - test split

```
In [25]: X_train, X_test, y_train, y_test = train_test_split(df['Content_Parsed'],
                                                             df['Category_Code'],
                                                             test_size=0.15,
                                                             random_state=8)
```

### 4. Text representation

TF-IDF Vectors

unigrams & bigrams corresponding to a particular category

```
In [26]: # Parameter election
ngram_range = (1,2)
min_df = 10
max_df = 1.
max_features = 300
```

```
In [27]: tfidf = TfidfVectorizer(encoding='utf-8',
                                   ngram_range=ngram_range,
                                   stop_words=None,
                                   lowercase=False,
                                   max_df=max_df,
                                   min_df=min_df,
                                   max_features=max_features,
                                   norm='l2',
                                   sublinear_tf=True)

features_train = tfidf.fit_transform(X_train).toarray()
labels_train = y_train
print(features_train.shape)
```



```
features_test = tfidf.transform(X_test).toarray()  
labels_test = y_test  
print(features_test.shape)
```

```
(1891, 300)
```

```
(334, 300)
```

In [28]: **from** sklearn.feature\_selection **import** chi2  
**import** numpy **as** np

```
for Product, category_id in sorted(category_codes.items()):  
    features_chi2 = chi2(features_train, labels_train == category_id)  
    indices = np.argsort(features_chi2[0])  
    feature_names = np.array(tfidf.get_feature_names())[indices]  
    unigrams = [v for v in feature_names if len(v.split(' ')) == 1]  
    bigrams = [v for v in feature_names if len(v.split(' ')) == 2]  
    print("# '{}' category:".format(Product))  
    print(" . Most correlated unigrams:\n. {}".format('\n. '.join(unigrams[-5:])))  
    print(" . Most correlated bigrams:\n. {}".format('\n. '.join(bigrams[-2:])))  
    print("")
```

```
# 'business' category:
. Most correlated unigrams:
. price
. market
. economy
. growth
. bank
. Most correlated bigrams:
. last year
. year old

# 'entertainment' category:
. Most correlated unigrams:
. best
. music
. star
. award
. film
. Most correlated bigrams:
. mr blair
. prime minister

# 'politics' category:
. Most correlated unigrams:
. blair
. party
. election
. tory
. labour
. Most correlated bigrams:
. prime minister
. mr blair

# 'sport' category:
. Most correlated unigrams:
. side
. player
. team
. game
. match
. Most correlated bigrams:
. say mr
. year old

# 'tech' category:
. Most correlated unigrams:
. mobile
. software
. technology
. computer
. user
. Most correlated bigrams:
. year old
. say mr
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: Future
Warning: Function get_feature_names is deprecated; get_feature_names is deprecated
in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: Future
Warning: Function get_feature_names is deprecated; get_feature_names is deprecated
in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: Future
Warning: Function get_feature_names is deprecated; get_feature_names is deprecated
in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: Future
Warning: Function get_feature_names is deprecated; get_feature_names is deprecated
in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: Future
Warning: Function get_feature_names is deprecated; get_feature_names is deprecated
in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

In [29]: bigrams

Out[29]: ['tell bbc', 'last year', 'mr blair', 'prime minister', 'year old', 'say mr']

Unigrams are more relevant to the category as compared with bigrams