Name:Mahfuzatul Bushra, ID:22-92354-1

Project for feature engineering to represent text data using TF-IDF model.

Import libraries and dependencies and settings

```
import warnings
warnings.filterwarnings('ignore') #supress warning in python

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pd.options.display.max_colwidth = 100
%matplotlib inline

import nltk
import re
```

Corpus Sample

```
corpus_sample=['I would say football is my favourite sport.',
               'The dog next door kept barking all night.',
               'A youth was stuffing a bag full of medical supplies.',
               'Billy turned on a radio to get the sports news.',
               'His job requires him to travel frequently.',
               'She excels at sport.',
               'We can expect rainy weather tomorrow.',
               'My wife works in a travel agency.',
               'Korean food is generally very spicy.',
               'We cannot exist without food or water.',
               'The weather is unpredictable around here.',
               'I spent the day at the medical facility.'
               ]
labels=['sport','animal','medicine','sport','travel','sport','weather'
,'travel','food','food','weather','medicine']

corpus_array=np.array(corpus_sample)
corpus_df = pd.DataFrame({'Document': corpus_sample,
                          'Category': labels})
corpus_df = corpus_df[['Document', 'Category']]
corpus_df
```

| | Document | Category |
|---|---|---|
| 0 | I would say football is my favourite sport. | sport |
| 1 | The dog next door kept barking all night. | animal |
| 2 | A youth was stuffing a bag full of medical supplies. | medicine |
| 3 | Billy turned on a radio to get the sports news. | sport |
| 4 | His job requires him to travel frequently. | travel |
| 5 | She excels at sport. | sport |
| 6 | We can expect rainy weather tomorrow. | weather |
| 7 | My wife works in a travel agency. | travel |

```
8                   Korean food is generally very spicy.      food
9                 We cannot exist without food or water.      food
10             The weather is unpredictable around here.   weather
11              I spent the day at the medical facility.  medicine
```

Corpus Pre_Processing

```
nltk.download('stopwords')
word_per_text=nltk.WordPunctTokenizer()
#Removing stop words with NLTK in Python(such as “the”, “a”, “an”, “in”)
stop_words = nltk.corpus.stopwords.words('english')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
def norm_doc(Document):
  Document = re.sub(r'[^a-zA-Z\s]', '', Document, re.I|re.A)#convert to lower case and remove special characters\whitespaces
  Document= Document.lower()
  Document= Document.strip()
  Tokens = word_per_text.tokenize(Document) # Tokenize the Document
  filtered_tokens = [token for token in Tokens if token not in stop_words]
  Document=' '.join(filtered_tokens)# re-create document from filtered tokens
  return Document

normalize_corpus = np.vectorize(norm_doc)
normalize_corpus = normalize_corpus(corpus_sample)
normalize_corpus
```

```
array(['would say football favourite sport',
       'dog next door kept barking night',
       'youth stuffing bag full medical supplies',
       'billy turned radio get sports news',
       'job requires travel frequently', 'excels sport',
       'expect rainy weather tomorrow', 'wife works travel agency',
       'korean food generally spicy', 'cannot exist without food water',
       'weather unpredictable around', 'spent day medical facility'],
      dtype='<U40')
```

TF-IDF Model

```
from sklearn.feature_extraction.text import CountVectorizer

#Get Features in Sparse Format
Count_Vectorizer=CountVectorizer(min_df=0., max_df=1.)
#Fit and transform
CV_BOW= Count_Vectorizer.fit_transform(normalize_corpus)
CV_BOW
```

```
<12x48 sparse matrix of type '<class 'numpy.int64'>'
     with 53 stored elements in Compressed Sparse Row format>
```

```
CV_BOW = CV_BOW.toarray() # Convert to an array
CV_BOW
```

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 1, 0],
       [0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1,
        0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0,
        0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 1],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
        0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
0,
        0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
0,
        0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1,
        0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0,
        1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0],
```

```
       [0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0,
        1, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
0,
        0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0,
        0, 0, 0, 0]])
```

```
from sklearn.feature_extraction.text import TfidfTransformer

#Compute the IDF values
TD_IDF= TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True)
#fit and transform
TDIDF_matrix = TD_IDF.fit_transform(CV_BOW)
#to array
TDIDF_matrix=TDIDF_matrix.toarray()
#Create DataFrame
features=Count_Vectorizer.get_feature_names()

pd.DataFrame(np.round(TDIDF_matrix, 2), columns=features)
```

```
    agency  around   bag  barking  billy  cannot   day   dog  door
excels  \
0     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
1     0.00     0.0  0.00     0.41   0.00    0.00  0.00  0.41  0.41
0.00
2     0.00     0.0  0.42     0.00   0.00    0.00  0.00  0.00  0.00
0.00
3     0.00     0.0  0.00     0.00   0.41    0.00  0.00  0.00  0.00
0.00
4     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
5     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.76
6     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
7     0.52     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
8     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
9     0.00     0.0  0.00     0.00   0.00    0.46  0.00  0.00  0.00
0.00
```

```
10    0.00     0.6  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
11    0.00     0.0  0.00     0.00   0.00    0.00  0.52  0.00  0.00
0.00

    ...  travel  turned  unpredictable  water  weather  wife  without
works  \
0   ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00
1   ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00
2   ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00
3   ...    0.00    0.41            0.0   0.00     0.00  0.00     0.00
0.00
4   ...    0.44    0.00            0.0   0.00     0.00  0.00     0.00
0.00
5   ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00
6   ...    0.00    0.00            0.0   0.00     0.44  0.00     0.00
0.00
7   ...    0.44    0.00            0.0   0.00     0.00  0.52     0.00
0.52
8   ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00
9   ...    0.00    0.00            0.0   0.46     0.00  0.00     0.46
0.00
10  ...    0.00    0.00            0.6   0.00     0.52  0.00     0.00
0.00
11  ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00

    would  youth
0    0.46   0.00
1    0.00   0.00
2    0.00   0.42
3    0.00   0.00
4    0.00   0.00
5    0.00   0.00
6    0.00   0.00
7    0.00   0.00
8    0.00   0.00
9    0.00   0.00
10   0.00   0.00
11   0.00   0.00

[12 rows x 48 columns]
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
#Compute the IDF values
TD_IDF_V= TfidfVectorizer(min_df=0.,max_df=1.,norm='l2', use_idf=True,
smooth_idf=True)

#Fit and transform
TDIDF_V_matrix = TD_IDF_V.fit_transform(normalize_corpus)
#to array
TDIDF_V_matrix=TDIDF_V_matrix.toarray()
#Create DataFrame
features_V=Count_Vectorizer.get_feature_names()
pd.DataFrame(np.round(TDIDF_V_matrix, 2), columns=features_V)
```

```
    agency  around   bag  barking  billy  cannot   day   dog  door
excels  \
0     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
1     0.00     0.0  0.00     0.41   0.00    0.00  0.00  0.41  0.41
0.00
2     0.00     0.0  0.42     0.00   0.00    0.00  0.00  0.00  0.00
0.00
3     0.00     0.0  0.00     0.00   0.41    0.00  0.00  0.00  0.00
0.00
4     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
5     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.76
6     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
7     0.52     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
8     0.00     0.0  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
9     0.00     0.0  0.00     0.00   0.00    0.46  0.00  0.00  0.00
0.00
10    0.00     0.6  0.00     0.00   0.00    0.00  0.00  0.00  0.00
0.00
11    0.00     0.0  0.00     0.00   0.00    0.00  0.52  0.00  0.00
0.00

    ...  travel  turned  unpredictable  water  weather  wife  without
works  \
0   ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00
1   ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00
2   ...    0.00    0.00            0.0   0.00     0.00  0.00     0.00
0.00
3   ...    0.00    0.41            0.0   0.00     0.00  0.00     0.00
0.00
4   ...    0.44    0.00            0.0   0.00     0.00  0.00     0.00
```

```
0.00
5   ...    0.00    0.00           0.0   0.00     0.00  0.00     0.00
0.00
6   ...    0.00    0.00           0.0   0.00     0.44  0.00     0.00
0.00
7   ...    0.44    0.00           0.0   0.00     0.00  0.52     0.00
0.52
8   ...    0.00    0.00           0.0   0.00     0.00  0.00     0.00
0.00
9   ...    0.00    0.00           0.0   0.46     0.00  0.00     0.46
0.00
10  ...    0.00    0.00           0.6   0.00     0.52  0.00     0.00
0.00
11  ...    0.00    0.00           0.0   0.00     0.00  0.00     0.00
0.00

    would  youth
0    0.46   0.00
1    0.00   0.00
2    0.00   0.42
3    0.00   0.00
4    0.00   0.00
5    0.00   0.00
6    0.00   0.00
7    0.00   0.00
8    0.00   0.00
9    0.00   0.00
10   0.00   0.00
11   0.00   0.00

[12 rows x 48 columns]
```