PEICE OF CODES FOR COUNT VECTORIZERS AND TFIDF

```
#COUNT VECTORIZER
#vectorizer converts the doc in the rows and columns. Rows --> number of docs or lines(=5)
docs=["the house had a tiny little mouse",
"the cat saw the mouse,cat is intelligent",
"the mouse ran away from the house, mouse is fast",
"the cat finally ate the mouse,cat is very happy",
"the end of the mouse story as mouse died"
]


from sklearn.feature_extraction.text import CountVectorizer
#Parameters
#      input--> corpus
#      stop words
#      n_gram range(unary, binary, etc)
#      max_df, min_df
#      max_features
cv=CountVectorizer(stop_words=['the','of','a','as','is'])
X=cv.fit_transform(docs)



cv.get_feature_names()# see the unique words and the stop words are not incluce here

     ['ate',
      'away',
      'cat',
      'died',
      'end',
      'fast',
      'finally',
      'from',
      'had',
      'happy',
      'house',
      'intelligent',
      'little',
      'mouse',
      'ran',
      'saw',
      'story',
      'tiny',
      'very']


print(X)# this is the sparse matrix, this will store the non zero frequencies with respect

       (0, 10)       1
       (0, 8)        1
       (0, 17)       1
       (0, 12)       1
       (0, 13)       1
       (1, 13)       1
       (1, 2)        2
```

```
  (1, 15)       1
  (1, 11)       1
  (2, 10)       1
  (2, 13)       2
  (2, 14)       1
  (2, 1)        1
  (2, 7)        1
  (2, 5)        1
  (3, 13)       1
  (3, 2)        2
  (3, 6)        1
  (3, 0)        1
  (3, 18)       1
  (3, 9)        1
  (4, 13)       2
  (4, 4)        1
  (4, 16)       1
  (4, 3)        1
```

```
X.shape# this will give us the array shape but it is stored in above format to preserve st
```

```
(5, 19)
```

```
X.toarray()# to convert to an array
```

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0],
       [0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0],
       [0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 2, 1, 0, 0, 0, 0],
       [1, 0, 2, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1],
       [0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 1, 0, 0]])
```

```
# FORM DATAFRAME
import pandas as pd
df=pd.DataFrame(X.toarray(),columns=cv.get_feature_names())
df.head()
```

	ate	away	cat	died	end	fast	finally	from	had	happy	house	intelligent	li
0	0	0	0	0	0	0	0	0	1	0	1	0	
1	0	0	2	0	0	0	0	0	0	0	0	1	
2	0	1	0	0	0	1	0	1	0	0	1	0	
3	1	0	2	0	0	0	1	0	0	1	0	0	
4	0	0	0	1	1	0	0	0	0	0	0	0	

```
cv.vocabulary_
```

```
{'ate': 0,
 'away': 1,
 'cat': 2,
 'died': 3,
 'end': 4,
 'fast': 5,
```

```
 'finally': 6,
 'from': 7,
 'had': 8,
 'happy': 9,
 'house': 10,
 'intelligent': 11,
 'little': 12,
 'mouse': 13,
 'ran': 14,
 'saw': 15,
 'story': 16,
 'tiny': 17,
 'very': 18}
```

WHAT IS IDF? IDF--> inverse document frequency -->log(number of documents or records/ number of records containing the word

```
# IDF is important because it slashes out most used common words.But the importance of log
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer(stop_words=['the','of','a','as','is'])
Y=tf.fit_transform(docs)
Y.shape
```

```
(5, 19)
```

```
tf.vocabulary_
```

```
{'ate': 0,
 'away': 1,
 'cat': 2,
 'died': 3,
 'end': 4,
 'fast': 5,
 'finally': 6,
 'from': 7,
 'had': 8,
 'happy': 9,
 'house': 10,
 'intelligent': 11,
 'little': 12,
 'mouse': 13,
 'ran': 14,
 'saw': 15,
 'story': 16,
 'tiny': 17,
 'very': 18}
```

```
tf.idf_# this gives the tfidf ie--> idf and these ar take only for non -zero elements
```

```
array([2.09861229, 2.09861229, 1.69314718, 2.09861229, 2.09861229,
       2.09861229, 2.09861229, 2.09861229, 2.09861229, 2.09861229,
       1.69314718, 2.09861229, 2.09861229, 1.        , 2.09861229,
       2.09861229, 2.09861229, 2.09861229, 2.09861229])
```

```
Y.toarray()
```

```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.50780572, 0.        ,
        0.40969445, 0.        , 0.50780572, 0.24197214, 0.        ,
        0.        , 0.        , 0.50780572, 0.        ],
       [0.        , 0.        , 0.73415285, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.45498177, 0.        , 0.21680125, 0.        ,
        0.45498177, 0.        , 0.        , 0.        ],
       [0.        , 0.42412706, 0.        , 0.        , 0.        ,
        0.42412706, 0.        , 0.42412706, 0.        , 0.        ,
        0.34218304, 0.        , 0.        , 0.40419763, 0.42412706,
        0.        , 0.        , 0.        , 0.        ],
       [0.38261915, 0.        , 0.61738945, 0.        , 0.        ,
        0.        , 0.38261915, 0.        , 0.        , 0.38261915,
        0.        , 0.        , 0.        , 0.18232008, 0.        ,
        0.        , 0.        , 0.        , 0.38261915],
       [0.        , 0.        , 0.        , 0.50583628, 0.50583628,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.4820674 , 0.        ,
        0.        , 0.50583628, 0.        , 0.        ]])
```

```
df1=pd.DataFrame(Y.toarray(),columns=cv.get_feature_names())
df1.head(4)
```

	ate	away	cat	died	end	fast	finally	from	had	h
0	0.000000	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.507806	0.00
1	0.000000	0.000000	0.734153	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.00
2	0.000000	0.424127	0.000000	0.0	0.0	0.424127	0.000000	0.424127	0.000000	0.00
3	0.382619	0.000000	0.617389	0.0	0.0	0.000000	0.382619	0.000000	0.000000	0.38

TFIDF TRANSFORMER ?

```
from sklearn.feature_extraction.text import TfidfTransformer

tt=TfidfTransformer()
z=tt.fit(cv.fit_transform(docs))
z.idf_
```

```
array([2.09861229, 2.09861229, 1.69314718, 2.09861229, 2.09861229,
       2.09861229, 2.09861229, 2.09861229, 2.09861229, 2.09861229,
       1.69314718, 2.09861229, 2.09861229, 1.        , 2.09861229,
       2.09861229, 2.09861229, 2.09861229, 2.09861229])
```

```
k=tt.transform(cv.fit_transform(docs))
k
```

```
<5x19 sparse matrix of type '<class 'numpy.float64'>'
        with 25 stored elements in Compressed Sparse Row format>
```

```
k.toarray()
```

```
array([[0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.50780572, 0.        ,
        0.40969445, 0.        , 0.50780572, 0.24197214, 0.        ,
        0.        , 0.        , 0.50780572, 0.        ],
       [0.        , 0.        , 0.73415285, 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.45498177, 0.        , 0.21680125, 0.        ,
        0.45498177, 0.        , 0.        , 0.        ],
       [0.        , 0.42412706, 0.        , 0.        , 0.        ,
        0.42412706, 0.        , 0.42412706, 0.        , 0.        ,
        0.34218304, 0.        , 0.        , 0.40419763, 0.42412706,
        0.        , 0.        , 0.        , 0.        ],
       [0.38261915, 0.        , 0.61738945, 0.        , 0.        ,
        0.        , 0.38261915, 0.        , 0.        , 0.38261915,
        0.        , 0.        , 0.        , 0.18232008, 0.        ,
        0.        , 0.        , 0.        , 0.38261915],
       [0.        , 0.        , 0.        , 0.50583628, 0.50583628,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.4820674 , 0.        ,
        0.        , 0.50583628, 0.        , 0.        ]])
```

MAJOR difference of tfidf vectorizer and tfidf transformer is

ormer feeds on count vectorizer. TFidf vectorizer need no countvectorizer function

ransform function in transformer doesnot work, sepertely we need to fit first and then transform wi

vectorizer is like a short cut