```
import string
import random
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('reuters')
from nltk.corpus import reuters
from nltk import FreqDist
import pandas as pd
import re
from nltk import ngrams , defaultdict, Counter
from nltk.util import ngrams
from nltk.tokenize import word_tokenize
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from collections import Counter
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
from nltk.lm.preprocessing import padded_everygram_pipeline
```

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

## NGRARM

```
data = pd.read_csv('/content/Dataset/clean_train.csv')
data.head(5)
sents = data['synopsis']
```

```
def preprocess_text(text):
    # Implement text cleaning and tokenization here (if needed)
    tokens = word_tokenize(text)
    return tokens
```

```
data_list = list(data['synopsis'].apply(word_tokenize))
```

```
n = 2
train_data, padded_sents = padded_everygram_pipeline(n, data_list)
```

```
from nltk.lm import MLE
model = MLE(n)
```

```
len(model.vocab)
```

```
0
```

```
model.fit(train_data, padded_sents)
print(model.vocab)
```

```
<Vocabulary with cutoff=1 unk_label='<UNK>' and 50706 items>
```

```
len(model.vocab)
```

```
50706
```

```
print(model.vocab.lookup(data_list[0]))
```

```
('A', 'young', 'scriptwriter', 'starts', 'bringing', 'valuable', 'objects', 'back', 'from', 'his', 'short', 'nightmares', 'of', 'bei
```

```
print(model.vocab.lookup('language is never random lah .'.split()))
```

```
('language', 'is', 'never', 'random', '<UNK>', '.')
```

```
model.counts['nightmares']
```

```
115
```

```
model.counts[['nightmares']]['are']
```

```
5
```

```
model.score('are', 'nightmares'.split())
```

```
0.043478260869565216
```

## TF-IDF

```
df_train = pd.read_csv('/content/Dataset/clean_train.csv', index_col=0)
# df_test  = pd.read_csv('./dataset/test.csv', index_col=0)
```

```
#using only 5 sentences for training
df_train = df_train.head(5)
```

```
data_list_train = list(df_train['synopsis'].apply(word_tokenize))
# data_list_test = list(df_test['synopsis'].apply(word_tokenize))
```

```
data_list_train
```

```
        'the',
        'Fire',
        'of',
        'their',
        'Fury',
        'Against',
        'the',
        'Hated',
        'Oppressors',
        '.']]
```

```
for i in data_list_train:
    print(i)
```

```
['A', 'young', 'scriptwriter', 'starts', 'bringing', 'valuable', 'objects', 'back', 'from', 'his', 'short', 'nightmares', 'of', 'bei
['A', 'director', 'and', 'her', 'friends', 'renting', 'a', 'haunted', 'house', 'to', 'capture', 'paranormal', 'events', 'in', 'order
['This', 'is', 'an', 'educational', 'video', 'for', 'families', 'and', 'family', 'therapists', 'that', 'describes', 'the', 'Behavior
['Scientists', 'working', 'in', 'the', 'Austrian', 'Alps', 'discover', 'that', 'a', 'glacier', 'is', 'leaking', 'a', 'liquid', 'that
['Buy', 'Day', '-', 'Four', 'Men', 'Widely', '-', 'Apart', 'in', 'Life', '-', 'By', 'Night', 'Shadows', 'United', 'in', 'One', 'Figh
```

```
from nltk.stem.porter import PorterStemmer


def tokenize(text):
    tokens = nltk.word_tokenize(text)
    stems = []
    for item in tokens:
        stems.append(PorterStemmer().stem(item))
    return stems


# create object
tfidf = TfidfVectorizer(tokenizer=tokenize, stop_words='english')

# get tf-df values
result = tfidf.fit_transform(df_train['synopsis'])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:521: UserWarning: The parameter 'token_pattern' will not
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/feature_extraction/text.py:406: UserWarning: Your stop_words may be inconsistent wit
  warnings.warn(
```

```
feature_names = tfidf.get_feature_names_out()

# Create a DataFrame to display the TF-IDF values for the first movie synopsis
tfidf_df = pd.DataFrame(result[0].T.todense(), index=feature_names, columns=['TF-IDF'])
tfidf_df = tfidf_df.sort_values(by=['TF-IDF'], ascending=False)
```

```
tfidf_df.head(30)
```

| | TF-IDF |
|---|---|
| young | 0.258992 |
| bring | 0.258992 |
| scriptwrit | 0.258992 |
| sell | 0.258992 |
| make | 0.258992 |
| demon | 0.258992 |
| short | 0.258992 |
| start | 0.258992 |
| chase | 0.258992 |
| rich | 0.258992 |
| hi | 0.258992 |
| valuabl | 0.258992 |
| object | 0.258992 |
| nightmar | 0.258992 |
| . | 0.246823 |
| order | 0.000000 |
| rent | 0.000000 |
| popular | 0.000000 |
| oppressor | 0.000000 |
| prove | 0.000000 |
| night | 0.000000 |
| psychiatr | 0.000000 |
| paranorm | 0.000000 |
| seriou | 0.000000 |
| scientist | 0.000000 |
| shadow | 0.000000 |
| therapi | 0.000000 |
| therapist | 0.000000 |
| thi | 0.000000 |
| unit | 0.000000 |

Next steps: Generate code with `tfidf_df` View recommended plots New interactive sheet

```
print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names_out(), tfidf.idf_):
    print(ele1, ':', ele2)
```

```
behavior : 2.09861228866811
bring : 2.09861228866811
buy : 2.09861228866811
captur : 2.09861228866811
chase : 2.09861228866811
day : 2.09861228866811
deal : 2.09861228866811
demon : 2.09861228866811
describ : 2.09861228866811
director : 2.09861228866811
discov : 2.09861228866811
educ : 2.09861228866811
event : 2.09861228866811
famili : 2.09861228866811
```

```
liquid : 2.09861228866811
local : 2.09861228866811
make : 2.09861228866811
men : 2.09861228866811
night : 2.09861228866811
nightmar : 2.09861228866811
object : 2.09861228866811
oppressor : 2.09861228866811
order : 2.09861228866811
paranorm : 2.09861228866811
popular : 2.09861228866811
prove : 2.09861228866811
psychiatr : 2.09861228866811
rent : 2.09861228866811
rich : 2.09861228866811
scientist : 2.09861228866811
scriptwrit : 2.09861228866811
sell : 2.09861228866811
seriou : 2.09861228866811
shadow : 2.09861228866811
short : 2.09861228866811
start : 2.09861228866811
therapi : 2.09861228866811
therapist : 2.09861228866811
thi : 2.09861228866811
unit : 2.09861228866811
valuabl : 2.09861228866811
vent : 2.09861228866811
video : 2.09861228866811
wide : 2.09861228866811
wildlif : 2.09861228866811
work : 2.09861228866811
young : 2.09861228866811
```

```
print('\nWord indexes:')
print(tfidf.vocabulary_)

# display tf-idf values
print('\ntf-idf value:')
# print(result)
```

```
Word indexes:
{'young': 66, 'scriptwrit': 50, 'start': 55, 'bring': 10, 'valuabl': 60, 'object': 40, 'hi': 29, 'short': 54, 'nightmar': 39, 'chase

tf-idf value:
```

```
# in matrix form
print('\ntf-idf values in matrix form:')
print(result.toarray())
```

```
  0.         0.25899237 0.         0.         0.25899237 0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.25899237
  0.         0.         0.         0.         0.         0.
  0.25899237 0.         0.         0.25899237 0.25899237 0.
  0.         0.         0.         0.         0.         0.
  0.25899237 0.         0.25899237 0.25899237 0.         0.
  0.25899237 0.25899237 0.         0.         0.         0.
  0.25899237 0.         0.         0.         0.         0.
  0.25899237]
 [0.         0.13627206 0.         0.         0.         0.
  0.         0.         0.28598221 0.         0.         0.
  0.28598221 0.         0.         0.         0.         0.
  0.28598221 0.         0.         0.         0.28598221 0.
  0.28598221 0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.28598221 0.28598221 0.28598221 0.28598221 0.         0.28598221
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.        ]
 [0.         0.10342437 0.         0.         0.         0.
  0.21704766 0.         0.         0.21704766 0.         0.
  0.         0.         0.         0.21704766 0.         0.21704766
  0.         0.         0.21704766 0.         0.65114297 0.
  0.         0.         0.         0.         0.         0.
  0.         0.21704766 0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.21704766 0.
```

```
[0.         0.13627206 0.28598221 0.28598221 0.         0.28598221
 0.         0.28598221 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.28598221 0.         0.         0.         0.
 0.         0.         0.28598221 0.         0.         0.
 0.         0.         0.28598221 0.         0.28598221 0.28598221
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.28598221 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.28598221 0.28598221
 0.        ]
[0.62247822 0.0988714  0.         0.         0.20749274 0.
```