naive-bayes

October 16, 2023

[54]:
```python
# Loading Libraries
import pandas as pd
import numpy as np
```

[2]:
```python
# Load Dataset
df = pd.read_csv('emails.csv')
df.head()
```

[2]:
```
                                                text  spam
0  Subject: naturally irresistible your corporate…     1
1  Subject: the stock trading gunslinger  fanny i…     1
2  Subject: unbelievable new homes made easy  im …     1
3  Subject: 4 color printing special  request add…     1
4  Subject: do not have money , get software cds …     1
```

[3]:
```python
df.shape
```

[3]:
```
(5728, 2)
```

[4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5728 entries, 0 to 5727
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    5728 non-null   object
 1   spam    5728 non-null   int64
dtypes: int64(1), object(1)
memory usage: 89.6+ KB
```

[5]:
```python
df.spam.value_counts()
```

[5]:
```
0    4360
1    1368
Name: spam, dtype: int64
```

[6]:
```
# Checking Duplicated rows
df.duplicated()
```

[6]:
```
0       False
1       False
2       False
3       False
4       False
        …
5723    False
5724    False
5725    False
5726    False
5727    False
Length: 5728, dtype: bool
```

[7]:
```
# Duplicated summery
df.duplicated().value_counts()
```

[7]:
```
False    5695
True       33
dtype: int64
```

[8]:
```
(~df.duplicated(subset=['text','spam'])).sum()
```

[8]:
```
5695
```

[9]:
```
# Total Duplicated rows
df.duplicated(subset=['text','spam']).sum()
```

[9]:
```
33
```

[10]:
```
# Duplicated Rows
df[df.duplicated()]
```

[10]:
```
                                                   text  spam
2155  Subject: research allocations to egm  hi becky…     0
2260  Subject: departure of grant masson  the resear…     0
2412  Subject: re : schedule and more . .  jinbaek ,…     0
2473  Subject: day off tuesday  stinson ,  i would l…     0
2763  Subject: re : your mail  zhendong ,  dr . kami…     0
3123  Subject: re : grades  pam ,  the students rese…     0
3152  Subject: tiger evals - attachment  tiger hosts…     0
3248  Subject: re : i am zhendong  zhendong ,  thank…     0
3249  Subject: hello from enron  dear dr . mcmullen …     0
3387  Subject: term paper  dr . kaminski ,  attached…     0
3573  Subject: telephone interview with the enron re…     0
```

```
3660  Subject: re : summer work . .  jinbaek ,  this…     0
3690  Subject: re : weather and energy price data  m…     0
3823  Subject: research get - together at sandeep ko…     0
4203  Subject: re : willow and pathstar evaluations …     0
4390  Subject: re : eprm 2001 houston  layla ,  my a…     0
4394  Subject: vp & director count for the research …     0
4771  Subject: re : weather and energy price data  m…     0
4785  Subject: re : interviews  vince ,  no problem …     0
5073  Subject: re : grades  pam ,  another group :  …     0
5180  Subject: phone time  dear dr . kaminski  thank…     0
5216  Subject: re : term project :  brian ,  no prob…     0
5316  Subject: re :  frank ,  yes .  vince  from : f…     0
5340  Subject: re : enron visit - - thanks  larry , …     0
5473  Subject: re : pserc industrial advisory board …     0
5521  Subject: re : get together this coming tuesday…     0
5542  Subject: re : contact info  glenn ,  please , …     0
5628  Subject: retail markets conference  i would li…     0
5632  Subject: term project :  this is the list of p…     0
5664  Subject: june 21 - 22 retail electricity confe…     0
5674  Subject: re : enron weather research  good aft…     0
5698  Subject: schedule and more . .  dr . kaminski …     0
5716  Subject: * special notification * aurora versi…     0
```

[11]:
```python
# Dropping Duplicated rows
df.drop_duplicates(keep=False,inplace=True)
```

[12]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5662 entries, 0 to 5727
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    5662 non-null   object
 1   spam    5662 non-null   int64
dtypes: int64(1), object(1)
memory usage: 132.7+ KB
```

[13]:
```python
df.spam.value_counts()
```

[13]:
```
0    4294
1    1368
Name: spam, dtype: int64
```

[14]:
```python
df.spam.unique()
```

[14]:
```
array([1, 0], dtype=int64)
```

0.1 Text Preprocessing

```
[15]: # Loading Text libraries
import nltk
import string
from nltk.corpus import stopwords, words
nltk.download('stopwords')
```

```
[nltk_data] Error loading stopwords: <urlopen error [Errno 11001]
[nltk_data]     getaddrinfo failed>
```

```
[15]: False
```

```
[16]: string.punctuation
```

```
[16]: '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

```
[17]: stopwords.fileids()
```

```
[17]: ['arabic',
 'azerbaijani',
 'basque',
 'bengali',
 'catalan',
 'chinese',
 'danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'greek',
 'hebrew',
 'hinglish',
 'hungarian',
 'indonesian',
 'italian',
 'kazakh',
 'nepali',
 'norwegian',
 'portuguese',
 'romanian',
 'russian',
 'slovene',
 'spanish',
 'swedish',
 'tajik',
 'turkish']
```

```
[18]: # Bangali stopwords
      stopwords.words('bengali')
```

```
[18]: ['   ',
       '  ',
       '   ',
       '    ',
       '   ',
       '    ',
       '    ',
       '   ',
       '  ',
       '   ',
       '   ',
       '    ',
       '  ',
       '    ',
       '  ',
       '   ',
       '  ',
       '  ',
       '     ',
       '    ',
       '   ',
       '   ',
       '   ',
       '    ',
       '    ',
       '   ',
       '  ',
       '  ',
       '  ',
       ' ',
       '    ',
       '  ',
       '   ',
       '   ',
       '  ',
       '  ',
       '   ',
       ' ',
       '   ',
       '  ',
       '  ',
       '  ',
       '   ',
       '    ',
'  ',
' ',
'  ',
'   ',
'    ',
'    ',
'  ',
'   ',
'  ',
' ',
'    ',
'  ',
'   ',
' ',
'  ',
'   ',
'  ',
'    ',
'   ',
' ',
'  ',
' ',
' ',
'  ',
'',
'',
'   ',
' ',
'  ',
' ',
'  ',
'    ',
'   ',
' ',
'  ',
'   ',
' ',
'  ',
'   ',
'   ',
'    ',
'   ',
'    ',
'   ',
'   ',
'    ',
'   ',
'      ',
'    ',
'     ',
'     ',
'     ',
'    ',
'      ',
'      ',
'      ',
'    ',
'     ',
'         ',
'      ',
'       ',
'     ',
'      ',
'    ',
'     ',
'    ',
'     ',
'     ',
'     ',
'   ',
'      ',
'    ',
'     ',
'     ',
'   ',
'   ',
'    ',
'     ',
'     ',
'    ',
'     ',
'    ',
'     ',
'     ',
'       ',
'     ',
'   ',
'     ',
'       ',
'     ',
'    ',
'     ',
'    ',
'     ',
'   ',
'  ',
'  ',
'  ',
'  ',
'  ',
'   ',
'    ',
'   ',
'    ',
'  ',
'    ',
' ',
'   ',
'   ',
'  ',
'     ',
'    ',
'   ',
'     ',
'    ',
'      ',
'       ',
' ',
'   ',
' ',
'  ',
'  ',
' ',
'  ',
' ',
'  ',
' ',
'   ',
'    ',
'  ',
'   ',
'      ',
'  ',
'  ',
'   ',
'   ',
'    ',
'  ',
'    ',
'   ',
'   ',
'      ',
'     ',
'       ',
'      ',
'     ',
'     ',
'      ',
'    ',
'    ',
'     ',
'   ',
'      ',
'      ',
'       ',
'     ',
'      ',
'     ',
'      ',
'     ',
'      ',
'      ',
'     ',
'     ',
'    ',
'     ',
'       ',
'        ',
'      ',
'  ',
'   ',
'    ',
'    ',
'      ',
'       ',
'      ',
'      ',
'     ',
'     ',
'    ',
'    ',
'      ',
'    ',
'    ',
'      ',
'    ',
'   ',
'   ',
'   ',
'    ',
'     ',
'    ',
'    ',
'     ',
'       ',
'     ',
'    ',
'    ',
'    ',
'   ',
'     ',
'      ',
'     ',
'  ',
'    ',
'  ',
'   ',
'    ',
'    ',
'     ',
'     ',
'   ',
'    ',
'    ',
'     ',
'  ',
'    ',
'    ',
'    ',
'    ',
'     ',
'     ',
'       ',
'    ',
'    ',
'   ',
'    ',
'   ',
'     ',
'    ',
'  ',
'   ',
'    ',
'   ',
'     ',
'    ',
'    ',
'       ',
'     ',
'    ',
'   ',
'   ',
'     ',
'    ',
'   ',
'     ',
'       ',
'      ',
'       ',
'    ',
'     ',
'       ',
'        ',
'     ',
'      ',
'    ',
'     ',
'        ',
'     ',
'      ',
'      ',
'    ',
'     ',
'       ',
'    ',
'      ',
'    ',
'   ',
'     ',
'      ',
'    ',
'     ',
'   ',
'    ',
'     ',
'      ',
'       ',
'      ',
'     ',
'      ',
'     ',
'      ',
'   ',
'    ',
'   ',
'   ',
'    ',
'    ',
'  ',
'      ',
'    ',
'   ',
'    ',
' ',
'   ',
'     ',
'    ',
'    ',
'   ',
'  ',
'  ',
'    ',
'     ',
'  ',
'    ',
'    ',
'      ',
'  ',
'    ',
'      ',
'     ',
'  ',
'     ',
'  ',
'   ',
'     ',
'      ',
'    ',
'     ',
'     ',
'    ',
'    ',
'    ',
'    ',
'    ',
'    ',
'    ',
'     ',
'     ',
 '   ',
 ' ',
 '  ',
 '   ',
 ' ',
 '  ',
 '   ',
 ' ',
 '   ',
 '   ',
 '  ',
 '   ',
 '     ',
 '    ',
 '     ',
 ' ',
 '  ',
 '   ',
 '   ',
 '  ',
 '    ',
 '     ',
 '   ',
 '  ',
 ' ']
```

[19]:
```
# English Stopwords
stopwords.words('english')
```

[19]:
```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
'him',
'his',
'himself',
'she',
"she's",
'her',
'hers',
'herself',
'it',
"it's",
'its',
'itself',
'they',
'them',
'their',
'theirs',
'themselves',
'what',
'which',
'who',
'whom',
'this',
'that',
"that'll",
'these',
'those',
'am',
'is',
'are',
'was',
'were',
'be',
'been',
'being',
'have',
'has',
'had',
'having',
'do',
'does',
'did',
'doing',
'a',
'an',
'the',
'and',
'but',
'if',
'or',
'because',
'as',
'until',
'while',
'of',
'at',
'by',
'for',
'with',
'about',
'against',
'between',
'into',
'through',
'during',
'before',
'after',
'above',
'below',
'to',
'from',
'up',
'down',
'in',
'out',
'on',
'off',
'over',
'under',
'again',
'further',
'then',
'once',
'here',
'there',
'when',
'where',
'why',
'how',
'all',
'any',
'both',
'each',
'few',
'more',
'most',
'other',
'some',
'such',
'no',
'nor',
'not',
'only',
'own',
'same',
'so',
'than',
'too',
'very',
's',
't',
'can',
'will',
'just',
'don',
"don't",
'should',
"should've",
'now',
'd',
'll',
'm',
'o',
're',
've',
'y',
'ain',
'aren',
"aren't",
'couldn',
"couldn't",
'didn',
"didn't",
'doesn',
"doesn't",
'hadn',
"hadn't",
'hasn',
"hasn't",
'haven',
"haven't",
'isn',
"isn't",
'ma',
'mightn',
"mightn't",
'mustn',
"mustn't",
'needn',
"needn't",
'shan',
"shan't",
'shouldn',
"shouldn't",
'wasn',
"wasn't",
'weren',
"weren't",
'won',
"won't",
'wouldn',
"wouldn't"]
```

[20]:
```
# Total Bangali Stopwords
len(stopwords.words('bengali'))
```

[20]:
```
398
```

[21]:
```
def clean_text_data(text):
#     removing punctuation
    rem_punc = [char for char in text if char not in string.punctuation]
    rem_punc = ''.join(rem_punc)

#     remove all valueless stop words
    words = [word for word in rem_punc.split() if word.lower not in stopwords.
 ↪words('english')]
    return words
```

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

[22]:
```
0    Subject: naturally irresistible your corporate…
1    Subject: the stock trading gunslinger  fanny i…
2    Subject: unbelievable new homes made easy  im …
3    Subject: 4 color printing special  request add…
4    Subject: do not have money , get software cds …
Name: text, dtype: object
```

[23]:
```
# Applying function to all rows
df.text.head().apply(clean_text_data)
```

```
[23]: 0    [Subject, naturally, irresistible, your, corpo…
      1    [Subject, the, stock, trading, gunslinger, fan…
      2    [Subject, unbelievable, new, homes, made, easy…
      3    [Subject, 4, color, printing, special, request…
      4    [Subject, do, not, have, money, get, software,…
      Name: text, dtype: object
```

0.2 Feature Extraction

```
[24]: from sklearn.feature_extraction.text import TfidfVectorizer
```

```
[25]: tfidf = TfidfVectorizer(analyzer=clean_text_data)
```

```
[90]: all_texts = tfidf.fit_transform(df.text)
      all_texts
```

```
[90]: <5662x37356 sparse matrix of type '<class 'numpy.float64'>'
              with 732429 stored elements in Compressed Sparse Row format>
```

```
[91]: from sklearn.model_selection import train_test_split
      xtrain,xtest,ytrain,ytest = train_test_split(all_texts,df.spam,train_size=.
       ↪7,random_state=43)
```

```
[38]: xtrain.shape
```

```
[38]: (3963, 37356)
```

```
[39]: ytrain.shape
```

```
[39]: (3963,)
```

0.2.1 Multinomial Naive Bayes

```
[40]: from sklearn.naive_bayes import MultinomialNB
      mnb = MultinomialNB()
```

```
[41]: mnb.fit(xtrain,ytrain)
```

```
[41]: MultinomialNB()
```

```
[42]: mnb.score(xtest,ytest)
```

```
[42]: 0.8658034137728076
```

0.2.2 Bernoulli Naive Bayes

```
[43]: from sklearn.naive_bayes import BernoulliNB
      bnb = BernoulliNB()
```

```
[44]: bnb.fit(xtrain,ytrain)
```

```
[44]: BernoulliNB()
```

```
[45]: bnb.score(xtest,ytest)
```

```
[45]: 0.9846968805179518
```

0.2.3 Gaussian Naive Bayes

```
[81]: from sklearn.naive_bayes import GaussianNB
      gnb = GaussianNB()
```

```
[92]: all_texts2 = all_texts.toarray()
      all_texts2
```

```
[92]: array([[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.]])
```

```
[93]: xtrain,xtest,ytrain,ytest = train_test_split(all_texts2,df.spam,train_size=.
       ↪7,random_state=43)
```

```
[94]: gnb.fit(xtrain,ytrain)
```

```
[94]: GaussianNB()
```

```
[95]: gnb.score(xtest,ytest)
```

```
[95]: 0.951736315479694
```