spam

December 9, 2025

[1]:
```
HAM = 'ham'
SPAM = 'spam'

SOURCES = [
    ('enron1/enron1/spam',SPAM),
    ('enron1/enron1/ham',HAM),
    ('enron2/enron2/spam',SPAM),
    ('enron2/enron2/ham',HAM),
    ('enron3/enron3/spam',SPAM),
    ('enron3/enron3/ham',HAM),
]
```

[2]:
```
from pandas import DataFrame
data = DataFrame({'text': [], 'label': []})

#            text                                        label
#            este es un email que debe                   SPAM
#            estar etiquetado viagra como
#            esta en la columna de la derecha
```

[3]:
```
import os

def read_files(path):
    file_names = os.listdir(path)
    for file_name in file_names:
        file_path = os.path.join(path, file_name)
        lines = []
        f = open(file_path, encoding="latin-1")
        for line in f:
            lines.append(line)
        f.close()
    text = '\n'.join(lines)
    yield file_path, text
```

[4]:
```
def build_data_frame(path, label):
    rows = []
    index = []
    for file_path, text in read_files(path):
        rows.append({'text': text, 'label': label})
        index.append(file_path)
    data_frame = DataFrame(rows, index=index)

    return data_frame
```

```
[5]: for path, label in SOURCES:
         data = data._append(build_data_frame(path, label))
```

```
[6]: import numpy
     data = data.reindex(numpy.random.permutation(data.index))
```

```
[7]: '''
        Linux        today        Viagra       Free
     ham          619           67            0          50
     spam        3          432           291         534

     Este proceso se llama tokenización porque transformamos una
     colección de documentos de texto a una matriz de recuento de
     tokens.
     '''
```

```
[7]: '\n    Linux\ttoday\tViagra\tFree\nham\t  619\t   67\t     0\t  50\nspam\t3\t
     432\t   291\t 534\n\nEste proceso se llama tokenización porque transformamos una
     \ncolección de documentos de texto a una matriz de recuento de \ntokens. \n'
```

```
[8]: print(data)
     from sklearn.feature_extraction.text import CountVectorizer
     count_vectorizer = CountVectorizer()
     email_bodies = data['text'].values
     features = count_vectorizer.fit_transform(email_bodies)
     labels = data['label'].values
```

```
                 text  \
enron2/enron2/spam\5857.2005-07-22.SA_and_HP.sp…  Subject: microsoft autoroute
2005 dvd uk - $ 1…
enron1/enron1/spam\5171.2005-09-06.GP.spam.txt      Subject: save up to 89 % on
ink + no shipping …
enron3/enron3/spam\5511.2005-07-31.BG.spam.txt      Subject: = ? iso - 8859 - 1
? q ? v = eollum _…
enron1/enron1/ham\5172.2002-01-11.farmer.ham.txt    Subject: re : tenaska
iv\n\ni tried calling yo…
enron2/enron2/ham\5856.2001-05-22.kaminski.ham.txt  Subject: fw : memo : re :
your work phone numb…
enron3/enron3/ham\5512.2002-02-06.kitchen.ham.txt   Subject: re :
releases\n\nlouise ,\n\nthanks s…

                                                   label
```

```
enron2/enron2/spam\5857.2005-07-22.SA_and_HP.sp…  spam
enron1/enron1/spam\5171.2005-09-06.GP.spam.txt      spam
enron3/enron3/spam\5511.2005-07-31.BG.spam.txt      spam
enron1/enron1/ham\5172.2002-01-11.farmer.ham.txt     ham
enron2/enron2/ham\5856.2001-05-22.kaminski.ham.txt   ham
enron3/enron3/ham\5512.2002-02-06.kitchen.ham.txt    ham
```

[9]:

```
print(features)
#import numpy as np
#VecIdTexto = np.array(features)
#print(VecIdTexto)
```

```
<Compressed Sparse Row sparse matrix of dtype 'int64'
        with 973 stored elements and shape (6, 716)>
  Coords        Values
  (0, 605)      1
  (0, 405)      1
  (0, 85)       1
  (0, 12)       1
  (0, 206)      1
  (0, 663)      1
  (0, 9)        1
  (0, 31)       1
  (0, 193)      1
  (0, 585)      1
  (0, 602)      1
  (0, 301)      1
  (0, 712)      1
  (0, 345)      1
  (0, 137)      1
  (0, 340)      1
  (0, 427)      1
  (0, 348)      1
  (0, 261)      1
  (0, 692)      2
  (0, 714)      2
  (0, 527)      1
  (0, 700)      1
  (0, 125)      1
  (0, 334)      4
  :     :
  (5, 587)      1
  (5, 671)      2
  (5, 532)      4
  (5, 192)      1
  (5, 603)      1
  (5, 516)      1
  (5, 84)       1
  (5, 530)	1
  (5, 403)	1
  (5, 42)	1
  (5, 157)	1
  (5, 82)	1
  (5, 36)	1
  (5, 170)	1
  (5, 229)	1
  (5, 102)	1
  (5, 218)	1
  (5, 101)	1
  (5, 46)	1
  (5, 477)	1
  (5, 539)	1
  (5, 69)	1
  (5, 91)	1
  (5, 227)	1
  (5, 622)	1
```

[16]:

```python
from sklearn.model_selection import train_test_split

#dataset divide train y test
#                caracte_train   etiquetas train   ---> 80%
#                caracte_test    etiquetas_test    ---> 20%

features_train, features_test, labels_train, labels_test = \
    train_test_split(features, labels, train_size=0.8, test_size=0.2, random_state=0)
```

[17]:

```python
from sklearn.naive_bayes import MultinomialNB
clf = MultinomialNB()
```

[18]:

```python
clf.fit(features_train, labels_train)
```

[18]:

```
MultinomialNB()
```

[19]:

```python
from sklearn.metrics import accuracy_score

labels_predict = clf.predict(features_test)

accuracy = accuracy_score(labels_predict, labels_test)

print("Accuracy: %.2f" %(accuracy))
```

```
Accuracy: 1.00
```

[20]:

```python
from sklearn.metrics import confusion_matrix
confusion_matrix(labels_predict, labels_test)
```

```
[20]: array([[1, 0],
              [0, 1]])
```

```
[21]: test_emails = ["Free Viagra",
                      "I' am going an attend the meeting tomorrow"]

      test_features = count_vectorizer.transform(test_emails)
      labels_predict = clf.predict(test_features)
      print("Predictions: %s" % labels_predict)
```

```
Predictions: ['spam' 'ham']
```

```
[ ]:
```