D22：手刻 Naive Bayes (單純貝氏)

課程閱讀 本日作業 問題討論 學習心得(完成)

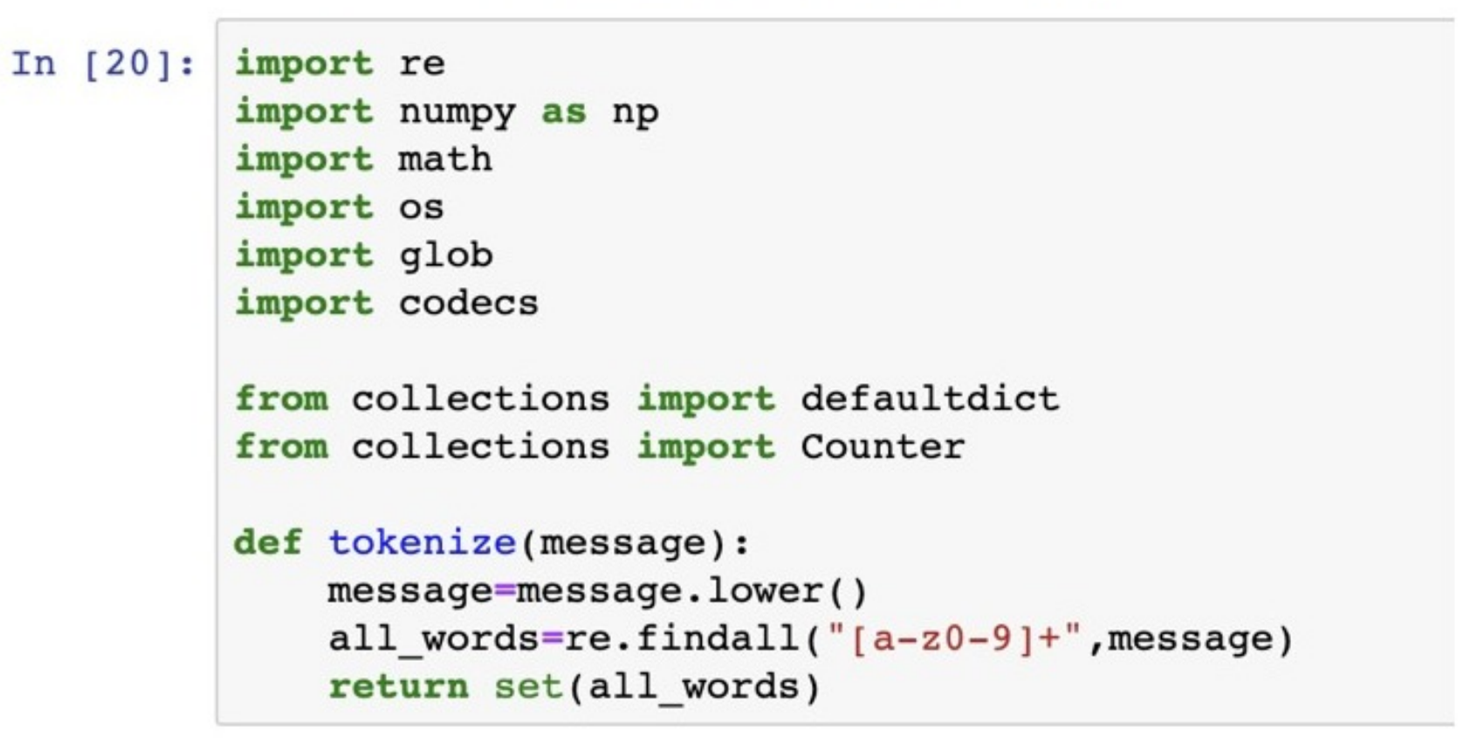
- 重要知識點 ›
- 導入 module(模組) 與 package(套件) ›
- 讀入訓練資料 ›

重要知識點

- 瞭解如何利用 Python 實踐單純貝氏模型

導入 module(模組) 與 package(套件)

定義 tokenize function ： tokenize 可以參考先前課程，有多種方式皆可實踐 tokenize。



```
In [20]: import re
         import numpy as np
         import math
         import os
         import glob
         import codecs

         from collections import defaultdict
         from collections import Counter

         def tokenize(message):
             message=message.lower()
             all_words=re.findall("[a-z0-9]+",message)
             return set(all_words)
```

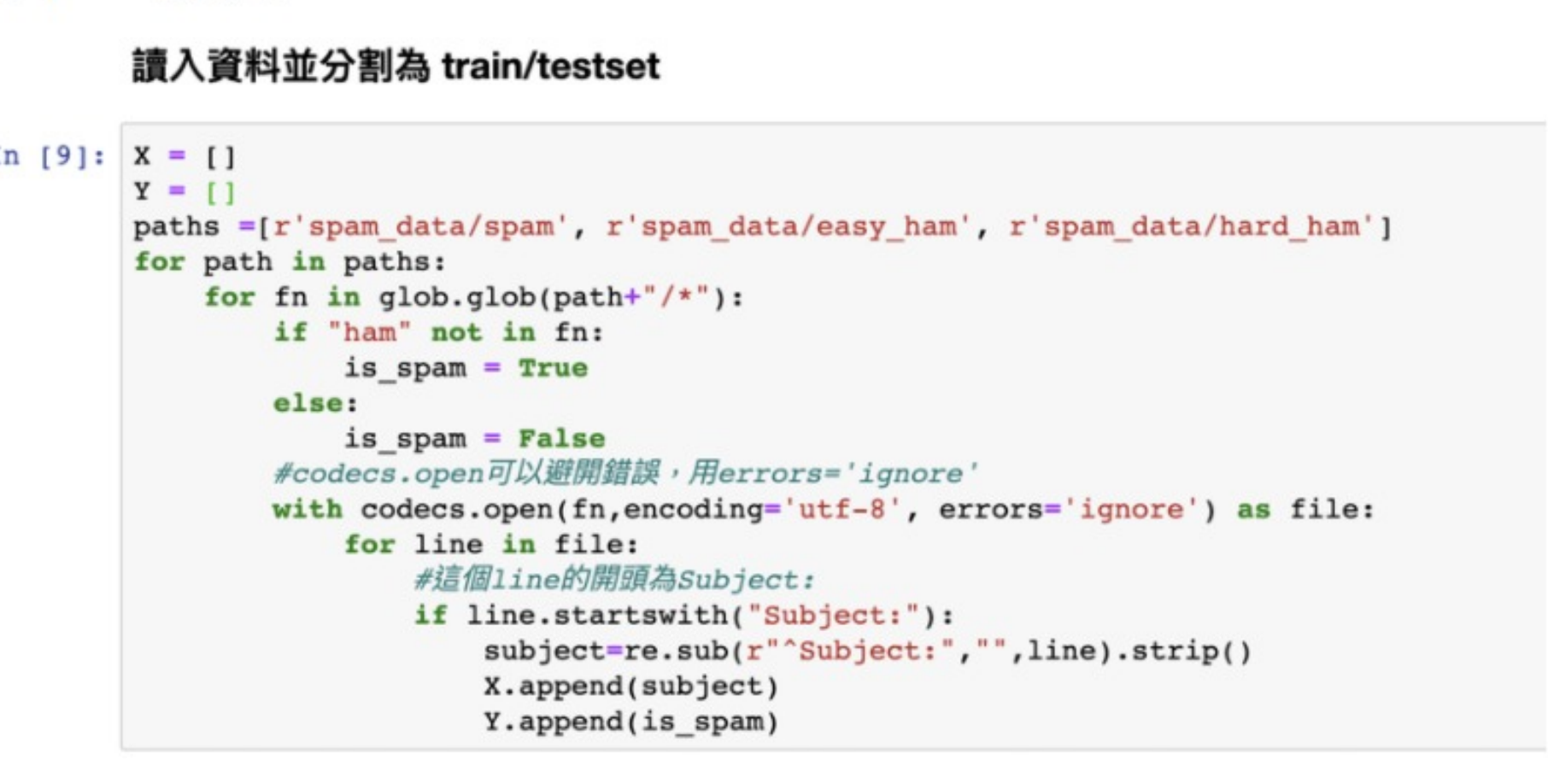
```
import re
import numpy as np
import math
import os
import glob
import codecs

from collections import defaultdict
from collections import Counter

def tokenize(message):
  message=message.lower()
```

讀入訓練資料

is_sapm ：垃圾郵件



讀入資料並分割為 train/testset

```
In [9]: X = []
        Y = []
        paths =[r'spam_data/spam', r'spam_data/easy_ham', r'spam_data/hard_ham']
        for path in paths:
            for fn in glob.glob(path+"/*"):
                if "ham" not in fn:
                    is_spam = True
                else:
                    is_spam = False
                #codecs.open可以避開錯誤，用errors='ignore'
                with codecs.open(fn,encoding='utf-8', errors='ignore') as file:
                    for line in file:
                        #這個line的開頭為Subject:
                        if line.startswith("Subject:"):
                            subject=re.sub(r"^Subject:","",line).strip()
                            X.append(subject)
                            Y.append(is_spam)
```

```
X = []
Y = []
paths =[r'spam_data/spam', r'spam_data/easy_ham', r'spam_data/hard_ham']
for path in paths:
  for fn in glob.glob(path+"/*"):
    if "ham" not in fn:
      is_spam = True
    else:
      is_spam = False
    #codecs.open可以避開錯誤，用errors='ignore'
    with codecs.open(fn,encoding='utf-8', errors='ignore') as file:
      for line in file:
```

分割 trainset / testset

```
In [10]: from sklearn.model_selection import train_test_split
         # random_state 是為了讓各為學員得到相同的結果，平時可以移除
         X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)

In [11]: train_data = []
         test_data = []

         for x_, y_ in zip(X_train, y_train):
             train_data.append([x_, y_])

         for x_, y_ in zip(X_test, y_test):
             test_data.append([x_, y_])
```

```
from sklearn.model_selection import train_test_split
# random_state 是為了讓各位學員得到相同的結果，平時可以移除
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0)
```

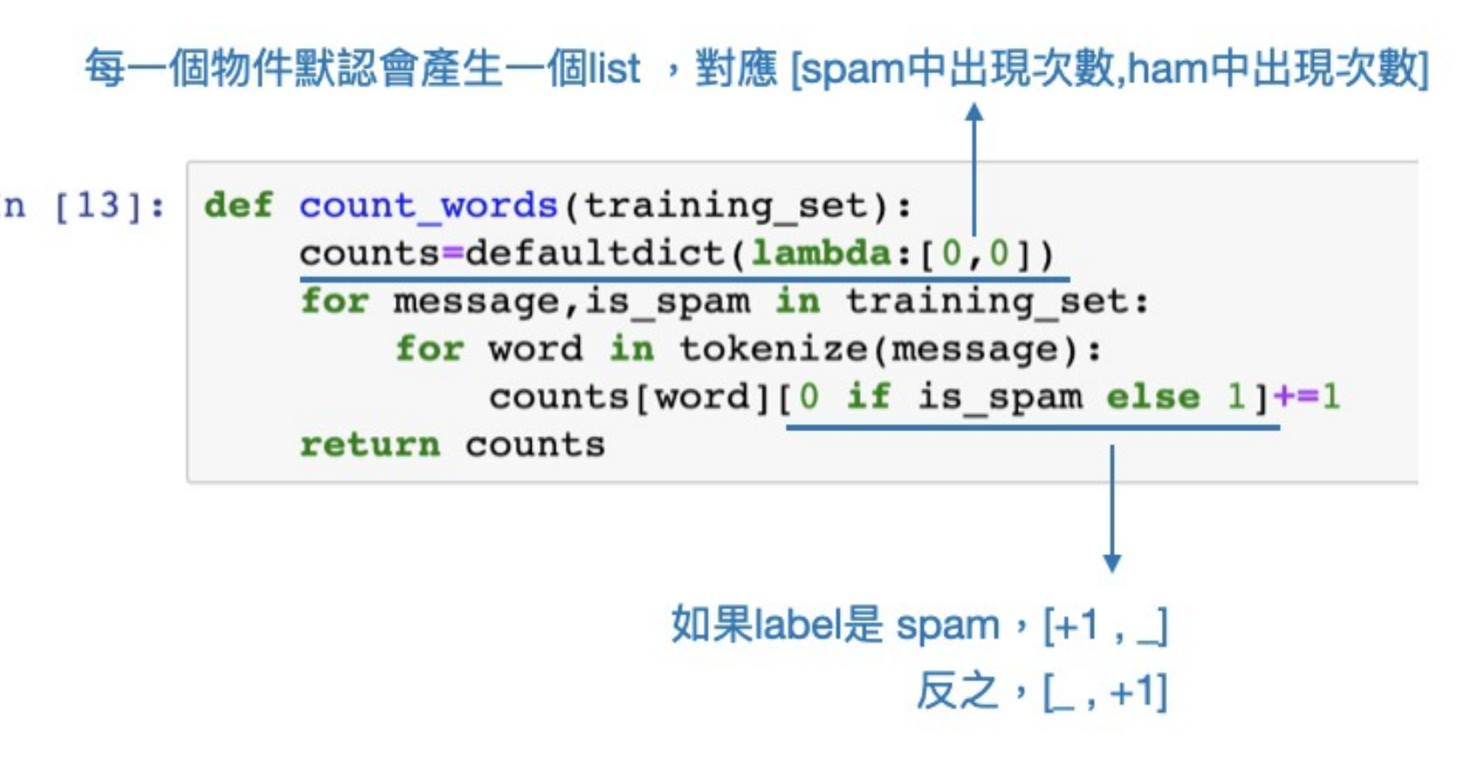
```
train_data = []
test_data = []

for x_, y_ in zip(X_train, y_train):
  train_data.append([x_, y_])

for x_, y_ in zip(X_test, y_test):
  test_data.append([x_, y_])
```

創立一個字典

紀錄每個單詞在 spam / ham 中出現的次數



每一個物件默認會產生一個list ，對應 [spam中出現次數,ham中出現次數]

```
In [13]: def count_words(training_set):
             counts=defaultdict(lambda:[0,0])
             for message,is_spam in training_set:
                 for word in tokenize(message):
                     counts[word][0 if is_spam else 1]+=1
             return counts
```

如果label是 spam，[+1 , _]
反之，[_ , +1]

```
def count_words(training_set):
  counts=defaultdict(lambda:[0,0])
  for message,is_spam in training_set:
    for word in tokenize(message):
      counts[word][0 if is_spam else 1]+=1
  return counts
```

計算 p(w|spam) / p(w|non_spam)

K是為了避免分母/分子為0，0.5為超參數可自行設計

- 其中K為超參數，為了確保分母/分子皆不為0

```
In [14]: def word_probabilities(counts,total_spams,total_non_spams,k=0.5):
             #獲得三組數據，分別為w這個字，p(w|spam)，p(w|non_spam)
             #counts[w][0]=spam,counts[w][1]=non_spam
             return [(w,(counts[w][0]+k)/(total_spams+2*k),(counts[w][1]+k)/(total_non_spams+2*k)) for w in counts]
```

分母乘2是為了降低k值對原本數值的影響。

```
def word_probabilities(counts,total_spams,total_non_spams,k=0.5):
  #獲得三組數據，分別為w這個字，p(w|spam)，p(w|non_spam)
  #counts[w][0]=spam,counts[w][1]=non_spam
  return [(w,(counts[w][0]+k/(total_spams+2*k),(counts[w][1]+k)/(total_non_spams+2*k)) for w in cou
```

計算貝氏

- 首先先將輸入的訊息 tokenize
- 初始化 spam / not_spam probability (設為0)

計算貝氏結果

```
In [16]: def spam_probability(word_probs,message, is_spam_probability, is_not_spam_probability):
             #先把這個mail的文字處理一下
             message_words=tokenize(message)
             #初始化值
             log_prob_if_spam=log_prob_if_not_spam=0.0
             #將w這個字，p(w|spam)，p(w|non_spam)依序引入
             for word,prob_if_spam,prob_if_not_spam in word_probs:
                 if word in message_words:
                     #假如這個字有在這個mail中出現
                     #把他的p(w|spam)轉log值加上log_prob_if_spam
                     log_prob_if_spam=log_prob_if_spam+math.log(prob_if_spam)
                     #把他的p(w|non_spam)轉log值加上log_prob_if_not_spam
                     log_prob_if_not_spam=log_prob_if_not_spam+math.log(prob_if_not_spam)
                 else:
                     #如果沒出現log_prob_if_spam+上得值就是1-p(w|spam)也就是這封信是垃圾郵件但是w這個字卻沒在裡面
                     log_prob_if_spam=log_prob_if_spam+math.log(1-prob_if_spam)
                     log_prob_if_not_spam=log_prob_if_not_spam+math.log(1-prob_if_not_spam)
             log_prob_if_spam = log_prob_if_spam + math.log(is_spam_probability)
             log_prob_if_not_spam = log_prob_if_not_spam + math.log(is_not_spam_probability)

             #把+起來的值轉成exp再算NaiveBayes
             prob_if_spam=math.exp(log_prob_if_spam)
             prob_if_not_spam=math.exp(log_prob_if_not_spam)
             #貝氏
             return prob_if_spam/(prob_if_spam+prob_if_not_spam)
```

```
def spam_probability(word_probs,message, is_spam_probability, is_not_spam_probability):
  #先把這個mail的文字處理一下
  message_words=tokenize(message)
  #初始化值
  log_prob_if_spam=log_prob_if_not_spam=0.0
  #將w這個字，p(w|spam)，p(w|non_spam)依序引入
  for word,prob_if_spam,prob_if_not_spam in word_probs:
    if word in message_words:
      #假如這個字有在這個mail中出現
      #把他的p(w|non_spam)轉log值加上log_prob_if_not_spam
      log_prob_if_not_spam=log_prob_if_not_spam+math.log(prob_if_not_spam)
    else:
```

下溢

$$\boxed{P(X1 \mid Y_{normal})} * \boxed{P(X2 \mid Y_{normal})} * \boxed{P(X3 \mid Y_{normal})} \ldots * P(Y_{normal})$$

- 然而接連的小數相乘會出現下溢的情況。
- 下溢：假設一段訊息內有 100 個字，每個字在 spam 中出現的機率是 0.1，那相乘就為 0.1**100，由於電腦紀錄數值是用有限浮點數儲存，太小的數值會導致訊息無法正確儲存。

- 因此我們可以將機率取 log，相加後再使用 exp 復原。
- 之所以可以這樣操作是由於連續數值相乘等於取對數(log)相加後再取指數(exponential)

計算貝氏結果

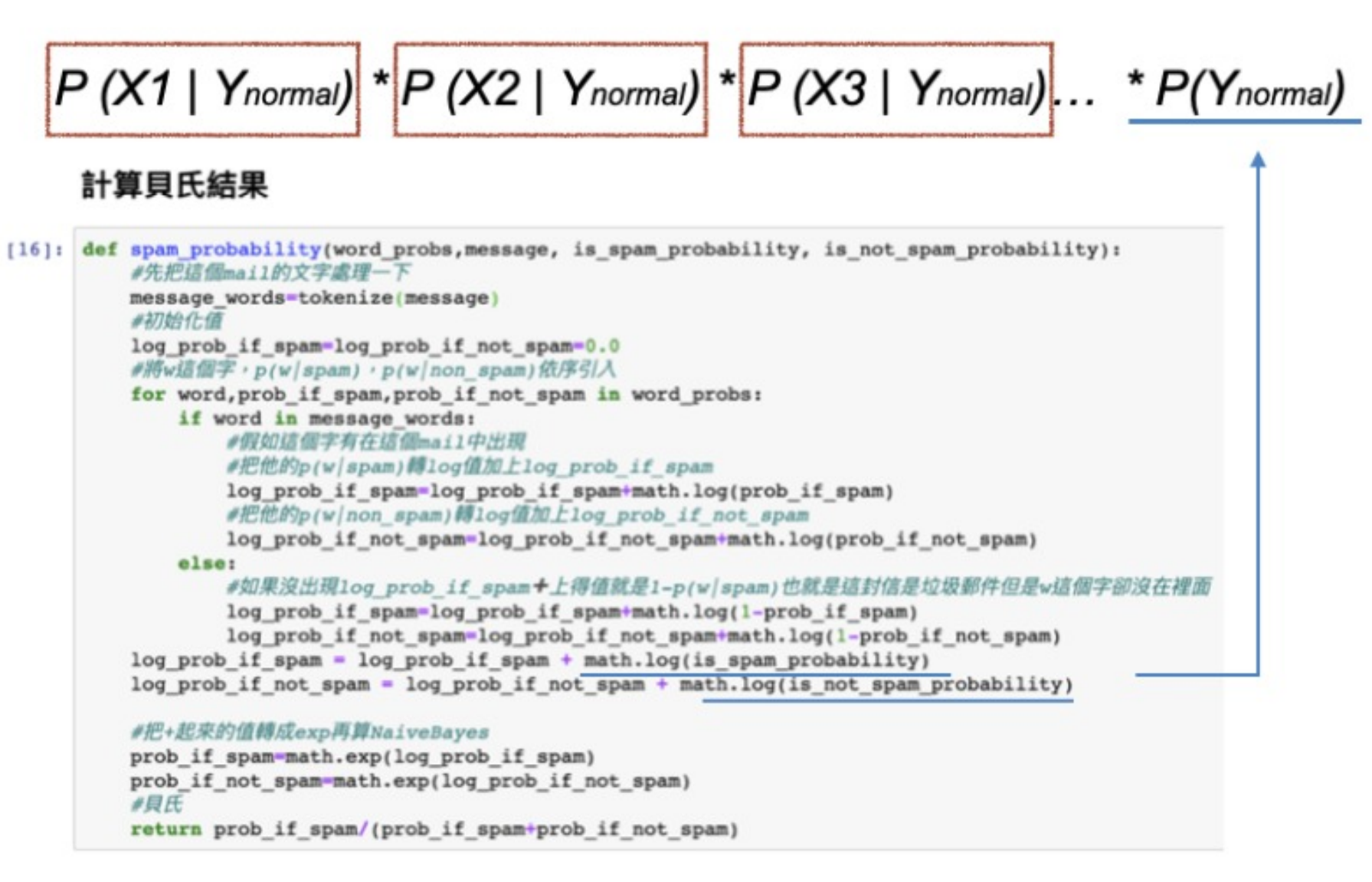
```
In [16]: def spam_probability(word_probs,message, is_spam_probability, is_not_spam_probability):
             #先把這個mail的文字處理一下
             message_words=tokenize(message)
             #初始化值
             log_prob_if_spam=log_prob_if_not_spam=0.0
             #將w這個字，p(w|spam)，p(w|non_spam)依序引入
             for word,prob_if_spam,prob_if_not_spam in word_probs:
                 if word in message_words:
                     #假如這個字有在這個mail中出現
                     #把他的p(w|spam)轉log值加上log_prob_if_spam
                     log_prob_if_spam=log_prob_if_spam+math.log(prob_if_spam)
                     #把他的p(w|non_spam)轉log值加上log_prob_if_not_spam
                     log_prob_if_not_spam=log_prob_if_not_spam+math.log(prob_if_not_spam)
                 else:
                     #如果沒出現log_prob_if_spam+上得值就是1-p(w|spam)也就是這封信是垃圾郵件但是w這個字卻沒在裡面
                     log_prob_if_spam=log_prob_if_spam+math.log(1-prob_if_spam)
                     log_prob_if_not_spam=log_prob_if_not_spam+math.log(1-prob_if_not_spam)
             log_prob_if_spam = log_prob_if_spam + math.log(is_spam_probability)
             log_prob_if_not_spam = log_prob_if_not_spam + math.log(is_not_spam_probability)

             #把+起來的值轉成exp再算NaiveBayes
             prob_if_spam=math.exp(log_prob_if_spam)
             prob_if_not_spam=math.exp(log_prob_if_not_spam)
             #貝氏
             return prob_if_spam/(prob_if_spam+prob_if_not_spam)
```

```
def spam_probability(word_probs,message, is_spam_probability, is_not_spam_probability):
  #先把這個mail的文字處理一下
  message_words=tokenize(message)
  #初始化值
  log_prob_if_spam=log_prob_if_not_spam=0.0
  #將w這個字，p(w|spam)，p(w|non_spam)依序引入
  for word, prob_if_spam,prob_if_not_spam in word_probs:
    if word in message_words:
      #假如這個字有在這個mail中出現
      #把他的p(w|spam)轉log值加上log_prob_if_spam
      log_prob_if_spam=log_prob_if_spam+math.log(prob_if_spam)
      #把他的p(w|non_spam)轉log值加上log_prob_if_not_spam
```

計算貝氏

這邊我們可以參考上一章公式



$$\boxed{P(X1 \mid Y_{normal})} * \boxed{P(X2 \mid Y_{normal})} * \boxed{P(X3 \mid Y_{normal})} \ldots * P(Y_{normal})$$

計算貝氏結果

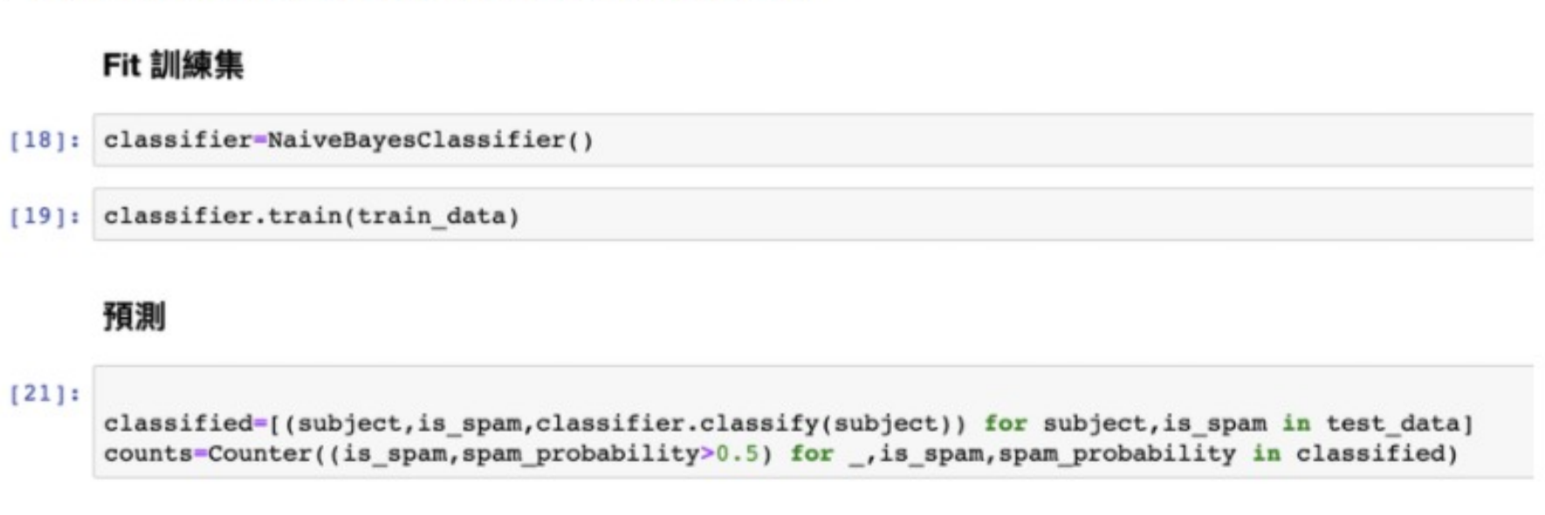
```
In [16]: def spam_probability(word_probs,message, is_spam_probability, is_not_spam_probability):
             #先把這個mail的文字處理一下
             message_words=tokenize(message)
             #初始化值
             log_prob_if_spam=log_prob_if_not_spam=0.0
             #將w這個字，p(w|spam)，p(w|non_spam)依序引入
             for word,prob_if_spam,prob_if_not_spam in word_probs:
                 if word in message_words:
                     #假如這個字有在這個mail中出現
                     #把他的p(w|spam)轉log值加上log_prob_if_spam
                     log_prob_if_spam=log_prob_if_spam+math.log(prob_if_spam)
                     #把他的p(w|non_spam)轉log值加上log_prob_if_not_spam
                     log_prob_if_not_spam=log_prob_if_not_spam+math.log(prob_if_not_spam)
                 else:
                     #如果沒出現log_prob_if_spam+上得值就是1-p(w|spam)也就是這封信是垃圾郵件但是w這個字卻沒在裡面
                     log_prob_if_spam=log_prob_if_spam+math.log(1-prob_if_spam)
                     log_prob_if_not_spam=log_prob_if_not_spam+math.log(1-prob_if_not_spam)
             log_prob_if_spam = log_prob_if_spam + math.log(is_spam_probability)
             log_prob_if_not_spam = log_prob_if_not_spam + math.log(is_not_spam_probability)

             #把+起來的值轉成exp再算NaiveBayes
             prob_if_spam=math.exp(log_prob_if_spam)
             prob_if_not_spam=math.exp(log_prob_if_not_spam)
             #貝氏
             return prob_if_spam/(prob_if_spam+prob_if_not_spam)
```

```
def spam_probability(word_probs,message, is_spam_probability, is_not_spam_probability):
  #先把這個mail的文字處理一下
  message_words=tokenize(message)
  #初始化值
  log_prob_if_spam=log_prob_if_not_spam=0.0
  #將w這個字，p(w|spam)，p(w|non_spam)依序引入
  for word, prob_if_spam,prob_if_not_spam in word_probs:
    if word in message_words:
      #假如這個字有在這個mail中出現
      #把他的p(w|spam)轉log值加上log_prob_if_spam
      log_prob_if_spam=log_prob_if_spam+math.log(prob_if_spam)
      #把他的p(w|non_spam)轉log值加上log_prob_if_not_spam
```

訓練與預測

- 訓練: 建構字典並將 p (w/spam)、 p(w|non_spam)、p(spam)、p(non_spam)的值算出。
- 預測: 我們這裡用大於小於0.5區分是否為垃圾郵件。



Fit 訓練集

```
In [18]: classifier=NaiveBayesClassifier()

In [19]: classifier.train(train_data)
```

預測

```
In [21]:
         classified=[(subject,is_spam,classifier.classify(subject)) for subject,is_spam in test_data]
         counts=Counter((is_spam,spam_probability>0.5) for _,is_spam,spam_probability in classified)
```

```
classifier=NaiveBayesClassifier()
```

```
classifier.train(train_data)
```

```
classified=[(subject,is_spam,classifier.classify(subject)) for subject, is_spam in test_data]
counts=Counter((is_spam,spam_probability>0.5) for _,is_spam,spam_probability in classified)
```

Metrics

最基本的貝氏模型也能獲得不錯的效果

```
In [17]: counts

Out[17]: Counter({(False, False): 578,
                  (True, True): 57,
                  (False, True): 9,
                  (True, False): 41})

In [25]: precision=counts[(True, True)]/(counts[(True, True)]+counts[(False, True)])
         recall=counts[(True, True)]/(counts[(True, True)]+counts[(True, False)])
         binary_accuracy = (counts[(True, True)]+counts[(False, False)])/(counts[(False, True)]+counts[(False, False)]+counts[(T
         print('accuracy : {:.2f}%'.format(binary_accuracy*100))
         print('precision : {:.2f}%'.format(precision*100))
         print('recall : {:.2f}%'.format(recall*100))

         accuracy : 92.70%
         precision : 86.36%
         recall : 58.16%
```

```
counts
```

```
precision=counts[(True, True)]/(counts[(True, True)]+counts[(False, True)])
recall=counts[(True, True)]/(counts[(True, True)]+counts[(True, False)])
binary_accuracy = (counts[(True, True)]+counts[(False, False)])/(counts[(False, True)]+counts[(Fals
print('accuracy : {:.2f}%'.format(binary_accuracy*100))
print('precision : {:.2f}%'.format(precision*100))
print('recall : {:.2f}%'.format(recall*100))
```

重點回顧

本日課程帶各位學員理解了 Naive Bayes 的實作，下一章節會更深入的討論其優缺點，以及使用 scikit-learn API。

下一步：完成作業