HW5.1. Spam Detection

Bayesian Spam Filtering

A common way to detect spam in emails is to use a *Naive Bayes classifier*, which identifies key words that are frequently used in spam messages and then calculates the probability that a given message is either spam or not spam using the words contained in the message.

If you are interested, you can read more about Bayes' theorem and Naive Bayes spam filtering, but you do not need to understand it to do this problem.

As always, feel free to work on this incrementally to check your progress. You have unlimited attempts for this problem.

1. You are given a dataset of sample messages as `spam.csv`, using the Pandas library read this file into a DataFrame named `df`. This dataset gives text messages and their classifications, either `spam` or `ham` (not spam).

   Here is part of the dataset so you can get an idea of the values:

	Label	Text
0	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives around here though

   5 rows x 2 columns

2. At this point, lets split the dataset into spam and ham values. Save the rows that have only spam or ham into the variables `spam` and `ham`.

   We can use this information to approximate the overall probability of receiving a spam or ham message, $P(S)$ and $P(H)$, using:

   $$\frac{\text{number of messages}}{\text{total messages}}$$

   Calculate these values and store as `prob_spam` and `prob_ham`.

3. We will want to decompose a message into a set of words, or *tokens*. Implement the function `tokenize_text` below, which should take as input a string and return a set of tokens. Use NLTK's `word_tokenize()` to split a string into a list of words. Then, remove anything that is not a word (`str.isalpha()` may be helpful).

   To help identify words, we can reduce them to their *lemma*, which is the base form of the word. For example, the words "produced" and "production" have the same lemma, "produce", this will reduce the

Homework 5

Assessment overview

Total points: 20/20

Score: 100%

Question

Value: 20

History: 20 20

Awarded points: 20/20

Report an error in this question

Previous question

Next question

Attached files

No attached files

Attach a file

Attach text

number of redundant entries we get. For this we can use NLTK's WordNetLemmatizer; replace all tokens with their lower-cased lemma.

```
# Example code
lemmatizer = WordNetLemmatizer()
lemma = lemmatizer.lemmatize("rock")
```

4. Now, we can calculate the probability of finding a word in a spam or a ham message. This is equivalent to finding the percentage that each word occurs in either. Loop through each word in each message, and if it is spam increase the word's spam probability by $1/N$ (where N is the total number of messages), and if it is ham increase the word's ham probability by the same amount. This is a similar method to generating the probabilities in Markov chains in the lesson activity.

   Create the two dictionaries, `prob_spam_words` and `prob_ham_words` to store the probability of each word. Each dictionary should map a string token to its probability of showing up in a spam or ham message. You can do this by looping over the tokens in each message and either incrementing the probability by $1/N$ if the word exists in the dictionary, or setting the probability to $1/N$ if it does not exist yet.

5. After generating all the above probabilities, we now have all the information needed to classify a piece of text as spam or not.

   You are given the function `get_class_probability()` that will give you the probability of getting a body of text given it is ham or spam, $P(\mathbf{W} \mid S)$ and $P(\mathbf{W} \mid H)$. Using Bayes' theorem, we can turn this into the probability that some piece of text is spam or not:

   $$P(S \mid \mathbf{W}) \propto P(\mathbf{W} \mid S)\, P(S)$$

   $$P(H \mid \mathbf{W}) \propto P(\mathbf{W} \mid H)\, P(H)$$

   (Because we are comparing both probabilities against each other, we can drop the denominators since they're the same.)

   To find the probability a message is spam, we can multiply the probability given by `get_class_probability()` by the overall probability of spam. You can get a similar result for the probability of ham. To classify a message, we can just check which probability of the two is higher. I.e. if the spam probability is larger, then the message is classified as spam.

   You are given an example named `test_message` to classify. Tokenize the message and find the probability that it is spam and ham, save these as `prob_msg_spam` and `prob_msg_ham`. Finally, determine if it is spam or not and save this result in `is_spam`.

You are given the following variable:

Name	Type	Description
`test_message`	string	Text to classify

Your code snippet should define the following variables:

Name	Type	Description
`df`	pandas dataframe	Full dataset
`tokenize_text`	function	Function to clean a message and split it into tokens
`spam`	pandas dataframe	Dataset of only spam messages
`ham`	pandas dataframe	Dataset of only ham messages
`prob_spam`	decimal number	Probability of having a spam message
`prob_ham`	decimal number	Probability of having a ham message
`prob_spam_words`	dictionary	Probability of having a specific word, given a spam message
`prob_ham_words`	dictionary	Probability of having a specific word, given a ham message
`prob_msg_spam`	decimal number	Probability that the test message is spam
`prob_msg_ham`	decimal number	Probability that the test message is ham
`is_spam`	boolean	Is the test message spam?

user_code.py

```python
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

# 1. Load dataset

df = pd.read_csv("spam.csv")

# 2. Split Spam and Ham

spam = df[df['Label'] == 'spam']
ham = df[df['Label'] == 'ham']

num_total = len(spam.to_numpy()) + len(ham.to_numpy())
num_spam = len(spam.to_numpy())
num_ham = len(ham.to_numpy())

prob_spam = float(num_spam)/num_total
prob_ham = float(num_ham)/num_total




# 3. Create tokenizer function

```

```python
def tokenize_text(message):
    """
    message: A string containing a single sentence.
    returns: A list of tokens.
    """
    # Implement me!
    # - Tokenize the message
    # - Remove non word tokens and stopwords
    # - Convert tokens to lemmas
    # - Convert everything to lowercase
    stopWords = set(stopwords.words('english'))
    message = nltk.word_tokenize(message)
    ret = []

    for word in message:

        if str.isalpha(word) and word not in stopWords:

            lemmatizer = WordNetLemmatizer()
            lemma = lemmatizer.lemmatize(word)
            lemma = lemma.lower()
            ret.append(lemma)

    return ret

# an example to test your function:
print(tokenize_text(df.values[0, 1]))
# the expected output:
print(['go', 'jurong', 'point', 'available', 'bugis', 'n',
    'great', 'world', 'la', 'e', 'buffet', 'cine', 'got',
    'amore', 'wat'])

# 4. Get Word Probabilities

prob_spam_words = {}
prob_ham_words = {}


# Uncomment this once you get to it.

for text in spam.values[:,1]:
    tokens = tokenize_text(text)
    for token in tokens:
        if token not in prob_spam_words:
            prob_spam_words[token] = 1./num_total
        else:
            prob_spam_words[token] += 1./num_total

for text in ham.values[:,1]:
    tokens = tokenize_text(text)
    for token in tokens:
        if token not in prob_ham_words:
            prob_ham_words[token] = 1./num_total
        else:
            prob_ham_words[token] += 1./num_total


# 5. Classify message

def get_class_probability(tokens, probabilities):
    eta = 0
    for token in tokens:
        p = 1e-4
```

```
        if token in probabilities:
            p = probabilities[token]
        eta += np.log(p)
    return np.exp(eta)



pws = get_class_probability(test_message,prob_spam_words)
prob_msg_spam = pws*prob_spam
phs = get_class_probability(test_message,prob_ham_words)
prob_msg_ham = phs*prob_ham
is_spam = False if prob_msg_spam > prob_msg_ham else True

```

Restore original file

Save & Grade   Save only

Submitted answer 30 correct: 100%

Submitted at 2019-12-03 17:17:52 (CST)

hide

Score: 9/9 (100%)

Output

```
['go', 'jurong', 'point', 'available', 'bugis', 'n',
'great', 'world', 'la', 'e', 'buffet', 'cine', 'got',
'amore', 'wat']
['go', 'jurong', 'point', 'available', 'bugis', 'n',
'great', 'world', 'la', 'e', 'buffet', 'cine', 'got',
'amore', 'wat']
```

Test Results

✔ **[1/1] Check df**

✔ **[1/1] Check tokenize_text**

✔ **[1/1] Check spam**

✔ **[1/1] Check ham**

✔ **[1/1] Check prob_spam**

✔ **[1/1] Check prob_ham**

✔ **[1/1] Check prob_spam_words**

✓ [1/1] Check prob_msg_ham

✓ [1/1] Check is_spam

✓ [0/0] Student Code

Submitted answer 29 partially correct: 88%
Submitted at 2019-12-03 17:16:42 (CST)
show

Submitted answer 28 correct: 100%
Submitted at 2019-12-03 17:15:31 (CST)
show

Submitted answer 27 partially correct: 88%
Submitted at 2019-12-03 17:13:47 (CST)
show

Submitted answer 26 partially correct: 88%
Submitted at 2019-12-03 17:11:31 (CST)
show

Submitted answer 25 incorrect: 0%
Submitted at 2019-12-03 17:10:52 (CST)
show

Submitted answer 24 incorrect: 0%
Submitted at 2019-12-03 17:09:44 (CST)
show

Submitted answer 23 partially correct: 77%
Submitted at 2019-12-03 16:55:30 (CST)
show

Submitted answer 22 partially correct: 66%
Submitted at 2019-12-03 16:45:22 (CST)
show

Submitted answer 21 partially correct: 55%
Submitted at 2019-12-03 16:37:05 (CST)
show

Submitted answer 20 partially correct: 55%
Submitted at 2019-12-03 16:30:31 (CST)
show

Submitted answer 19 **partially correct: 55%**
Submitted at 2019-12-03 16:29:38 (CST)
show

Submitted answer 18 **partially correct: 55%**
Submitted at 2019-12-03 16:25:16 (CST)
show

Submitted answer 17 **partially correct: 55%**
Submitted at 2019-12-03 16:24:13 (CST)
show

Submitted answer 16 **partially correct: 55%**
Submitted at 2019-12-03 16:15:46 (CST)
show

Submitted answer 15 **partially correct: 55%**
Submitted at 2019-12-03 16:14:15 (CST)
show

Submitted answer 14 saved, not graded
Submitted at 2019-12-03 16:13:19 (CST)
show

Submitted answer 13 incorrect: 0%
Submitted at 2019-12-03 12:15:35 (CST)
show

Submitted answer 12 **partially correct: 55%**
Submitted at 2019-12-03 12:12:12 (CST)
show

Submitted answer 11 **partially correct: 55%**
Submitted at 2019-12-03 12:10:22 (CST)
show

Submitted answer 10 **partially correct: 55%**
Submitted at 2019-12-03 12:09:08 (CST)
show

Submitted answer 9 **partially correct: 55%**
Submitted at 2019-12-03 12:05:32 (CST)
show

Submitted answer 8 **partially correct: 33%**
Submitted at 2019-12-03 12:04:49 (CST)
show

Submitted answer 7 **partially correct: 11%**

show

Submitted at 2019-12-03 12:03:27 (CST)

Submitted answer 6 **partially correct: 11%**

Submitted at 2019-12-03 12:02:20 (CST)

show

Submitted answer 5 **partially correct: 11%**

Submitted at 2019-12-03 11:58:19 (CST)

show

Submitted answer 4 **partially correct: 11%**

Submitted at 2019-12-03 11:56:58 (CST)

show

Submitted answer 3 **partially correct: 11%**

Submitted at 2019-12-03 11:54:00 (CST)

show

Submitted answer 2 **incorrect: 0%**

Submitted at 2019-12-03 11:53:25 (CST)

show

Submitted answer 1 **partially correct: 11%**

Submitted at 2019-12-03 11:48:56 (CST)

show