DAT405 Introduction to Data Science and AI

2022-2023, Reading Period 3

Assignment 4: Spam classification using Naïve Bayes

This assignmetn has three obligatory questions which will be grades as PASS/FAIL. Questions 4-5 are optional and will not be graded, but can be interesting for students aiming for higher grades.

The exercise takes place in this notebook environment where you can chose to use Jupyter or Google Colabs. We recommend you use Google Colabs as it will facilitate remote group-work and makes the assignment less technical. Hints: You can execute certain linux shell commands by prefixing the command with `!`. You can insert Markdown cells and code cells. The first you can use for documenting and explaining your results the second you can use writing code snippets that execute the tasks required.

In this assignment you will implement a Naïve Bayes classifier in Python that will classify emails into spam and non-spam ("ham") classes. Your program should be able to train on a given set of spam and "ham" datasets. You will work with the datasets available at https://spamassassin.apache.org/old/publiccorpus/. There are three types of files in this location:

- easy-ham: non-spam messages typically quite easy to differentiate from spam messages.
- hard-ham: non-spam messages more difficult to differentiate
- spam: spam messages

Execute the cell below to download and extract the data into the environment of the notebook -- it will take a few seconds. If you chose to use Jupyter notebooks you will have to run the commands in the cell below on your local computer, with Windows you can use 7zip (https://www.7-zip.org/download.html) to decompress the data.

What to submit: Convert the notebook to a pdf-file and submit it. Make sure all cells are executed so all your code and its results are included. Double check the pdf displays correctly before you submit it.

```
#Download and extract data
!wget https://spamassassin.apache.org/old/publiccorpus/20021010_easy_ham.tar.bz2
!wget https://spamassassin.apache.org/old/publiccorpus/20021010_hard_ham.tar.bz2
!wget https://spamassassin.apache.org/old/publiccorpus/20021010_spam.tar.bz2
!tar -xjf 20021010_easy_ham.tar.bz2
!tar -xjf 20021010_hard_ham.tar.bz2
!tar -xjf 20021010_spam.tar.bz2
```

```
--2023-02-10 11:08:08--  https://spamassassin.apache.org/old/publiccorpus/20021010_easy_ham.tar.bz2
Resolving spamassassin.apache.org (spamassassin.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to spamassassin.apache.org (spamassassin.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1677144 (1.6M) [application/x-bzip2]
Saving to: '20021010_easy_ham.tar.bz2'

20021010_easy_ham.t 100%[===================>]   1.60M  --.-KB/s    in 0.07s

2023-02-10 11:08:08 (23.5 MB/s) - '20021010_easy_ham.tar.bz2' saved [1677144/1677144]

--2023-02-10 11:08:09--  https://spamassassin.apache.org/old/publiccorpus/20021010_hard_ham.tar.bz2
Resolving spamassassin.apache.org (spamassassin.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to spamassassin.apache.org (spamassassin.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1021126 (997K) [application/x-bzip2]
Saving to: '20021010_hard_ham.tar.bz2'

20021010_hard_ham.t 100%[===================>] 997.19K  --.-KB/s    in 0.005s

2023-02-10 11:08:09 (191 MB/s) - '20021010_hard_ham.tar.bz2' saved [1021126/1021126]

--2023-02-10 11:08:10--  https://spamassassin.apache.org/old/publiccorpus/20021010_spam.tar.bz2
Resolving spamassassin.apache.org (spamassassin.apache.org)... 151.101.2.132, 2a04:4e42::644
Connecting to spamassassin.apache.org (spamassassin.apache.org)|151.101.2.132|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1192582 (1.1M) [application/x-bzip2]
Saving to: '20021010_spam.tar.bz2'

20021010_spam.tar.b 100%[===================>]   1.14M  --.-KB/s    in 0.005s

2023-02-10 11:08:10 (213 MB/s) - '20021010_spam.tar.bz2' saved [1192582/1192582]
```

The data is now in the three folders `easy_ham`, `hard_ham`, and `spam`.

```
!ls -lah
```

```
total 9.5M
drwxrwxrwx 7 root root   15 Feb  8 17:09 .
drwxr-xr-x 1 root root  186 Feb  8 18:53 ..
-rw-r--r-- 1 root root 1.6M Jun 29  2004 20021010_easy_ham.tar.bz2
-rw-r--r-- 1 root root 1.6M Jun 29  2004 20021010_easy_ham.tar.bz2.1
-rw-r--r-- 1 root root 998K Dec 16  2004 20021010_hard_ham.tar.bz2
-rw-r--r-- 1 root root 998K Dec 16  2004 20021010_hard_ham.tar.bz2.1
-rw-r--r-- 1 root root 1.2M Jun 29  2004 20021010_spam.tar.bz2
-rw-r--r-- 1 root root 1.2M Jun 29  2004 20021010_spam.tar.bz2.1
-rw-rw-r-- 1 root root 914K Feb  3 09:23 data_assignment3.csv
drwxrwxr-x 2 root root    3 Feb  3 09:31 .deepnote
drwxr-xr-x 2 root root    3 Feb  1 09:41 deepnote_exports
drwx--x--x 2  500  500 2.5K Oct 10  2002 easy_ham
drwx--x--x 2 1000 1000  252 Dec 16  2004 hard_ham
-rw-rw-r-- 1 root root 2.0M Jan 20 07:47 life-expectancy-vs-gdp-per-capita.csv
drwxr-xr-x 2  500  500  503 Oct 10  2002 spam
```

Jihad Almahal(5 hrs) Oscar Karbin (5 hrs)

1. Preprocessing:

Note that the email files contain a lot of extra information, besides the actual message. Ignore that for now and run on the entire text (in the optional part further down can experiment with filtering out the headers and footers).

1. We don't want to train and test on the same data (it might help to reflect on why if you don't recall). Split the spam and the ham datasets in a training set and a test set. (`hamtrain`, `spamtrain`, `hamtest`, and `spamtest`). Use easy_ham for quesions 1 and 2.

```
import pathlib
from sklearn.model_selection import train_test_split
import os
import numpy as np

def get_files(path):
    files = []
    for filename in os.scandir(path):
        try:
            if filename.is_file():
                with open(filename, encoding='latin-1') as f:
                    files.append(f.read())
        except:
            print("Invalid file from {}, encoding error".format(path))
    print('{} files found in {}'.format(len(files), path))
    return files


x_easy_ham = get_files('/work/easy_ham')
x_spam = get_files('/work/spam')

#Question 3
x_hard_ham = get_files('/work/hard_ham')

y_easy_ham = np.zeros(len(x_easy_ham))
y_hard_ham = np.zeros(len(x_hard_ham))
y_spam = np.ones(len(x_spam))

x_train_easy_ham, x_test_easy_ham, y_train_easy_ham, y_test_easy_ham = train_test_split(x_easy_ham, y_easy_ham, test_size=0.2, random_sta
x_train_spam, x_test_spam, y_train_spam, y_test_spam = train_test_split(x_spam, y_spam, test_size=0.2, random_state=42)
x_train_hard_ham, x_test_hard_ham, y_train_hard_ham, y_test_hard_ham = train_test_split(x_hard_ham, y_hard_ham, test_size=0.2, random_sta

x_train = np.concatenate((x_train_easy_ham,  x_train_spam))
y_train = np.concatenate((y_train_easy_ham, y_train_spam))
x_test = np.concatenate((x_test_easy_ham,  x_test_spam))
y_test = np.concatenate((y_test_easy_ham,  y_test_spam))

#Question 3
x_train_hard = np.concatenate((x_train_hard_ham, x_train_spam))
y_train_hard = np.concatenate((y_train_hard_ham, y_train_spam))
x_test_hard = np.concatenate((x_test_hard_ham, x_test_spam))
y_test_hard = np.concatenate((y_test_hard_ham, y_test_spam))
```

```
2551 files found in /work/easy_ham
501 files found in /work/spam
250 files found in /work/hard_ham
```

2. Write a Python program that:

1. Uses the four datasets from Qustion 1 (`hamtrain`, `spamtrain`, `hamtest`, and `spamtest`)

2. Trains a Naïve Bayes classifier (use the scikit-learn library) on `hamtrain` and `spamtrain`, that classifies the test sets and reports True Positive and False Negative rates on the `hamtest` and `spamtest` datasets. You can use `CountVectorizer` (Documentation here) to transform the email texts into vectors. Please note that there are different types of Naïve Bayes Classifier in scikit-learn (Documentation here). Test two of these classifiers that are well suited for this problem:

- Multinomial Naive Bayes
- Bernoulli Naive Bayes.

Please inspect the documentation to ensure input to the classifiers is appropriate before you start coding.

```
#Write your code here
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.metrics import confusion_matrix
import pandas as pd

def classify(x_train, y_train, x_test, y_test):
    vectorizer = CountVectorizer()

    # tokenize
    x_train_vec = vectorizer.fit_transform(x_train)
    x_test_vec = vectorizer.transform(x_test)

    # initialize classifiers
    mnb = MultinomialNB()
    bnb = BernoulliNB()

    # train and predict using MultinomialNB
    multinomial_nb = MultinomialNB().fit(x_train_vec, y_train)
    mnb_pred = multinomial_nb.predict(x_test_vec)
    mnb_cm = confusion_matrix(y_test, mnb_pred)

    # train and predict using BernoulliNB
    bernoulli_nb = BernoulliNB(binarize=1).fit(x_train_vec, y_train)
    bnb_pred = bernoulli_nb.predict(x_test_vec)
    bnb_cm = confusion_matrix(y_test, bnb_pred)

    return mnb_cm, bnb_cm
mnb_cm, bnb_cm = classify(x_train, y_train, x_test, y_test)
print("Confusion Matrix for MultinomialNB:")
print(pd.DataFrame(mnb_cm, columns=['pred_ham', 'pred_spam'], index=['actual_ham', 'actual_spam']))
print("\nConfusion Matrix for BernoulliNB:")
print(pd.DataFrame(bnb_cm, columns=['pred_ham', 'pred_spam'], index=['actual_ham', 'actual_spam']))
```

```
     Confusion Matrix for MultinomialNB:
                 pred_ham  pred_spam
     actual_ham        510          1
     actual_spam        12         89

     Confusion Matrix for BernoulliNB:
                 pred_ham  pred_spam
     actual_ham        509          2
     actual_spam        65         36
```

3.Run on hard ham:

Run the two models from Question 2 on spam versus hard-ham and compare to easy-ham results.

```
#Code to report results here
mnb_cm, bnb_cm = classify(x_train_hard, y_train_hard, x_test_hard, y_test_hard)
print("Confusion Matrix for MultinomialNB:")
print(pd.DataFrame(mnb_cm, columns=['pred_ham', 'pred_spam'], index=['actual_ham', 'actual_spam']))
print("\nConfusion Matrix for BernoulliNB:")
print(pd.DataFrame(bnb_cm, columns=['pred_ham', 'pred_spam'], index=['actual_ham', 'actual_spam']))
```

```
     Confusion Matrix for MultinomialNB:
                 pred_ham  pred_spam
     actual_ham         37         13
     actual_spam         2         99

     Confusion Matrix for BernoulliNB:
                 pred_ham  pred_spam
     actual_ham         26         24
     actual_spam         0        101
```

Comparing the results of easy_ham vs spam and hard_ham vs spam:

For the MultinomialNB classifier, the results show that it has a higher accuracy in classifying easy ham emails as ham compared to hard ham emails as ham. In the easy_ham vs spam classification, it correctly classified 510 emails as ham and only misclassified 1 email as spam. On the other hand, in the hard_ham vs spam classification, it only correctly classified 37 emails as ham and misclassified 13 emails as spam.

For the BernoulliNB classifier, the results show that it has a higher accuracy in classifying hard ham emails as ham compared to easy ham emails as ham. In the easy_ham vs spam classification, it correctly classified 509 emails as ham and misclassified 2 emails as spam. On the other hand, in the hard_ham vs spam classification, it correctly classified 26 emails as ham and misclassified 24 emails as spam.

In comparing the results of the MultinomialNB and BernoulliNB classifiers for classifying easy_ham and hard_ham emails as ham or spam, it's interesting to note that the performance of both classifiers is worse for hard_ham emails compared to easy_ham emails.

One reason for this could be a domain shift, where the distribution of features in the hard_ham emails is significantly different from that in the easy_ham emails. This could be due to the fact that the hard_ham emails are more diverse in their content and come from a wider range of sources compared to the easy_ham emails, which are more uniform in their content and come from a more limited set of sources.

Another reason is that the features that the classifiers find important are not as effective in capturing the information needed to classify hard_ham emails correctly. For example, the vocabulary used in the hard_ham emails is more diverse and includes more rare or uncommon words that the classifiers are not trained to recognize as important for distinguishing between ham and spam emails. In contrast, the vocabulary used in the easy_ham emails may be more common and well-represented in the training data, making it easier for the classifiers to classify them correctly.

Further analysis could involve examining the most important features or words used by the classifiers in making their classifications, and comparing them between the easy_ham and hard_ham datasets. This could provide insights into the differences in the content and vocabulary used in the two types of emails, and how these differences affect the performance of the classifiers.

4. OPTIONAL - NOT MARKED:

To avoid classification based on common and uninformative words it is common to filter these out.

a. Think about why this may be useful. Show a few examples of too common and too uncommon words.

b. Use the parameters in scikit-learn's `CountVectorizer` to filter out these words. Update the program from point 2 and run it on easy ham vs spam and hard ham vs spam and report your results.

```
#Write your code here
```

5. OPTIONAL - NOT MARKED: Eeking out further performance

Filter out the headers and footers of the emails before you run on them. The format may vary somewhat between emails, which can make this a bit tricky, so perfect filtering is not required. Run your program again and answer the following questions:

- Does the result improve from 3 and 4?
- What do you expect would happen if your training set were mostly spam messages while your test set were mostly ham messages or vice versa?
- Look at the `fit_prior` parameter. What does this parameter mean? Discuss in what settings it can be helpful (you can also test your hypothesis).

```
#Write your code here
```