Assignment 4 - Spam classification using Naïve Bayes

Nathan HAUDOT (10 hours), Hugo MATH (10 hours)

November 30, 2021

1 Preprocessing

For the first part of the assignment, we will train the classifiers without filtering data. We concatenate ham spam messages together in order to have a training set composed with both of them. The datasets labels are made of 1 for ham, and -1 for spam (cf. figure 1). Datasets have been split in order to obtain smaller datasets to train the classifiers, and datasets to test them.

```
# Dataset splitting

# Resulting labels
y_easy_ham = np.ones(len(x_easy_ham))
y_hard_ham = np.ones(len(x_hard_ham))
y_spam = np.ones(len(x_spam)) - 2 # -2 in order to get -1

# Concatenate ham & spam together (x & y)
x_easy_ham_spam = np.concatenate((x_easy_ham, x_spam))
x_hard_ham_spam = np.concatenate((x_hard_ham, x_spam))

y_easy_ham_spam = np.concatenate((y_easy_ham, y_spam))
y_hard_ham_spam = np.concatenate((y_hard_ham, y_spam))

# Splitting data
X_easy_train, X_easy_test, Y_easy_train, Y_easy_test = train_test_split(x_easy_ham_spam, y_easy_ham_spam, test_size=
X_hard_train, X_hard_test, Y_hard_train, Y_hard_test = train_test_split(x_hard_ham_spam, y_hard_ham_spam, test_size=
```

Figure 1: Splitting datasets into train & test sets

2 Classifiers results

2.a Easy-ham vs. spam

The classifiers are trained using default hyperparameter values from sklearn ($\alpha = 1$, $binarize = 1.0$ for BernouilliNB). We then obtain the following results (confusion matrixes have been normalized for simplified comprehension):

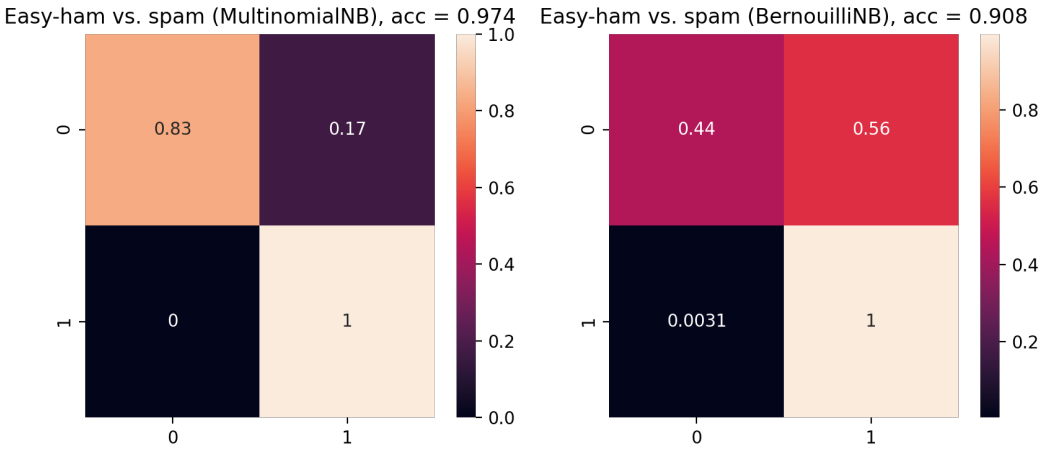

Figure 2: Confusion matrix & accuracy of Multinomial & Bernouilli classifiers vs. easy-ham (no filtering)

Note: Binarize is a threshold to binarize our vectorized words. The random variables of the law of Bernoulli by definition only takes a value of 0 or 1.

We can see that the Multinomial Naive Bayes classifier already works very well on the easy-ham with zero false-negatives score. The Bernouilli Naive Bayes classifier reports more false positives than true positives.

2.b Hard-ham vs. spam

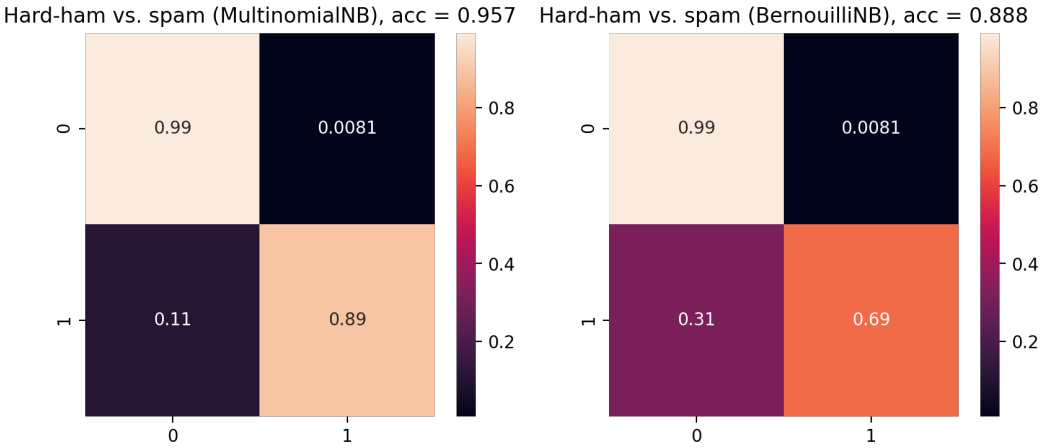

Figure 3: Confusion matrix & accuracy of Multinomial & Bernouilli classifiers vs. hard-ham (no filtering)

The Multinomial Naïve Bayes classifier does very well with very low false positives and false negatives. The Bernoulli Naïve Bayes classifier is less performing than the Multinomial Naïve Bayes classifier, with a lower accuracy. We notice that it gives many false negatives.

We can understand that the Bernoulli classification classifies the dataset with the concept of binarization, which makes us think that it is probably less accurate in its operation. Indeed, the multinomial distribution is a generalization of the binomial distribution. Thus, when binarizing the data, we lose information about our "vocabulary" obtained by the **CountVectorizer()**. We can use a smoothing technique for this classifier (parameter "alpha" in sklearn) to avoid "null frequencies" (when a data has never been seen in the training dataset).

3 Filtering most common words

One of the techniques to improve the accuracy of our models is to filter the most frequent words in our dataset: they do not contribute to differentiate classes, so it is relevant to remove them before training our models.

```
# Get most frequent words
filtered_vectorizer = CountVectorizer().fit(X_easy_train)
bag_of_words = filtered_vectorizer.transform(X_easy_train)
sum_words = bag_of_words.sum(axis=0)
words_freq = [(word, sum_words[0, idx]) for word, idx in filtered_v
words_freq =sorted(words_freq, key = lambda x: x[1], reverse=True)

# Extract most frequent words into list
stopwords = []
for word in words_freq[:500]:
    stopwords.append(word[0])

stopwords = frozenset(stopwords)

# Taking stopwords into account
filtered_vectorizer = CountVectorizer(stop_words=stopwords)
```

Figure 4: 500 most frequent words filtering in the dataset

These words are called "stopwords", and by definition, they do not contain information for our classification task. This step is often called **features selection** which consists of reducing the dataset space into a smaller one to speed up training and get better overall accuracy.

```
[('com', 33096), ('to', 22992), ('the', 21752), ('from', 19481), ('2002', 19333), ('net', 16551), ('for', 14949), ('w
ith', 14793), ('by', 14505), ('localhost', 13783), ('id', 12736), ('received', 12573), ('of', 11366), ('example', 112
93), ('and', 11264), ('list', 11141), ('11', 10257), ('org', 10184), ('sep', 9032), ('fork', 8911), ('xent', 8693),
('in', 8340), ('http', 8107), ('font', 7951), ('3d', 7807), ('esmtp', 7367), ('0100', 6865), ('is', 6332), ('127', 61
```

Figure 5: Most frequent words in the dataset

We can find that the most frequent words in our dataset are com, to, the, from, 2002, net, for, with, by... This list of words is then entered into the stop_words parameter of the CountVectorizer() function to filter them.

3.a Easy-ham vs. spam

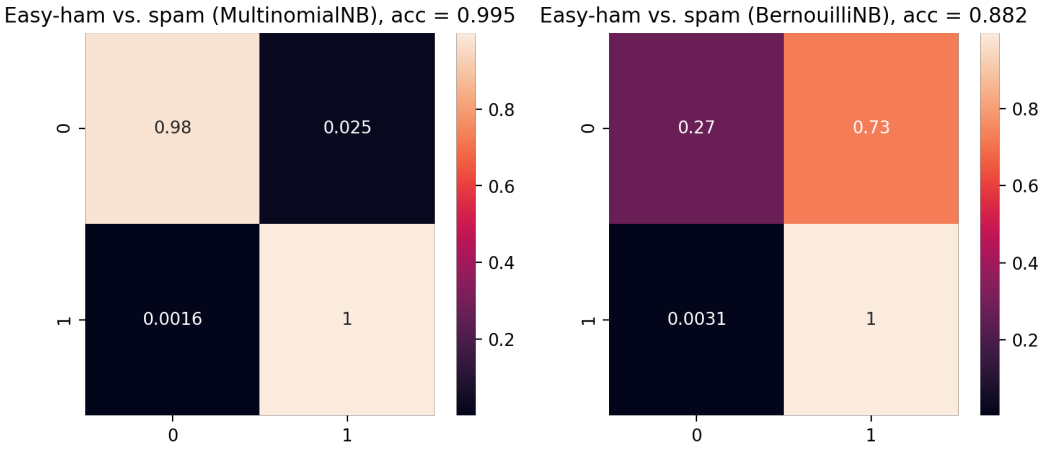

Figure 6: Confusion matrix & accuracy of Multinomial & Bernouilli classifier vs. easy-ham (filtered)

3.b Hard-ham vs. spam

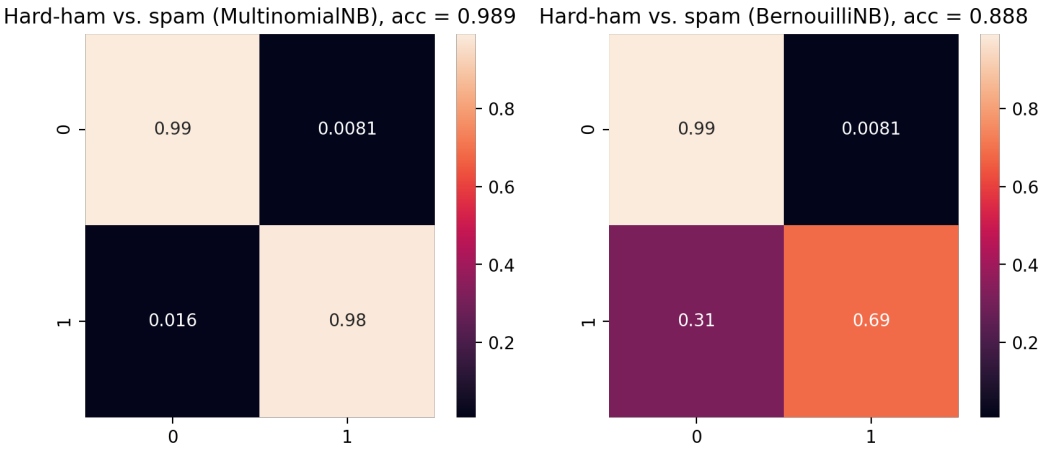

Figure 7: Confusion matrix & accuracy of Multinomial & Bernouilli classifier vs. hard-ham (filtered)

We notice that the Multinomial Naive Bayes classifier performs much better after a "most common words" filtering, we obtain a better accuracy on the two test sets easy-ham hard-ham.

However, the Bernouilli Naïve Bayes classifier does not fare as well, and even loses in accuracy on its classification. There is no difference with the hard-ham test set, but the classifier is less accurate on the easy-ham, with more false positives than before.

4 Filter out the headers and the footers

4.a a. Does the result improve from 3 and 4?

We already have an accuracy of 98.9% with the Multinomial model, moreover we did not see footer on the three email's types, and trying to filter badly the emails will result in losing information which are relevant for the training. Indeed, we want to extract as much as features as possible and only features, not noise. Headers and footers could be associated with noises, but in our case it is not relevant since we have already 98,5% on f1 score.

4.b b. The split of the data set into a training set and a test set can lead to very skewed results. Why is this, and do you have suggestions on remedies?

One major principle when it comes to prepocessing data is to **shuffle** them so that we have equal proportions (relative to their respective quantity) to the training and testing sets. Speaking of categories and proportions, we can also do a **stratification** to make sure that we get the same data distribution when splitting the dataset. It is especially useful when dealing with large amount of categories.

4.c What do you expect would happen if your training set were mostly spam messages while your test set were mostly ham messages?

The model will not learn proper features of the data. Resulting in a lot of false positive and true negative and therefore a low accuracy. Indeed, the data distribution is not representative of the real one after the splitting.

4.d Hyper-optimization

Instead of filtering the headers and footers, we have performed a simple grid search on the hyperparameters α and *binarize* of the two classifiers. We respectively got for Multinomial and Bernoulli : $\alpha = 0.1$ and $\alpha = 0.001$ $binarize = 1$.

We now have models that reach up to 99,2% of accuracy and a f1 score (mixed metric) of about 99,5% for both models.

```
Accuracy for multinomal All-Ham vs Spam
[[117   4]
 [  3 701]] : is the confusion matrix
0.9915 : is the accuracy score
0.9943 : is the precision score
0.9957 : is the recall score
0.995 : is the f1 score

Accuracy for bernoulli All-Ham vs Spam
[[116   5]
 [  1 703]] : is the confusion matrix
0.9927 : is the accuracy score
0.9929 : is the precision score
0.9986 : is the recall score
0.9958 : is the f1 score
```

```
[*] Starting hyperoptimization for Multinomial ..
Fitting 10 folds for each of 6 candidates, totalling 60 fits
[!] Best estimator MultinomialNB(alpha=0.1)
[*] Starting hyperoptimization for Bernoulli ..
Fitting 10 folds for each of 36 candidates, totalling 360 fits
[!] Best estimator BernoulliNB(alpha=0.001, binarize=1)
```

Figure 8: Grid search for two hyper-parameters