Assignment 4 - Bayesian models
Group 62
Fredrik Lilliecreutz & Erling Hjermstad

Time spent on the module:
Fredrik Lilliecreutz: 10 hours
Erling Hjermstad: 10 hours

1.Preprocessing:
a.
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. Further down (in the higher grade part), you will be asked to filter out the headers and footers.

b.
We don't want to train and test on the same data. Split the spam and the ham datasets in a training set and a test set.
As the website offered multiple datasets of each category, we used the predetermined split and sorted them into either a Test- or Train-folder. This gave an approx ⅔ train and ⅓ test split over all categories.

2.
Write a Python program that:

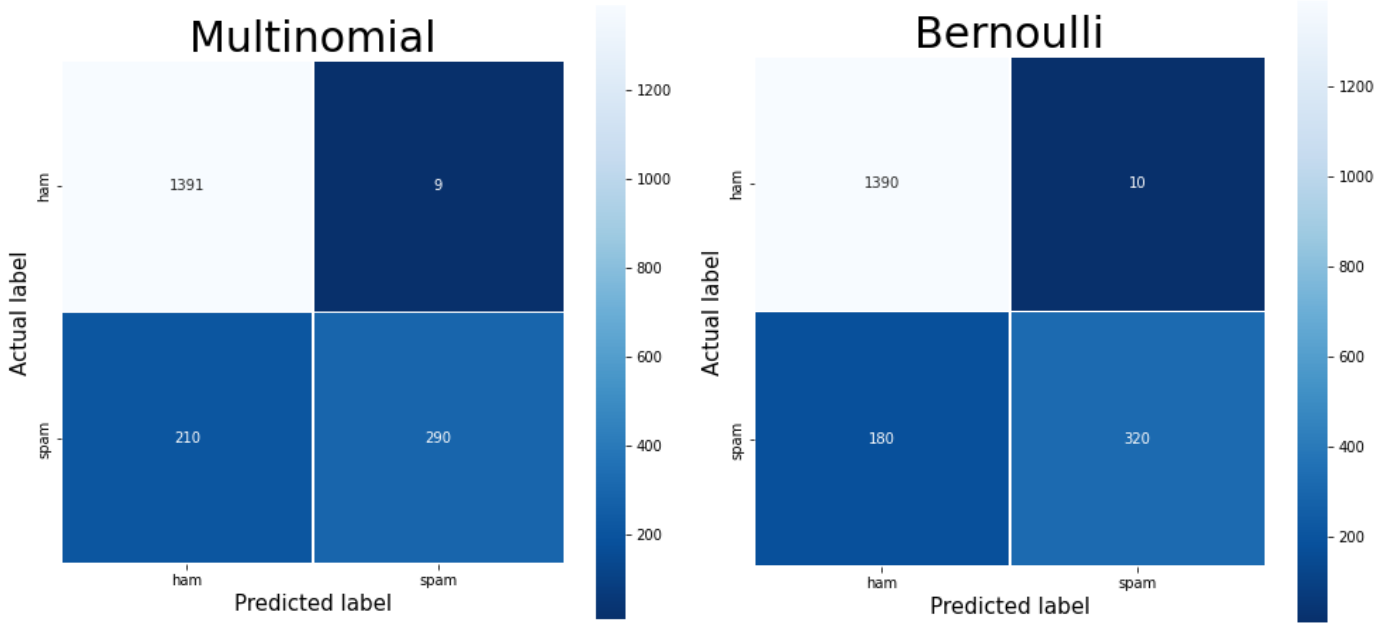
a.
Uses four datasets (hamtrain, spamtrain, hamtest, and spamtest)
See delivered notebook.
b.
Using a Naïve Bayes classifier (e.g. Sklearn), classifies the test sets and reports the percentage of ham and spam test sets that were classified correctly. You can use CountVectorizer to transform the email texts into vectors. Please note that there are different types of Naïve Bayes Classifier in SKlearn (Document is available here) Test two of these classifiers: 1. Multinomial Naive Bayes and 2. Bernoulli Naive Bayes that are well suited for this problem. For the case of Bernoulli Naive Bayes you should use the parameter binarize to make the features binary. Discuss the differences between these two classifiers.

The main difference between the two classifiers is that Bernoulli only checks whether the word is included or not. This is returned in a binary value, 1 if it is included and 0 if it is not included. The multinomial classifier counts how many times the word is included in the text and returns a value between 0 and n where n is the number of words. Intuitively, the multinomial model feels better since it measures everything Bernoulli does(if the word occurs), but gives additional information as to how many times a word has occurred. This can be beneficial to distinguish how the words are used in an email. For example, one could send a message asking "Are you free to grab a coffee" and the other message is "Free spins if you sign up today. Hurry up to grab your free reward". Both of the messages include "free",

resulting in a 1 in the Bernoulli classifier, but the second message sounds more suspicious than the first message. The multinomial would have caught this since it would count that “free” occurred twice instead of just giving a true or false return value and thus “free” might become a more decisive word when categorizing (it mostly occurs in spam).

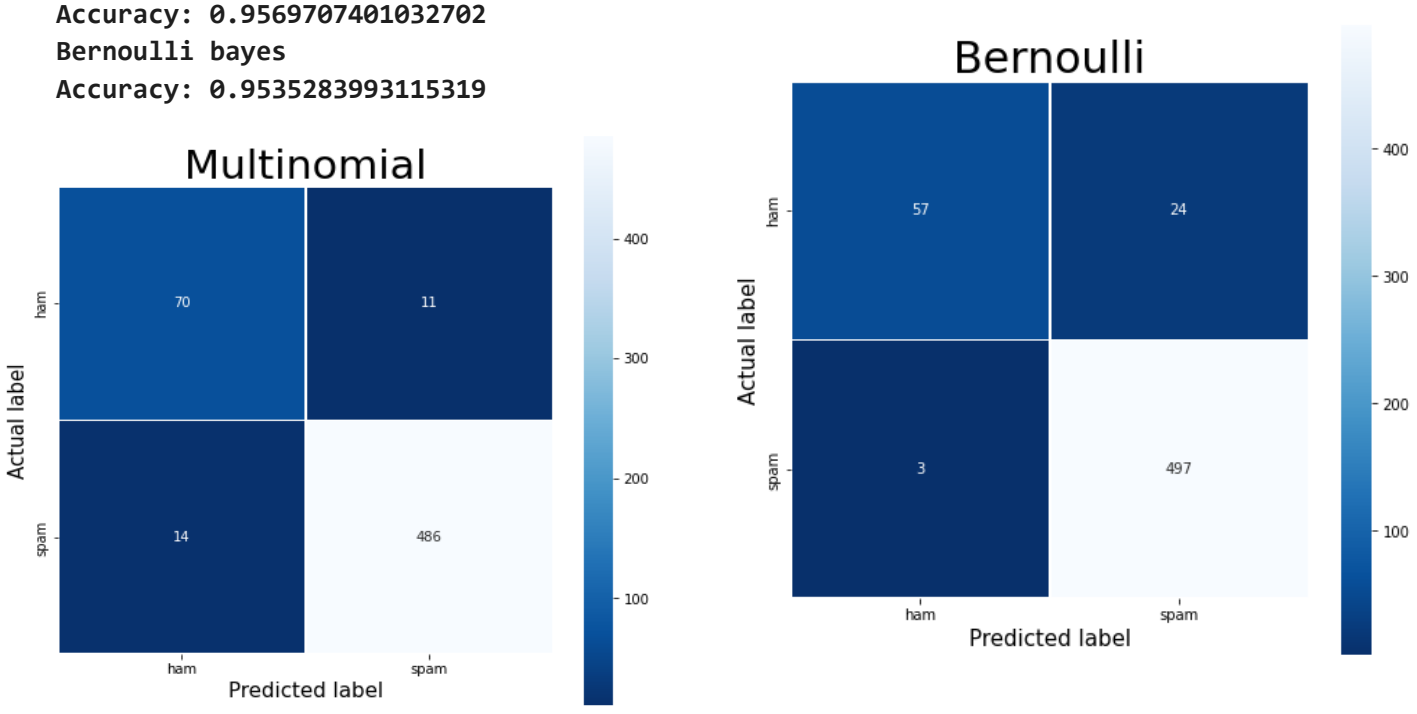
3.
Run your program on
i. Spam versus easy-ham

```
Learned vocabulary of 98392 words.
Multinomial bayes
Accuracy: 0.8847368421052632
Bernoulli bayes
Accuracy: 0.9
```



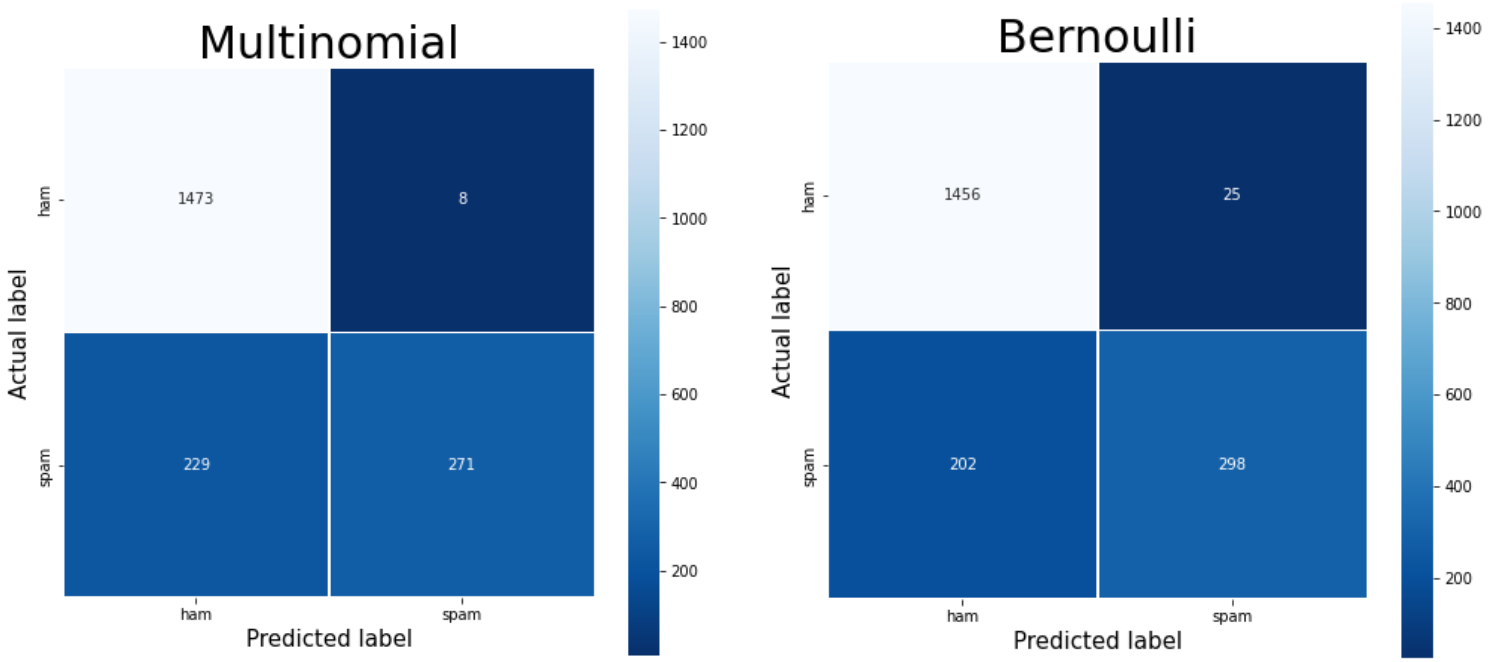
ii. Spam versus hard-ham and include the results in your report.

```
Learned vocabulary of 80107 words.
Multinomial bayes
Accuracy: 0.9569707401032702
Bernoulli bayes
Accuracy: 0.9535283993115319
```



iii. Spam versus hard-ham and easy-ham. (added this ourselves)

```
Learned vocabulary of 115230 words.
Multinomial bayes
Accuracy: 0.8803634528016153
Bernoulli bayes
Accuracy: 0.8854114083796063
```



4.
To avoid classification based on common and uninformative words it is common to filter these out.

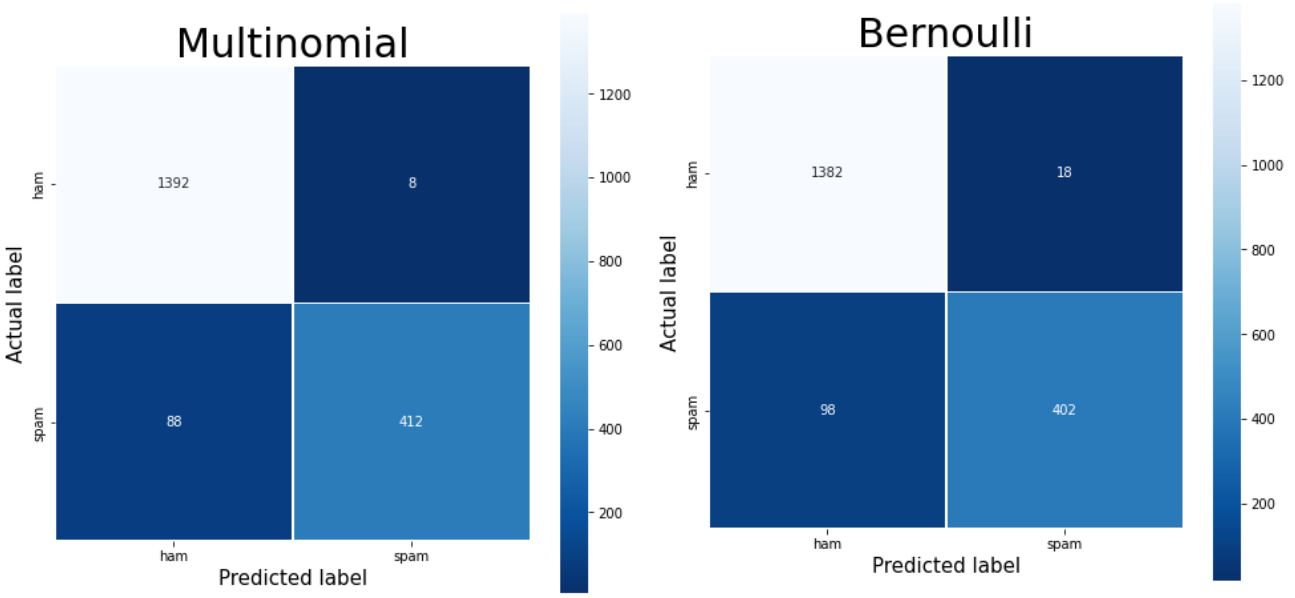
a. Argue why this may be useful. Try finding the words that are too common/uncommon in the dataset.
Common words are very hard to draw any conclusions from since they occur both in spam e-mails and as well as legitimate emails. Making analyses on these words are unnecessary to spend time on since they will not add any value to validate the email. Whilst looking at uncommon words it can be extremely hard to find any value from them since they rarely occur. It makes it hard to draw any parallels to whether the email is spam or not. If a word has been just once and it happens to be fraudulent mail, it won't give the model enough data to predict the coming mails with the same word vice versa. The main problem with uncommon words is that it can highly affect the predictions. For example, if the word "money" only occurred once in the training set, and the mail happened to be spam. The model will calculate the probability 1/1 = 100% spam when the word appears again. This can easily mess up future predictions for the classifier if these scenarios happen.

b. Use the parameters in Sklearn's CountVectorizer to filter out these words. Run the updated program on your data and record how the results differ from 3. You have two options to do this in Sklearn: either using the words found in part (a) or letting Sklearn do it for you.

We let SKlearn do the filtering for us and we used several different parameters and evaluated the results. This resulted in the following parameters:
Params: min_df = 2 (all words must occur at least twice), max_df = 0.1 (included words can't occur in more than 10 % of emails)

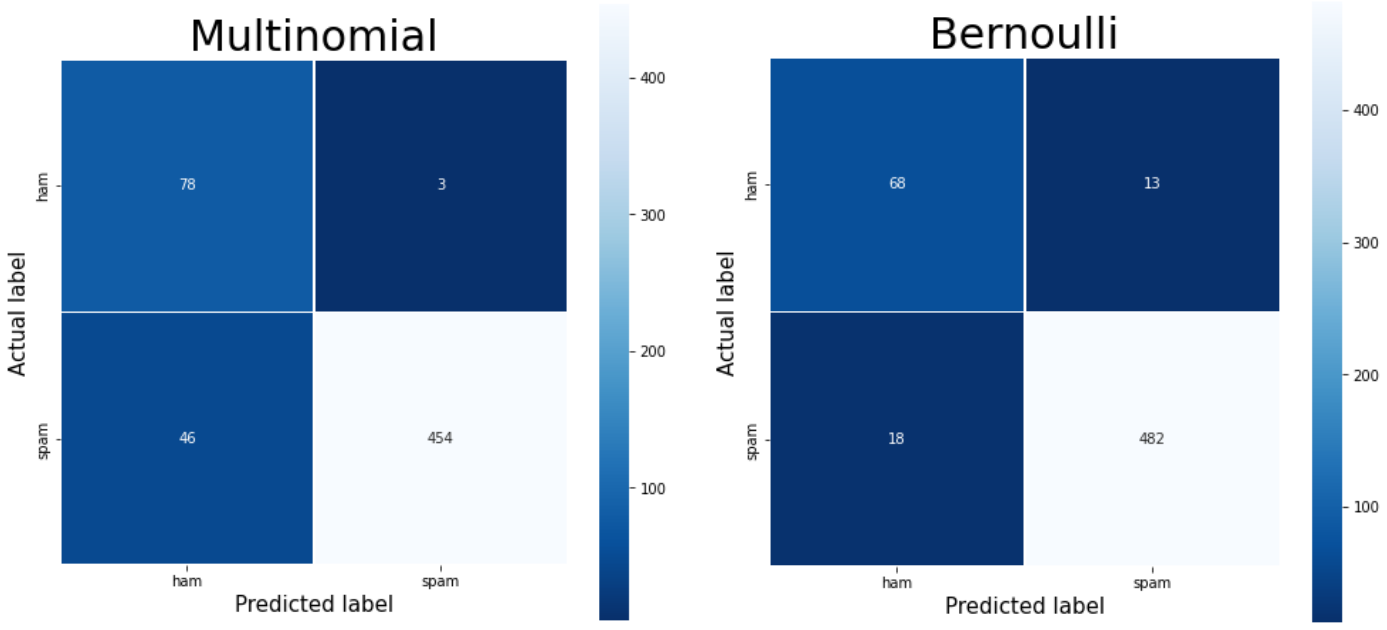
i) **Easy vs spam:**

Learned vocabulary of 30576 words.
Multinomial bayes
Accuracy: 0.9494736842105264
Bernoulli bayes
Accuracy: 0.9389473684210526



ii) **Hard vs Spam:**

Learned vocabulary of 21585 words.
Multinomial bayes
Accuracy: 0.9156626506024096
Bernoulli bayes
Accuracy: 0.9466437177280551



iii)**Both vs spam:**

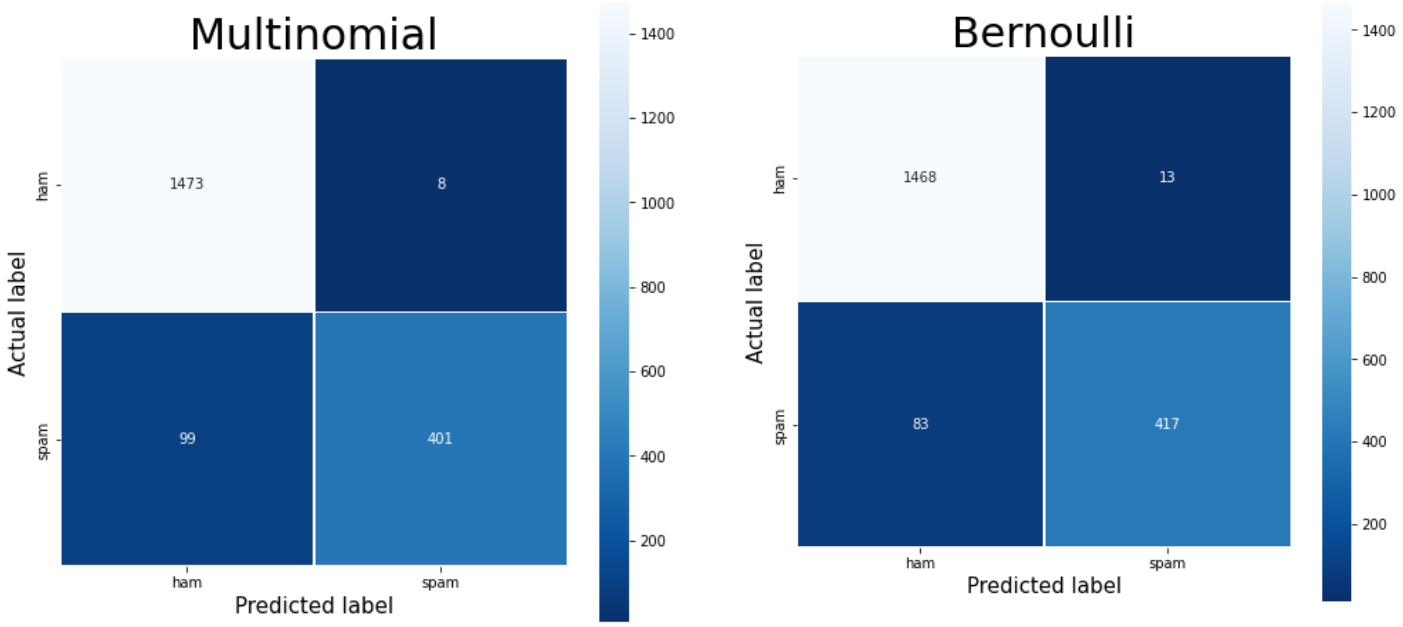
Learned vocabulary of 34227 words.

Multinomial bayes

Accuracy: 0.9459868753154972

Bernoulli bayes

Accuracy: 0.9515396264512872



No filter		
	Multinomial	**Bernoulli**
Easy	**88,47%**	**90,00%**
Hard	**95,70%**	**95,35%**
Both	**88,04%**	**88,54%**

Filter min_df = 2, max_df = 0,1		
	Multinomial	**Bernoulli**
Easy	**94,95%**	**93,89%**
Hard	**91,57%**	**94,66%**
Both	**94,60%**	**95,15%**

There was a significant increase in the easy-category, especially for the multinomial classifier. Surprisingly, there was a decrease in the predictions for the Hardham, both for the Multinomial and for the Bernoulli. Due to there being a lot fewer “hard ham” emails our models perform more varyingly on them. Without the word count filter it outperforms the other implementations, but with it out lags behind. The most significant increase was in our own metric both vs spam, where it increased over 6 percent for both of the classifiers.

5.

Filter out the headers and the 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:

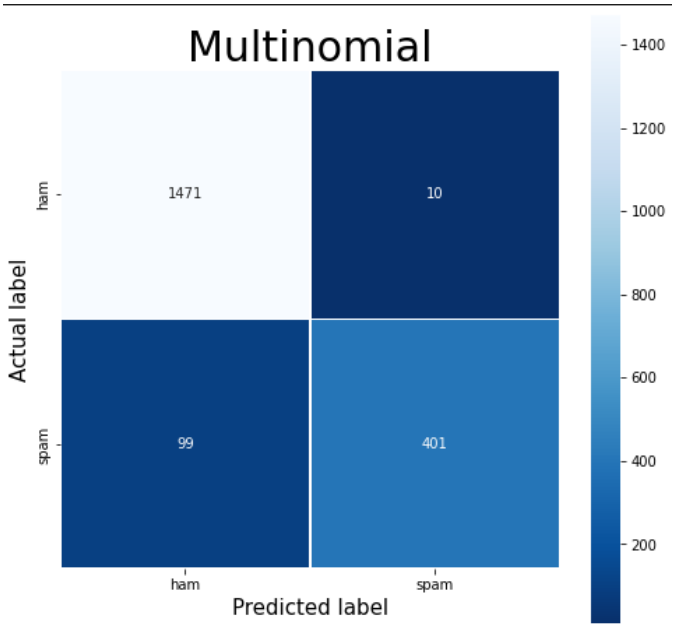
a.

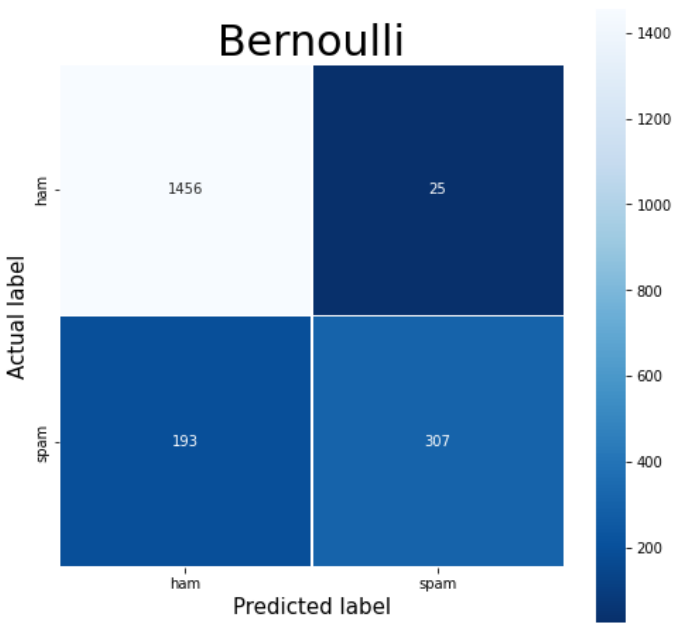
Does the result improve from 3 and 4?

Ran on all training data (both easy and hard with spam)

Without filter:

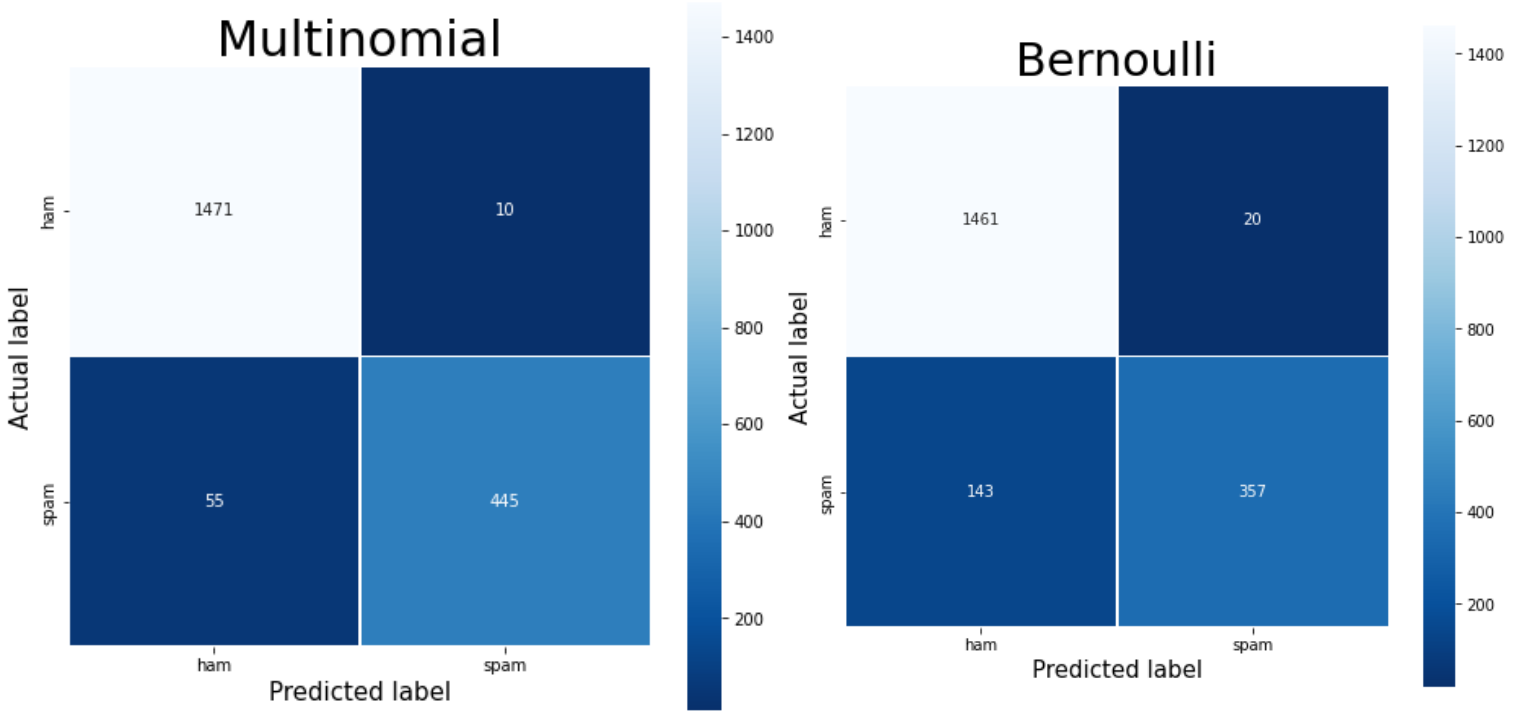
```
Learned vocabulary of 80162 words.
Multinomial bayes
Accuracy: 0.944977284199899
Bernoulli bayes
Accuracy: 0.889954568399798
```





With filter:

```
Learned vocabulary of 29122 words.
Multinomial bayes
Accuracy: 0.9671882887430591
Bernoulli bayes
Accuracy: 0.9177183240787481
```



The multinomial model performs far better, especially in the case of no word count filter.

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?

Since a model is trained on a specific dataset that dataset needs to be representative for the data the model is expected to handle. This is to give the right statistics of for example how often spam mails occur. If the entire dataset is split randomly, then it is possible (but very unlikely) that the training set can consist entirely of spam-mails. This will lead to the conclusion that all mails are spam mails. One way to fix this is to split the samples into categories (e.g. spam and ham) and then split each of them into a training and test set. Finally one can mix the two corresponding sets leaving one training set and one test set, both containing a representative amount of mails from both categories.

Another solution is to do cross-validation where the entire data set is partitioned into smaller datasets and the model trained many times on all but one, with the one dataset being used for testing and the different datasets taking turns being the one. Finally the results can be averaged and the final model trained on the entire dataset.

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 would most likely classify more ham messages as spam. Since we train the model to be either right or wrong about their predictions on certain words. The probabilities for being a spam mail will be skewed since the model will most likely be right if it classifies a mail as spam. Let's say we have 10 emails, of which 8 of them are spam. If we train on this

data and the model always says spam, leading to 80% accuracy on the training data. If we then start testing the data with mostly ham messages, our predictions will be severely wrong. Let's say “you” and other normal words occur in all 10 emails, and 8/10 were spam in the training set, the classifier will predict that this will be a scam mail even though it is normal emails. And if the test set is skewed the other way around, the results on the test set will be equally bad.