# DIT407 Introduction to data science and AI
# Assignment 3

Reynir Siik
reynir.siik@gmail.com

Franco Zambon
guszamfr@student.gu.se

2024-02-05

# 1 Problem 1: Spam & Ham

## 1.1 Introduction

*Abbreviated citation from assignment:*

The three files `20021010_easy_ham.tar.bz2`, `20021010_hard_ham.tar.bz2`, and `20021010_spam.tar.bz2` contain three sets of emails: easy *ham*, hard *ham*, and *spam*[1]. "Ham" refers to the benign, good emails that we want to read. "Spam", on the other hand, refers to unsolicited bulk email, also known as junk. We are going to adopt a bag of words approach to train a Naïve Bayesian Classifier to tell the two classes of email apart.

## 1.2 A. Data exploration

*Explore the three datasets. Use your judgment as a human being to describe what makes you able to tell spam apart from ham. Also explore the differences between the two classes of ham (easy and hard ham). What makes them different?*

### 1.2.1 Easy ham

The *Easy ham* dataset is at a quick glance personal email sent from specific mail address and addressed to one, or a few addresses that are enclosed in the header. It can also be a bulk mail, although

### 1.2.2 Hard ham

The *Hard ham* dataset is often bulk email that are presumably wanted. It could be from a mailing list we follow, or a company we do business with. It can also be a mail about something that the spam-assassin will mark as spam due to its content, ie. mail describing fraudulent mails, and what to look out for. The recipient address could be a fake address, ie. not my address. Many mails have html content with a lot of hyperlinks.

### 1.2.3 Spam

The *Spam* dataset is mail we haven't asked for, and don't want. It is mostly bulk email with undisclosed recipients. Many emails are offering porn or an offer "to good to be true".

#### 1.2.4 Conclusion

Overall it is not trivial to look at someone elses email, and just by looking at it decide whether it is spam or not. The truth lies in the eye of the beholder.

### 1.3 B. Data splitting

*Perform an appropriate train-test split on the datasets. We will use the training sets to train a classifier, and evaluate its performance against the test sets.*

The code for splitting the data sets into train and test set are listed in A.1. The data sets are stored in csv files after being converted to ISO-8859-1 character encoding. The Unicode character u2019 - Right Single Quotation Mark, is substituted with ASCII Single Quotation Mark. This is presumed to have a negligible impact on training result.

## 2 Problem 2: Preprocessing

### 2.1 Introduction

*Abbreviated citation from assignment:*

A *bag of words* maps text documents into vectors of word counts. That is, each word is assigned a token number, and we count the number of occurrences of each unique word in each document. The vector then consists of these counts.

### 2.2 Our preprocessing

What we did in this assignment was splitting each of the mail content into a vector of words by using the `CountVectorizer`. After doing that we applied it to the training split by using the function `fit_transform()` and to the test split with the `transform()` function. Finally we transformed the vectors into a matrix using the `toarray()` function.

## 3 Problem 3: Easy Ham

### 3.1 Introduction

*Abbreviated citation from assignment:*

We are going to try out two different Naïve Bayesian Classifiers:

- The Multinomial Naïve Bayesian Classifier, and [1]
- The Bernoulli Naive Bayesian Classifier3. [2]

For this problem, use the set easy ham as your ham. Use the training sets of ham and spam to train an instance of both classifiers. Then, evaluate the classifier against the test set.

[1] `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html`

[2] `https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html`

### 3.2 What we did

Fist of all we converted the `y` variables into integer (previously they were object) and after that we trained the Multinomial Bayes Naïve Classifier by using the `fit()` function. After that we used it to predict on the test. The result was pretty good, since we obtained these statistics:

Accuracy: 0.9748908296943232
Precision: 0.9751227237126167
Recall: 0.9748908296943232

Confusion Matrix:

$$\begin{bmatrix} 127 & 21 \\ 2 & 766 \end{bmatrix} \begin{matrix} 148 \\ 768 \end{matrix}$$

129 787

After that we use the Bernoulli Naive Bayesian Classifier. From that we obtained:

Accuracy: 0.9104803493449781
Precision: 0.9135972782624029
Recall: 0.9104803493449781

Confusion Matrix:

$$\begin{bmatrix} 70 & 78 \\ 4 & 764 \end{bmatrix} \begin{matrix} 148 \\ 768 \end{matrix}$$

74 842

## 4 Problem 4: Hard Ham

### 4.1 Introduction

*Abbreviated citation from assignment:*

Repeat the experiment of the previous problem, but use hard ham instead. Report the same values. Discuss the differences in results.

### 4.2 What we did

The procedure for this exercise was the same of the previous one. In this case we obtained for the Multinomial Bayes Naïve Classifier:

Accuracy: 0.9646017699115044
Precision: 0.9648439683278993
Recall: 0.9646017699115044

Confusion Matrix:

$$\begin{bmatrix} 144 & 2 \\ 6 & 74 \end{bmatrix} \begin{matrix} 146 \\ 80 \end{matrix}$$

150 76

For the Bernoulli Naive Bayesian Classifier:

Accuracy: 0.8938053097345132
Precision: 0.9054539575976868
Recall: 0.8938053097345132

Confusion Matrix:

$$\begin{bmatrix} 145 & 1 \\ 23 & 57 \end{bmatrix} \begin{matrix} 146 \\ 80 \end{matrix}$$

168 58

# 5 Discussion

What we can conclude from this analysis is that the Multinomial Bayes Naive Classifier work better in both cases.
In general the prediction are pretty good since they exceed 95% precision.

# References

[1] J. Mason, *Spam assassin public mail corpus.* Made available on Canvas, From assignment handout on Canvas, 2004. [Online]. Available: `https://spamassassin.apache.org/old/publiccorpus/`.

# A Source code

This is the complete listing of the source code.

## A.1 Problem 1

Code for splitting the datasets into training and test sets.

```
# Reynir Siik, Franco Zambon
# DIT407 lp3 2024-02-10
# Assignment 3
# Problem 1
# Spam & Ham
# Splitting the datasets into training set and test set
#############################################################################
import pandas as pd
import tarfile
import re
from sklearn.model_selection import train_test_split
## Read data from tar file ##################################################
def read_data(fn, ds):
    dataset = list()
    with tarfile.open(fn) as tf:
        for i in tf:
            if i.isfile():
                f = tf.extractfile(i)
                if f:
                    with f:
                        try:
                            s = f.read().decode('iso-8859-1')
                        except:
                            s = f.read().decode('utf-8')
                            s = re.sub(u"[\u2018|\u2019]", "'", s)
                        dataset.append((ds, s))
    return dataset
#############################################################################
trainingSet = list()
testingSet = list()
set_list = ["easy_ham", "hard_ham", "spam"]
for set in set_list:
    fname = "Uppgift03\\\\00_Uppg_Data\\\\20021010_"+set+".tar.bz2"
    print(fname)
    df = read_data(fname, set)
    tr, te = train_test_split(df, train_size=0.7, random_state=42) # 23145)
    trainingSet.extend(tr)
    testingSet.extend(te)
# Create the pandas DataFrame
DFtrainingSet = pd.DataFrame(trainingSet, columns =['Status', 'Message'])
DFtestingSet  = pd.DataFrame(testingSet, columns =['Status', 'Message'])
# Save to csv file
DFtrainingSet.to_csv(r"Uppgift03\\easy_ham\\20021010_training.csv")
DFtestingSet.to_csv(r"Uppgift03\\easy_ham\\20021010_test.csv")
```

## A.2 Problem 2 to 4

Code for making a bag of words and calculate statistics.

```
# Reynir Siik, Franco Zambon
# DIT407 lp3 2024-02-10
# Assignment 3
# Problem 2, 3, 4
# Spam & Ham
# Preprocessing and Classifying
#############################################################################
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.metrics import confusion_matrix
## ###########################################################################
filelocation = "Uppgift03\\\\easy_ham\\\\"
## ###########################################################################

df1=pd.read_csv(filelocation + '20021010_training.csv')
df2=pd.read_csv(filelocation + '20021010_test.csv')
df = pd.merge(df1, df2, how='outer')
print(df['Status'].value_counts())

# Removing the hard_ham from the dataset and converting the Status to a
# bolean value
df_eh = df[df['Status'] != 'hard_ham']
df_eh.loc[df_eh["Status"]=='spam', "Status"]=0
df_eh.loc[df_eh["Status"]=='easy_ham', "Status"]=1
df_eh = df_eh.drop(columns=['Unnamed: 0']).copy()
print(df_eh.head())

## PROBLEM 3 ################################################################
## Splitting the dataset, vectorizing the Message and counting each word
## and finally converting it to an array
print('********** Problem 3 **********')
X = df_eh['Message']
y = df_eh['Status']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=42)
vectorizer = CountVectorizer(lowercase=True, stop_words='english')
X_traincv = vectorizer.fit_transform(X_train)
X_testcv = vectorizer.transform(X_test)
X_traincv.toarray()
X_testcv.toarray()

# Applying the Multinomial Bayes Naive Classifier
mnb = MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
y_train=y_train.astype('int')
y_test=y_test.astype('int')
mnb.fit(X_traincv,y_train)
pred_Mult = mnb.predict(X_testcv)
# Accuracy
accuracy = accuracy_score(y_test, pred_Mult)
# Precision
```

```
precision = precision_score(y_test, pred_Mult, average='weighted')
# Recall
recall = recall_score(y_test, pred_Mult, average='weighted')
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
# Confusion matrix
conf_matrix = confusion_matrix(y_test, pred_Mult)
print("Confusion Matrix:")
print(conf_matrix)
# Applying the Bernulli Naive Bayesian Classifier
clf = BernoulliNB()
clf.fit(X_traincv,y_train)
pred_Bern = clf.predict(X_testcv)
# Accuracy
accuracy = accuracy_score(y_test, pred_Bern)
# Precision
precision = precision_score(y_test, pred_Bern, average='weighted')
# Recall
recall = recall_score(y_test, pred_Bern, average='weighted')
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
# Confusion matrix
conf_matrix = confusion_matrix(y_test, pred_Bern)
print("Confusion Matrix:")
print(conf_matrix)

## PROBLEM 4 ##################################################################
## Removing the easy_ham from the dataset and converting the Status to
## a bolean value
print('********** Problem 4 **********')
df_hh = df[df['Status'] != 'easy_ham']
df_hh.loc[df_hh["Status"]=='spam', "Status"]=0
df_hh.loc[df_hh["Status"]=='hard_ham', "Status"]=1
df_hh = df_hh.drop(columns=['Unnamed: 0']).copy()
df_hh.head()
# Splitting the dataset, vectorizing the Message and counting each word
# and finally converting it to an array
X = df_hh['Message']
y = df_hh['Status']
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,
                                                    random_state=42)
vectorizer = CountVectorizer(lowercase=True, stop_words='english')
X_traincv = vectorizer.fit_transform(X_train)
X_testcv = vectorizer.transform(X_test)
X_traincv.toarray()
X_testcv.toarray()
# Applying the Multinomial Bayes Naive Classifier
mnb = MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
y_train=y_train.astype('int')
y_test=y_test.astype('int')
mnb.fit(X_traincv,y_train)
pred_Mult = mnb.predict(X_testcv)
# Accuracy
accuracy = accuracy_score(y_test, pred_Mult)
# Precision
precision = precision_score(y_test, pred_Mult, average='weighted')
```

```
# Recall
recall = recall_score(y_test, pred_Mult, average='weighted')
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
# Confusion matrix
conf_matrix = confusion_matrix(y_test, pred_Mult)
print("Confusion Matrix:")
print(conf_matrix)
# Applying the Bernulli Naive Bayesian Classifier
clf = BernoulliNB()
clf.fit(X_traincv,y_train)
pred_Bern = clf.predict(X_testcv)
# Accuracy
accuracy = accuracy_score(y_test, pred_Bern)
# Precision
precision = precision_score(y_test, pred_Bern, average='weighted')
# Recall
recall = recall_score(y_test, pred_Bern, average='weighted')
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
# Confusion matrix
conf_matrix = confusion_matrix(y_test, pred_Bern)
print("Confusion Matrix:")
print(conf_matrix)
```