DAT565/DIT407 Assignment 3

Saif Sayed
gussayedfa@student.gu.se

Gona Ibrahim Abdulrahman
gusibrgo@student.gu.se

2024-04-18

This is a report for assignment 3 for the course *Introduction to Data Science & AI* from Chalmers and Gothenburg University.

Problem 1: Spam and Ham

A Data exploration

When analyzing email contents, it's crucial to differentiate between spam (unsolicited or promotional emails) and ham (legitimate personal or work-related correspondence). Spam emails often contain exaggerated language, excessive capitalization, and phrases like "Congratulations!" or "Don't miss your chance.". They originate from unfamiliar email addresses and may have grammatical errors. In contrast, ham emails are straightforward, relevant, and come from known contacts. There are two types of ham: easy ham, which includes simple work-related or personal messages, and hard ham, which involves more specialized content. Remember that spam expects no specific action, while ham often requires a reply or some form of follow-up.

Differences Between Easy Ham and Hard Ham:

Easy Ham: These straightforward emails are likely related to work, personal matters, or subscriptions. They have minimal noise or clutter and may include relevant attachments. Usually written in first person.

Hard Ham: Hard ham emails seem to be closer to spam as it contains promotions and newsletter from websites and companies the user has subscribed to. Hard ham emails tend to be presented in a more complex format and may require more careful evaluation to distinguish them from actual spam.

B Data splitting

We extracted email data from three archives: *easy_ham*, *hard_ham*, and *spam*. And then we split the data by performing a train-test split on each dataset (easy/hard ham and spam) and used the training sets to train the model and evaluated its performance against test sets.

To model Training and Evaluation: We selected an appropriate classification model (e.g., Naive Bayes) and trained the model using training data ($X\_train, X1\_train$) and corresponding labels ($Y\_train, Y1\_train$). Then evaluated model performance on test data ($X\_test, X1\_test$) and corresponding labels ($Y\_test, Y1\_test$) using metrics like accuracy and precision.

By splitting the data and training a classifier, we aim to create an effective email classification model. Evaluation results will guide us in understanding how well the model performs across different email types.

Our source code can be found in Appendix A of this document.

Problem 2: Preprocessing

Our goal is to transform a collection of emails into a matrix format. Each row in this matrix corresponds to an email, and each column represents the count of unique words within that email. This process allows us to map text documents (emails) into numerical vectors based on word frequencies .

We combined emails from two categories: Easy Ham and Spam. This simplification streamlines subsequent processes (problem 3).

We created an instance of CountVectorizer, as a tool for converting text data into numerical representations and then using the fit_transform method, we learned the vocabulary (unique words) from the entire email dataset and transformed the emails into a matrix representation of word counts.

By using the CountVectorizer, we successfully converted the emails into a matrix of word counts and this matrix will serve as input for training and evaluating our email classification model.

Problem 3: Easy Ham

We combined emails from two categories: Easy Ham and Spam and then assigned labels to each dataset.

Later we instantiated two classifiers: Multinomial Naive Bayes (MNB) and Bernoulli Naive Bayes (BNB).

And then trained both classifiers using the training data and evaluated their performance against the test set mentioned in **Problem 1B**. Now both classifiers have been successfully trained and evaluated and the confusion matrices provide valuable insights into their performance.

Table 1 shows the accuracy, precision and recall score of the Multinomial and Bernoulli Naive Bayes Classifier.

Figure 1 shows the confusion matrix of Easy ham and Spam emails using the Multinomial Naive Bayes Classifier.

Figure 2 shows the confusion matrix of Easy ham and Spam emails using the Bernoulli Naive Bayes Classifier.

Table 1: Classification Report (Easy & Spam emails)

Stats	Multinomial	Bernoulli
Easy Ham emails	2551	2551
Spam emails	501	501
Test size	763	763
Accuracy	0.9777195281782438	0.9043250327653998
Precision (ham)	0.9767801857585139	0.9011461318051576
Recall Score (ham)	0.9968404423380727	0.9936808846761453

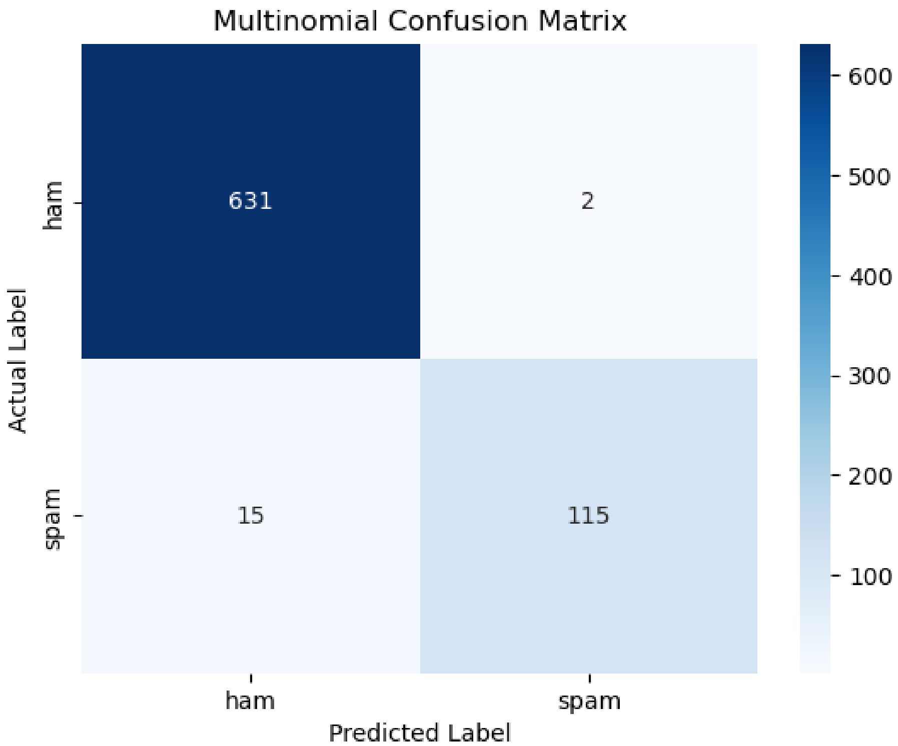

Figure 1: Multinomial Confusion Matrix of Easy Ham and Spam emails

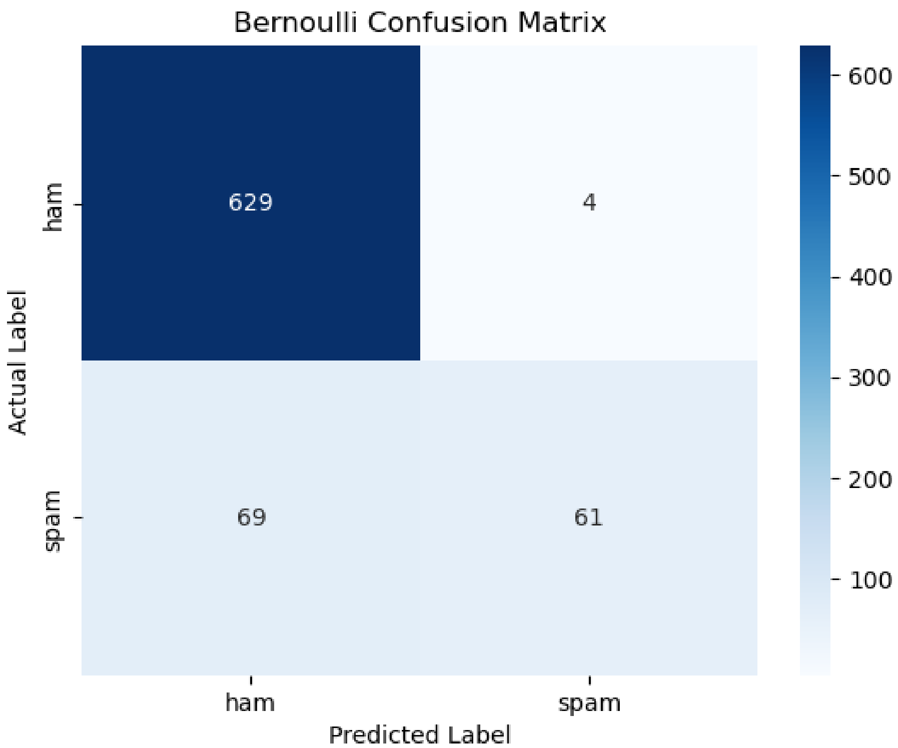

Figure 2: Bernoulli Confusion Matrix of Easy Ham and Spam emails

Problem 4: Hard Ham

We applied the same solution as in problem 3, but this time we changed it for 'Hard ham' instead of 'Easy ham'.

Table 2 shows the accuracy, precision and recall score of the Multinomial and Bernoulli Naive Bayes Classifier.

Figure 3 shows the confusion matrix of Hard ham and Spam emails using the Multinomial Naive Bayes Classifier.

Figure 4 shows the confusion matrix of Hard ham and Spam emails using the Bernoulli Naive Bayes Classifier.

Table 2: Classification Report (Hard & Spam emails)

Stats	Multinomial	Bernoulli
Hard Ham emails	250	250
Spam emails	501	501
Test size	188	188
Accuracy	0.925531914893617	0.8723404255319149
Precision (ham)	0.9827586206896551	0.96
Recall Score (ham)	0.8142857142857143	0.6857142857142857

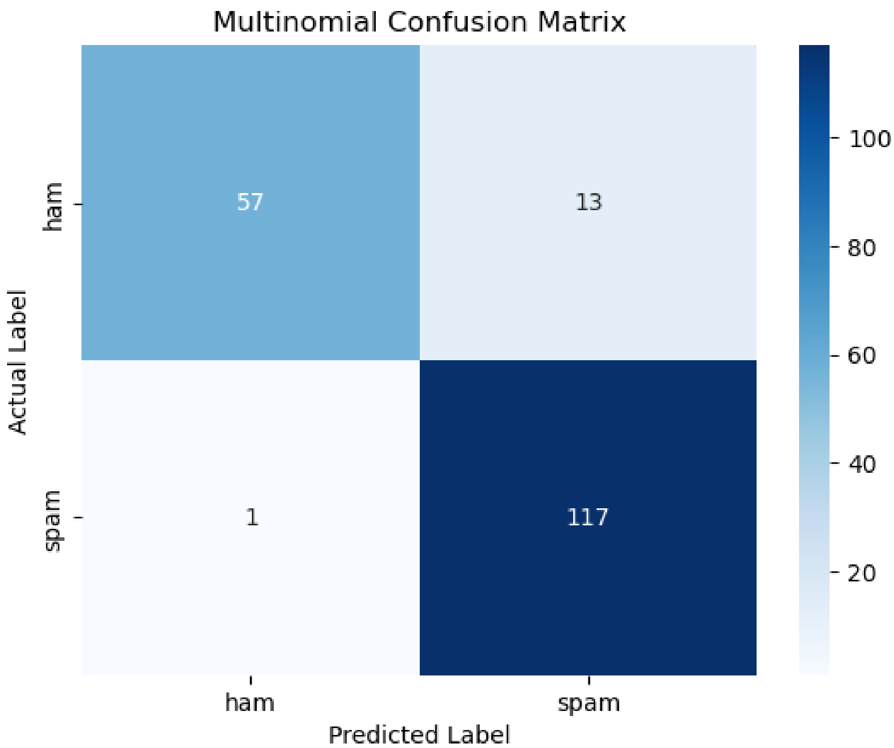

Figure 3: Multinomial Confusion Matrix of Hard Ham and Spam emails

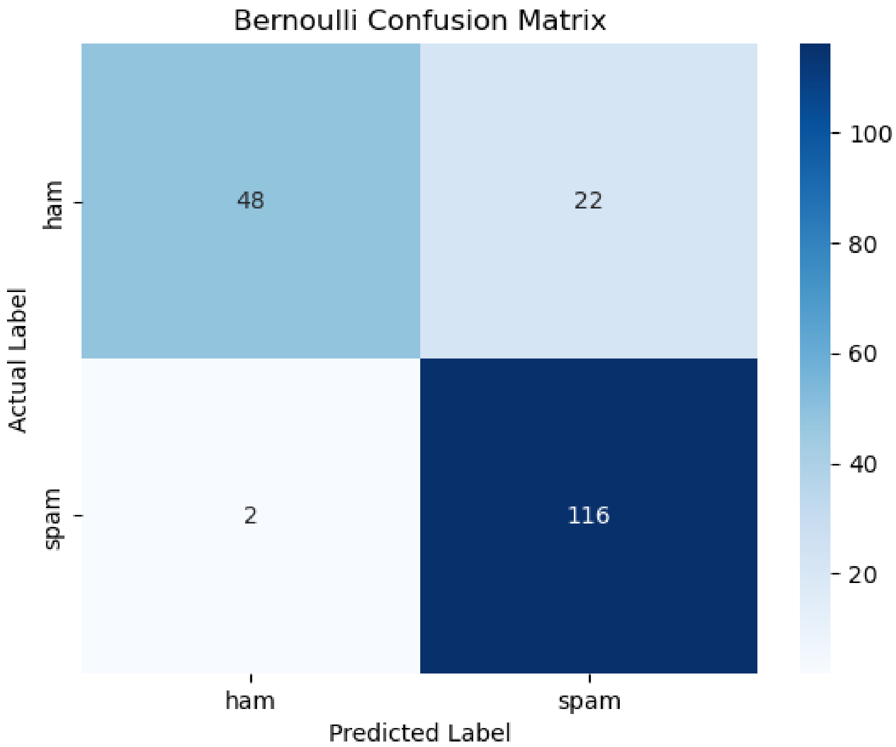

Figure 4: Bernoulli Confusion Matrix of Hard Ham and Spam emails

Discussion

Both the multinomial and Bernoulli classifiers have higher accuracy and precision in classifying 'Easy ham' and 'Spam' emails compared to classifying 'Hard ham' and 'Spam' emails. The recall score for 'Hard ham' and 'Spam' emails indicates that both classifiers have a lower proportion of actual ham emails predicted as 'Hard ham' emails, in contrast to the recall score for 'Easy ham' and 'Spam' emails. Consequently, both classifiers yield more false positive results for 'Hard ham' emails. Although the results might be biased since the classifiers were trained with a larger dataset of 'Easy ham' emails compared to the dataset of 'Hard ham' emails. Overall, the Multinomial Naive Bayes classifier performs well in terms of accuracy, precision, and recall for both datasets.

A Python code

This is the code we used to classify between Easy ham emails, Hard ham emails and Spam emails .

```
import matplotlib.pyplot as plt
import tarfile
import os
```

```python
import chardet # to identify encoding format
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
import seaborn as sns

# B. Data splitting and Preprocessing

current_directory = os.getcwd()
extract_dir = current_directory

tar_easy_ham = os.path.join(current_directory, "20021010_easy_ham.tar.bz2")
tar_hard_ham = os.path.join(current_directory, "20021010_hard_ham.tar.bz2")
tar_spam = os.path.join(current_directory, "20021010_spam.tar.bz2")

with tarfile.open(tar_easy_ham, "r") as tar:
    tar.extractall(path=extract_dir)
with tarfile.open(tar_hard_ham, "r") as tar:
    tar.extractall(path=extract_dir)
with tarfile.open(tar_spam, "r") as tar:
    tar.extractall(path=extract_dir)

easy_ham_dir = os.path.join(extract_dir, "easy_ham")
hard_ham_dir = os.path.join(extract_dir, "hard_ham")
spam_dir = os.path.join(extract_dir, "spam")

def read_emails_from_directory(directory):
    emails = []
    for filename in os.listdir(directory):
        file_path = os.path.join(directory, filename)
        with open(file_path, "rb") as file: # open in binary mode to detect encoding
            raw_content = file.read()
            detected_encoding = chardet.detect(raw_content)['encoding']
            try:
                content = raw_content.decode(detected_encoding) # decode according to detected encoding
            except UnicodeDecodeError:
                content = raw_content.decode(detected_encoding, errors='replace') # else identify the error and
```

```
                      decode using another encoding.
            emails.append(content)
    return emails

easy_ham_emails = read_emails_from_directory(easy_ham_dir)
hard_ham_emails = read_emails_from_directory(hard_ham_dir)
spam_emails = read_emails_from_directory(spam_dir)

easy_and_spam_emails = easy_ham_emails + spam_emails
hard_and_spam_emails = hard_ham_emails + spam_emails

vectorizer = CountVectorizer()
easy_spam_email_vectors_train = vectorizer.fit_transform(easy_and_spam_emails)
hard_spam_email_vectors_train = vectorizer.fit_transform(hard_and_spam_emails)

label_easy_spam = ['ham'] * len(easy_ham_emails) + ['spam'] * len(spam_emails)
label_hard_spam = ['ham'] * len(hard_ham_emails) + ['spam'] * len(spam_emails)

#split data for easy ham and spam combination ( 25% test emails and 75% emails for training)
X_train, X_test, Y_train, Y_test = train_test_split(easy_spam_email_vectors_train, label_easy_spam, test_size=0.25, random_state=42)
#split data for hard ham and spam combination ( 25% test emails and 75% emails for training)
X1_train, X1_test, Y1_train, Y1_test = train_test_split(hard_spam_email_vectors_train, label_hard_spam, test_size=0.25, random_state=42)

# Problem 3: Easy Ham

multinomial_classifier = MultinomialNB()
bernoulli_classifier = BernoulliNB()

multinomial_classifier.fit(X_train, Y_train)

bernoulli_classifier.fit(X_train, Y_train)

multinomial_predictions = multinomial_classifier.predict(X_test)
bernoulli_predictions = bernoulli_classifier.predict(X_test)

```

```
multinomial_accuracy = accuracy_score(Y_test, multinomial_predictions)
bernoulli_accuracy = accuracy_score(Y_test, bernoulli_predictions)

multinomial_precision = precision_score(Y_test, multinomial_predictions, pos_label='ham')
bernoulli_precision = precision_score(Y_test, bernoulli_predictions, pos_label='ham')

multinomial_recall = recall_score(Y_test, multinomial_predictions, pos_label='ham')
bernoulli_recall = recall_score(Y_test, bernoulli_predictions, pos_label='ham')

multinomial_confusion = confusion_matrix(Y_test, multinomial_predictions)
bernoulli_confusion = confusion_matrix(Y_test, bernoulli_predictions)

print("Size_of_easy-ham_emails:", len(easy_ham_emails))
print("Size_of_hard_ham_emails:", len(hard_ham_emails))
print("Size_of_spam_emails:", len(spam_emails))
print()
# Print the results
print("Easy_Ham_and_Spam")
print("-------------------")
print("Size_of_test_set:", len(Y_test))
print("Multinomial Naive Bayes Classifier:")
print("Accuracy:", multinomial_accuracy)
print("Precision:", multinomial_precision)
print("Recall:", multinomial_recall)
print("Confusion_Matrix:")
# Create a heatmap of the confusion matrix
sns.heatmap(multinomial_confusion, annot=True, cmap='Blues', fmt='d', xticklabels=['ham', 'spam'],
                yticklabels=['ham', 'spam'])
# Set the title and axis labels
plt.title('Multinomial_Confusion_Matrix')
plt.xlabel('Predicted_Label')
plt.ylabel('Actual-Label')
plt.show()
print()
print("Size_of_test_set:", len(Y_test))
print("Bernoulli_Naive_Bayes_Classifier:")
print("Accuracy:", bernoulli_accuracy)
print("Precision:", bernoulli_precision)
print("Recall:", bernoulli_recall)
```

```python
print("Confusion Matrix:")

sns.heatmap(bernoulli_confusion, annot=True, cmap='Blues', fmt='d', xticklabels=['ham', 'spam'],
                yticklabels=['ham', 'spam'])
plt.title('Bernoulli Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('Actual Label')
plt.show()
print()

# Problem 4: Hard Ham

multinomial_classifier.fit(X1_train, Y1_train)
bernoulli_classifier.fit(X1_train, Y1_train)

multinomial_predictions = multinomial_classifier.predict(X1_test)
bernoulli_predictions = bernoulli_classifier.predict(X1_test)

multinomial_accuracy = accuracy_score(Y1_test, multinomial_predictions)
bernoulli_accuracy = accuracy_score(Y1_test, bernoulli_predictions)

multinomial_precision = precision_score(Y1_test, multinomial_predictions, pos_label='ham')
bernoulli_precision = precision_score(Y1_test, bernoulli_predictions, pos_label='ham')

multinomial_recall = recall_score(Y1_test, multinomial_predictions, pos_label='ham')
bernoulli_recall = recall_score(Y1_test, bernoulli_predictions, pos_label='ham')

multinomial_confusion = confusion_matrix(Y1_test, multinomial_predictions)
bernoulli_confusion = confusion_matrix(Y1_test, bernoulli_predictions)
# Print the results
print("Hard Ham and Spam")
print("-------------------")
print("Size of test set:", len(Y1_test))
print("Multinomial Naive Bayes Classifier:")
print("Accuracy:", multinomial_accuracy)
print("Precision:", multinomial_precision)
print("Recall:", multinomial_recall)
print("Confusion Matrix:")
# Create a heatmap of the confusion matrix
```

```
sns.heatmap(multinomial_confusion, annot=True, cmap='Blues', fmt='d', xticklabels=['ham', 'spam'],
                yticklabels=['ham', 'spam'])
plt.title('Multinomial_Confusion_Matrix')
plt.xlabel('Predicted_Label')
plt.ylabel('Actual_Label')
plt.show()
print()
print("Size_of_test_set:", len(Y1_test))
print("\nBernoulli_Naive_Bayes_Classifier:")
print("Accuracy:", bernoulli_accuracy)
print("Precision:", bernoulli_precision)
print("Recall:", bernoulli_recall)
print("Confusion_Matrix:")
sns.heatmap(bernoulli_confusion, annot=True, cmap='Blues', fmt='d', xticklabels=['ham', 'spam'],
                yticklabels=['ham', 'spam'])
plt.title('Bernoulli_Confusion_Matrix')
plt.xlabel('Predicted_Label')
plt.ylabel('Actual_Label')

plt.show()
```