DAT565/DIT407 Assignment 3

Ola Bratt
ola.bratt@gmail.com

Patrick Attimont
patrickattimont@gmail.com

2024-02-xx

This paper is addressing the assignment 3 study queries within the *Introduction to Data Science & AI* course, DIT407 at the University of Gothenburg and DAT565 at Chalmers. The main source of information for this project is derived from the lectures and Skiena [1]. Assignment 3 is about text classification and the use of correct data splitting and encoding handling.

Problem 1: Spam and Ham

A. Data exploration

B. Data splitting

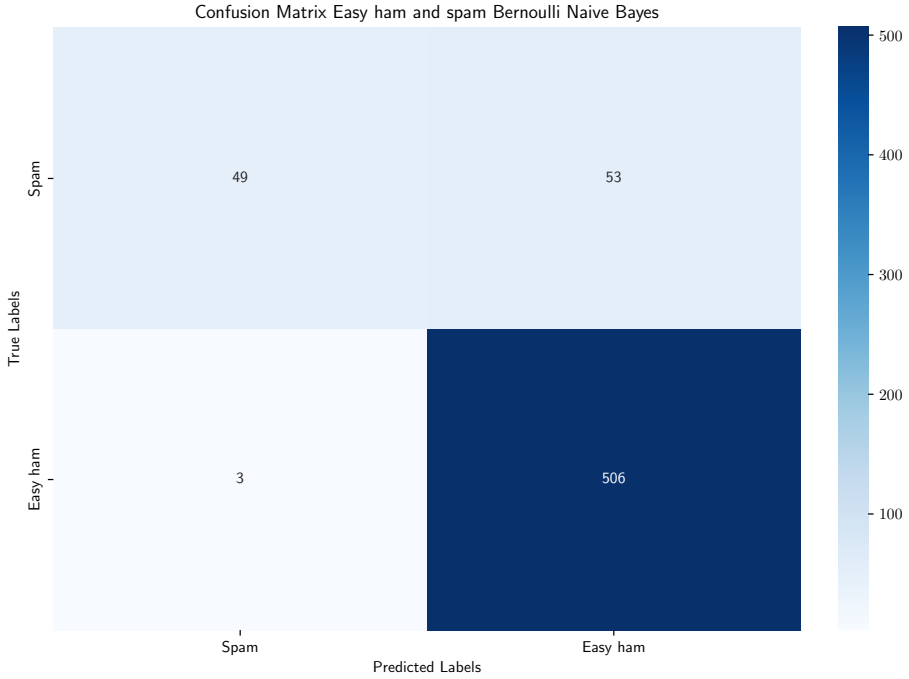
```
X_train, X_test, y_train, y_test =
train_test_split(email_matrix, labels, test_size=0.2)
```

Problem 2: Preprocessing

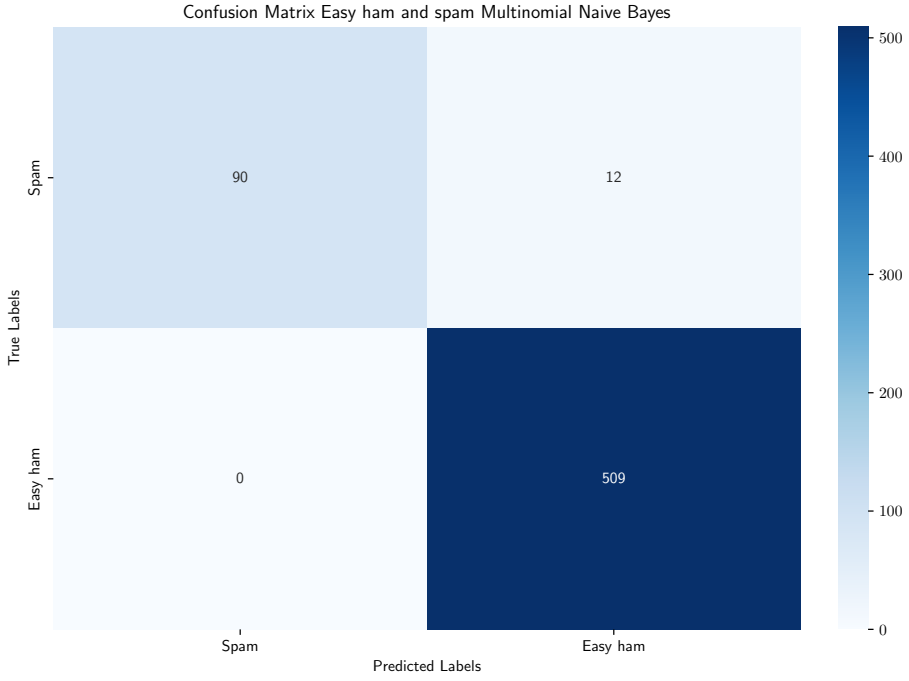
The "bag of words" model is a basic and intuitive way to analyze and compare documents based on their textual content. However, it does not consider the context or the order of words, which can limit its effectiveness in capturing the semantics and meaning of the text.

Problem 3: Easy Ham

```
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
```



(a) Easy ham vs spam, Bernoulli Naive Bayes



(b) Easy ham vs spam, Multinomial Naive Bayes

Figure 1: Confusion matrixes of easy ham and spam

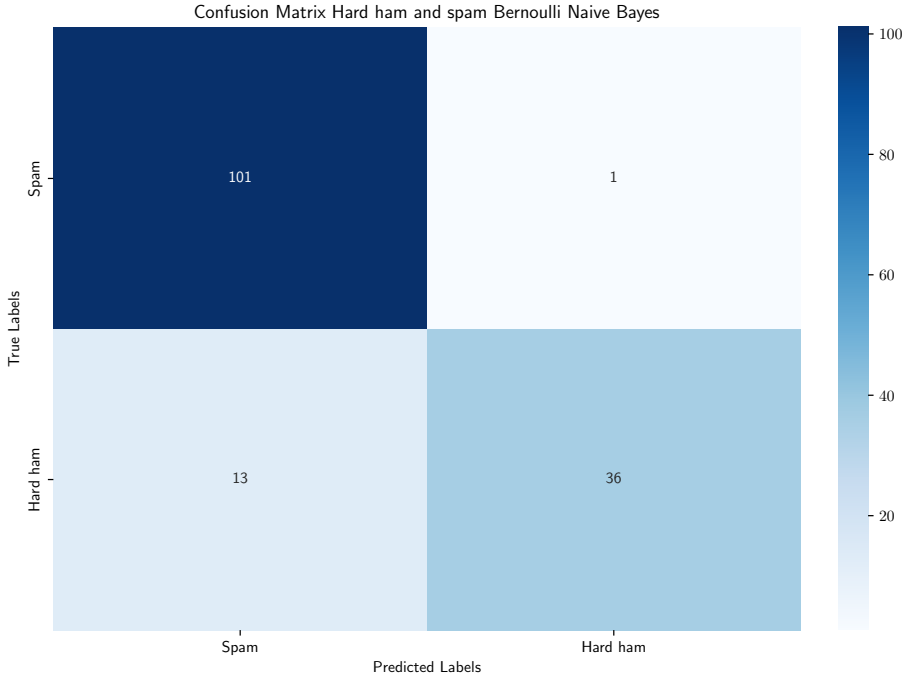
Model	accuracy	precision	recall	F1 score
Multinomial Naive Bayes	0.985	0.984	0.998	0.991
Bernoulli Naive Bayes	0.923	0.918	0.996	0.956

Table 1: Precision and accuracy for Easy Ham and Spam

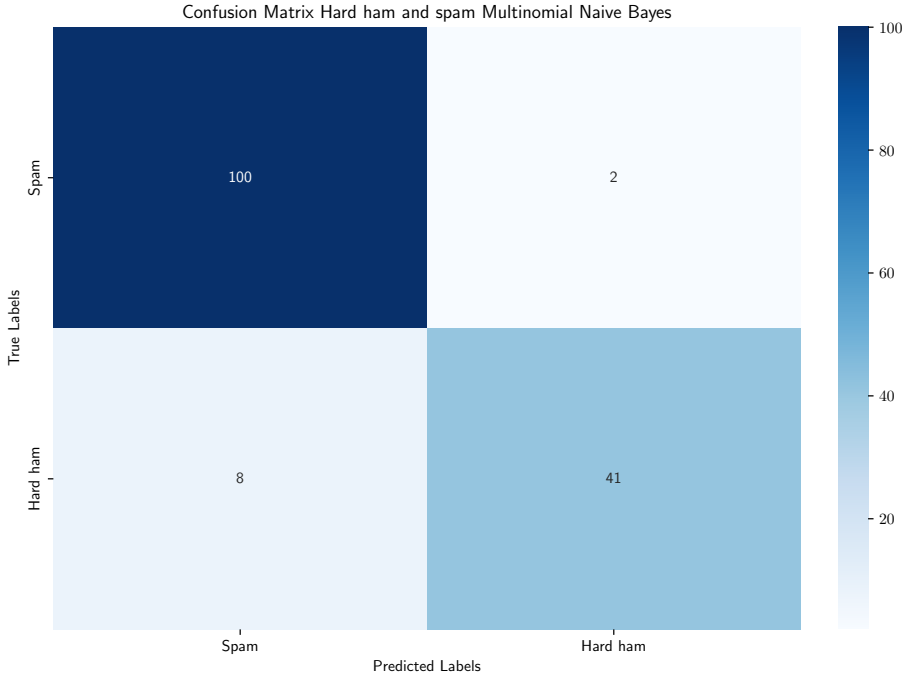
Model	accuracy	precision	recall	F1 score
Multinomial Naive Bayes	0.947	0.956	0.878	0.915
Bernoulli Naive Bayes	0.934	0.976	0.816	0.889

Table 2: Precision and accuracy for Hard Ham and Spam

Problem 3: Hard Ham



(a) Hard ham vs spam, Bernoulli Naive Bayes



(b) Hard ham vs spam, Multinomial Naive Bayes

Figure 2: Confusion matrixes of hard ham and spam

References

[1] Steven S Skiena. *The Data Science Design Manual*. Retrieved 2024-01-20. 2024. URL: https://ebookcentral.proquest.com/lib/gu/detail.action?docID=6312797.

Appendix: Source Code

```
from matplotlib import pyplot
import tarfile
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score
import seaborn as sns

def decode_bytes(bytes, encodings=('utf-8', 'ascii', 'ISO-8859-1')):
    for encoding in encodings:
        try:
            decoded_text = bytes.decode(encoding)
            return decoded_text
        except UnicodeDecodeError:
            continue
    return None

def parse_tar_bz2(file_path):
    emails = []
    try:
        with tarfile.open(file_path, 'r:bz2') as tar:
            for member in tar.getmembers():
                #print("File:", member.name)
                file = tar.extractfile(member)
                if file is not None:
                    content = file.read()
                    emails.append(decode_bytes(content))
    except tarfile.TarError as e:
        print("Error occurred while processing the tar.bz2 file:", e)
    return emails


def evaluate_model(y_test, y_pred, title, classifier):

    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    print(title + " and spam " + classifier + " accuracy:", accuracy)
    print(title + " and spam " + classifier + " precision:", precision)
    print(title + " and spam " + classifier + " recall:", recall)
    print(title + " and spam " + classifier + " F1 score:", 2 * (precision * recall) / (precision + recall))

    conf_matrix = confusion_matrix(y_test, y_pred)
```

```
48      fig, ax = pyplot.subplots(figsize=(8, 6), layout='constrained')
49      sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d',
50                  xticklabels=['Spam', title],
51                  yticklabels=['Spam', title])
52      ax.set_xlabel('Predicted Labels')
53      ax.set_ylabel('True Labels')
54      ax.set_title('Confusion Matrix ' + title + ' and spam ' +
            ↪ classifier)
55      filename = title + '_and_spam_' + classifier + '
            ↪ _confusion_matrix.pdf'
56      filename = filename.replace(' ', '_').lower()
57      fig.savefig(filename, bbox_inches='tight')
58
59
60  def classify_email(emails, labels, title):
61
62      vectorizer = CountVectorizer()
63
64      # Fit CountVectorizer object to email data and
65      # transform email data into a matrix of token counts
66      email_matrix = vectorizer.fit_transform(emails)
67      # Split data into training and test sets, with 20% of data
            ↪ reserved for testing
68      X_train, X_test, y_train, y_test = train_test_split(
            ↪ email_matrix, labels, test_size=0.2)
69
70      # Train classifier (Multinomial Naive Bayes and Bernoulli Naive
            ↪  Bayes)
71      classifierMNB = MultinomialNB()
72      classifierBNB = BernoulliNB()
73      classifierMNB.fit(X_train, y_train)
74      classifierBNB.fit(X_train, y_train)
75
76      # Evaluate the classifier
77      y_predMNB = classifierMNB.predict(X_test)
78      y_predBNB = classifierBNB.predict(X_test)
79
80
81
82      evaluate_model(y_test, y_predMNB, title, "Multinomial Naive
            ↪ Bayes")
83      evaluate_model(y_test, y_predBNB, title, "Bernoulli Naive Bayes
            ↪ ")
84
85  pyplot.rcParams['text.usetex'] = True
86  file_path_easy_ham = "../20021010_easy_ham.tar.bz2"
87  emails_easy_ham = parse_tar_bz2(file_path_easy_ham)
88  file_path_hard_ham = "../20021010_hard_ham.tar.bz2"
89  emails_hard_ham = parse_tar_bz2(file_path_hard_ham)
90  file_path_spam = "../20021010_spam.tar.bz2"
91  emails_spam = parse_tar_bz2(file_path_spam)
92
93  labels_easy_and_spam = [1] * len(emails_easy_ham) + [0] * len(
        ↪ emails_spam)
94  emails_easy_and_spam = emails_easy_ham + emails_spam
95
96  labels_hard_and_spam = [1] * len(emails_hard_ham) + [0] * len(
        ↪ emails_spam)
97  emails_hard_and_spam = emails_hard_ham + emails_spam
98
99  classify_email(emails_easy_and_spam, labels_easy_and_spam, "Easy
        ↪ ham")
```

```
100 classify_email(emails_hard_and_spam, labels_hard_and_spam, "Hard-
    ↪ ham")
```