DAT565/DIT407 Assignment 3

Vaibhav Talari
talari@chalmers.se

Stefán Ólafur Ingimarsson
stefanla@chalmers.se

November 25, 2024

This report is submitted by group 25 for **assignment three** in *Introduction to Data Science & AI.*

1 Problem 1: Spam & Ham

After extracting the tarball, we chose to only read 10 emails from each file by ourselves, since the amount of emails was rather overwhelming. We could see that in the easy ham folder, there are more 'normal' emails, and that the hard ham emails have html code in them which makes us assume that they are ads and are spam. The spam level indicator in some of the mails indicated that as well. The emails in the spam folder has mails that look like scams since they are using promotional language, which indicates they are trying desperately to sell something.

In the second part of the problem, we perfomed a train-test split, where we trained 80% of the data, and tested on 20% of the data.

2 Problem 2: Preprocessing

After performing the train-test split in the previous problem, we started the preprocessing. The emails were converted into numerical form using the Bag of Words model with the CountVectorizer class from Scikit-Learn. The fit_transform method was applied to the training data. That step was taken to teach the algorithm the words in the emails, and to create a document-term matrix, which represented each email as a vector of word counts, where rows represented emails, columns represented unique words in the training dataset. The transform method was applied to the test data using the vocabulary learned from the training data.

3 Problem 3: Easy Ham

3.1 Multinomial Naive Bayesian Classifier

Below are the metrics with easy ham after running with a multinomial naive Bayesian classifier. For concussion matrix refer to table 1

- Accuracy: 97%

- Precision: 100%
- Recall: 81%

Label	Predicted Negative	Predicted Positive	sum
Actual Negative	519(TN)	0(FP)	519
Actual Positive	17(FN)	75(TP)	92
sum	536	75	611

Table 1: Easy Ham: Multinomial Naive Bayesian Classifier Confusion Matrix

3.2 Bernoulli Naive Bayesian Classifier

Below are the metrics with easy ham after running with a Bernoulli naive Bayesian classifier. For concussion matrix refer to table 2

- Accuracy: 92%
- Precision: 100%
- Recall: 47%

Label	Predicted Negative	Predicted Positive	sum
Actual Negative	519(TN)	0(FP)	519
Actual Positive	48(FN)	44(TP)	92
sum	562	44	611

Table 2: Easy Ham: Bernoulli Naive Bayesian Classifier Confusion Matrix

4 Problem 4: Hard Ham

4.1 Multinomial Naive Bayesian Classifier

Below are the metrics with hard ham after running with a multinomial naive Bayesian classifier. For concussion matrix refer to table 3

- Accuracy: 94%
- Precision: 93%
- Recall: 98%

Label	Predicted Negative	Predicted Positive	sum
Actual Negative	45(TN)	7(FP)	52
Actual Positive	1(FN)	98(TP)	99
sum	46	105	151

Table 3: Hard Ham: Multinomial Naive Bayesian Classifier Confusion Matrix

4.2 Bernoulli Naive Bayesian Classifier

Below are the metrics with hard ham after running with a Bernoulli naive Bayesian classifier. For concussion matrix refer to table 4

- Accuracy: 88%
- Precision: 85%
- Recall: 98%

Label	Predicted Negative	Predicted Positive	sum
Actual Negative	36(TN)	16(FP)	52
Actual Positive	1(FN)	98(TP)	99
sum	37	114	151

Table 4: Hard Ham: Bernoulli Naive Bayesian Classifier Confusion Matrix

In the Multinomial Naive Bayes performance, the accuracy is 97% for the easy ham data set, while the hard ham/spam data set has an accuracy of 94%. This indicates that distinguishing spam from hard ham is more challenging than from easy ham.

The precision for problem 3 was 100%, meaning that all emails classified as spam were indeed spam. For problem 4, the precision decreased to 93%, showing that some hard ham emails were incorrectly classified as spam.

As for the recall in both problems, problem 3 listed recall as 81%, indicating that some spam emails were misclassified as ham. In problem 4, recall improved significantly to 98%, suggesting that nearly all spam emails were correctly identified, however it misclassified some hard ham.

The Bernoulli Naive Bayes classifier performed worse than the Multinomial Naive Bayes, with notable differences between the problems. In problem 3, the accuracy was 92%, which is decent, but it was lower than the Multinomial Naive Bayes. The accuracy for problem 4 dropped down to 88%, indicating more difficulty separating spam from hard ham.

The precision for problem 3 was at 100%, the same result as for the Multinomial Naive Bayes. However the accuracy for problem 4 dropped to 85%, which indicates that the classifier had more difficulty classifying hard ham as spam.

As for the recall, problem 3 had a recall of 47%, showing that many spam emails were misclassified as ham. In problem 4 that number improved to 98%, a similar number to the Multinomial classifier, but with a higher number of false positives.

Easy ham is simpler to distinguish from spam since it likely has fewer similarities in its structure and content. Hard ham shares more similarities with spam, such as promotional language and vocabulary, making it more challenging for the classifier. Overall the Multinomial Bayes performed consistently better than the Bernoulli Naive Bayes. This might be due to the Multinomial Naive Bayes is better suited for data with features that represent discrete frequencies and word counts.[2] The Bernoulli Naive Bayes is used for the classification of binary features, such as ’Yes’, ’No’, ’1’, or ’0’.[1] This may lead to less informative feature representation and lower performance. Overall the Multinomial Bayes consistently outperformed the Bernoulli Naive Bayes in both tasks, especially in handling more complex cases like hard ham.

5 Assignment code

Here is our code that shows how we performed the train-test and Naive Bayes.

```
set_one = easy_ham_data + spam_data
set_one_labels = [0] * len(easy_ham_data) + [1] * len(spam_data)

set_two = hard_ham_data + spam_data
set_two_labels = [0] * len(hard_ham_data) + [1] * len(spam_data)

set_one_train, set_one_test, set_one_train_label,
set_one_test_label = train_test_split(
    set_one, set_one_labels, test_size=0.2, random_state=42
)

set_two_train, set_two_test, set_two_train_label,
set_two_test_label = train_test_split(
    set_two, set_two_labels, test_size=0.2, random_state=42
)

def vectorize_data(taining_data, testing_data):
 vectorizer = CountVectorizer()
 training_matrix = vectorizer.fit_transform(taining_data)
 testing_matrix = vectorizer.transform(testing_data)
 print(f"Training␣data␣shape:␣{training_matrix.shape}")
 print(f"Testing␣data␣shape:␣{testing_matrix.shape}")
 return training_matrix, testing_matrix

def classification_multinomialNB(training_matrix,
training_label, testing_matrix):
mnb = MultinomialNB()
mnb.fit(training_matrix, training_label
mnb_predection_matrix = mnb.predict(testing_matrix)
return mnb_predection_matrix

def classification_bernoulliNB(training_matrix,
training_label, testing_matrix):
bnb = BernoulliNB()
bnb.fit(training_matrix, training_label
bnb_predection_matrix = bnb.predict(testing_matrix)
return bnb_predection_matrix
```

References

[1] unknown. “About Marstrand”. In: (2023). URL: https://www.geeksforgeeks.org/bernoulli-naive-bayes/.

[2] unknown. “Multinomial Naive Bayes”. In: (2024). URL: https://www.geeksforgeeks.org/multinomial-naive-bayes/.