DAT565/DIT407 Assignment 3

David Duong
thuan@chalmers.se

Savinjith Walisadeera
savwal@chalmers.se

October 1, 2024

Problem 1

A)

We can differentiate between easy ham, hard ham and spam by reading the email. Easy ham is usually a email conversation between two people. The difference between hard ham and easy ham is that hard ham emails are not necessarily written by a human. There can be part or all of it which is copied or a generated response. Spam emails are all generic responses or emails not specified to the receiver, they often contain words that are uncommon in conversations and links.

Problem 2

We split the data with `train-test-split` into training and testing data with the ratio 75:25. We organize all the training emails into vectors using `fit-transform()` and the test emails with `transform()`.

Problem 3

Multinomial	Pred. Positive	Pred. Negative	Sum
Actual Positive	641	1	642
Actual Negative	27	94	121
Sum	668	95	

Table 1: Easy ham confusion matrix for multinomal naive Bayesian classifier

Bernoulli	Pred. Positive	Pred. Negative	Sum
Actual Positive	642	0	642
Actual Negative	69	52	121
Sum	711	52	

Table 2: Easy ham confusion matrix for Bernoulli naive bayesian classifier

	Accuracy	Precision	Recall
Multinomial	0.9633	0.9595	0.9984
Bernoulli	0.9095	0.9029	1.0000

Table 3: Classifier metrics for easy ham + spam

Problem 4

Multinomial	Pred. Positive	Pred. Negative	Sum
Actual Positive	56	8	64
Actual Negative	3	121	124
Sum	59	129	

Table 4: Hard ham confusion matrix for Multinomial naive Bayesian classifier

Bernoulli	Pred. Positive	Pred. Negative	Sum
Actual Positive	43	21	64
Actual Negative	2	122	124
Sum	45	143	

Table 5: Hard ham confusion matrix for Bernoulli naive Bayesian classifier

	Accuracy	Precision	Recall
Multinomial	0.9414	0.9491	0.8750
Bernoulli	0.8776	0.9556	0.6719

Table 6: Classifier metrics for hard ham + spam

After changing all easy ham emails to hard ham the accuracy, precision and recall values all decrease. This is because hard ham contains words that are more used in spam emails, which makes it harder for the classifiers to differentiate between them. This causes the metrics to decrease between the two.

A Code

```
1  import numpy as np
2  from sklearn.model_selection import train_test_split
3  from sklearn.feature_extraction.text import
       CountVectorizer
4  import os
5  from pathlib import Path
6
7  #Extract the files
8  file_list_easy = []
9  file_list_hard = []
10 types_list_easy = []
```

```
types_list_hard = []
tar_list_easy = []
tar_list_hard = []

directory = './easy_ham'
for file in os.listdir(directory):
    content = Path(directory, file).read_text(encoding
        ='latin-1')
    file_list_easy.append(content)
    types_list_easy.append('ham')

directory = './hard_ham'
for file in os.listdir(directory):
    content = Path(directory, file).read_text(encoding
        ='latin-1')
    file_list_hard.append(content)
    types_list_hard.append('ham')

directory = './spam'
for file in os.listdir(directory):
    content = Path(directory, file).read_text(encoding
        ='latin-1')
    file_list_easy.append(content)
    types_list_easy.append('spam')
    file_list_hard.append(content)
    types_list_hard.append('spam')

#Split the easy data 75-25
file_train, file_test, type_train, type_test =
    train_test_split(file_list_easy, types_list_easy,
    test_size=0.25, random_state=6)
#Vectorize
vectorizer = CountVectorizer()
text_vector_train = vectorizer.fit_transform(
    file_train)
text_vector_test = vectorizer.transform(file_test)

#Train our models
from sklearn.naive_bayes import MultinomialNB
model_multinomial = MultinomialNB()
model_multinomial.fit(text_vector_train,type_train)

from sklearn.naive_bayes import BernoulliNB
model_bernoulli = BernoulliNB()
model_bernoulli.fit(text_vector_train, type_train)

#Evaluate the models
multinomial_TP = 0
multinomial_FP = 0
multinomial_TN = 0
```

```
55 multinomial_FN = 0
56
57 for i in range(np.shape(text_vector_test)[0]):
58     prediction = model_multinomial.predict(
           text_vector_test[i])[0]
59     answer = type_test[i]
60
61     if prediction == 'ham':
62         if answer == 'ham':
63             multinomial_TP += 1
64         elif answer == 'spam':
65             multinomial_FP += 1
66     elif prediction == 'spam':
67         if answer == 'spam':
68             multinomial_TN += 1
69         elif answer == 'ham':
70             multinomial_FN += 1
71
72
73
74 multinomial_accuracy = (multinomial_TN+multinomial_TP)
       /(multinomial_TP+multinomial_FP+multinomial_FN+
       multinomial_TN)
75 multinomial_precision = multinomial_TP/(multinomial_TP
       +multinomial_FP)
76 multinomial_recall = multinomial_TP/(multinomial_TP+
       multinomial_FN)
77
78 bernoulli_TP = 0
79 bernoulli_FP = 0
80 bernoulli_TN = 0
81 bernoulli_FN = 0
82
83 for i in range(np.shape(text_vector_test)[0]):
84     prediction = model_bernoulli.predict(
           text_vector_test[i])[0]
85     answer = type_test[i]
86
87     if prediction == 'ham':
88         if answer == 'ham':
89             bernoulli_TP += 1
90         elif answer == 'spam':
91             bernoulli_FP += 1
92     elif prediction == 'spam':
93         if answer == 'spam':
94             bernoulli_TN += 1
95         elif answer == 'ham':
96             bernoulli_FN += 1
97
```

```
98  bernoulli_accuracy = (bernoulli_TN+bernoulli_TP)/(
        bernoulli_TP+bernoulli_FP+bernoulli_FN+bernoulli_TN
        )
99  bernoulli_precision = bernoulli_TP/(bernoulli_TP+
        bernoulli_FP)
100 bernoulli_recall = bernoulli_TP/(bernoulli_TP+
        bernoulli_FN)
101
102
103 print('accuracy, precision, recall')
104 print('Multinomial: ', multinomial_accuracy,
        multinomial_precision, multinomial_recall)
105 print('Bernoulli: ', bernoulli_accuracy,
        bernoulli_precision, bernoulli_recall)
106
107 print(multinomial_TP, multinomial_FN, multinomial_FP,
        multinomial_TN)
108 print(bernoulli_TP, bernoulli_FN, bernoulli_FP,
        bernoulli_TN)
109
110 #Repeat experiment with hard_ham
111 file_train, file_test, type_train, type_test =
        train_test_split(file_list_hard, types_list_hard,
        test_size=0.25, random_state=6)
112
113 vectorizer = CountVectorizer()
114 text_vector_train = vectorizer.fit_transform(
        file_train)
115 text_vector_test = vectorizer.transform(file_test)
116
117 #Train our models
118 from sklearn.naive_bayes import MultinomialNB
119 model_multinomial = MultinomialNB()
120 model_multinomial.fit(text_vector_train,type_train)
121
122 from sklearn.naive_bayes import BernoulliNB
123 model_bernoulli = BernoulliNB()
124 model_bernoulli.fit(text_vector_train, type_train)
125
126 #Evaluate the models
127 multinomial_TP = 0
128 multinomial_FP = 0
129 multinomial_TN = 0
130 multinomial_FN = 0
131
132 for i in range(np.shape(text_vector_test)[0]):
133     prediction = model_multinomial.predict(
            text_vector_test[i])[0]
134     answer = type_test[i]
135
```

```
    if prediction == 'ham':
        if answer == 'ham':
            multinomial_TP += 1
        elif answer == 'spam':
            multinomial_FP += 1
    elif prediction == 'spam':
        if answer == 'spam':
            multinomial_TN += 1
        elif answer == 'ham':
            multinomial_FN += 1



multinomial_accuracy = (multinomial_TN+multinomial_TP)/(multinomial_TP+multinomial_FP+multinomial_FN+multinomial_TN)
multinomial_precision = multinomial_TP/(multinomial_TP+multinomial_FP)
multinomial_recall = multinomial_TP/(multinomial_TP+multinomial_FN)

bernoulli_TP = 0
bernoulli_FP = 0
bernoulli_TN = 0
bernoulli_FN = 0

for i in range(np.shape(text_vector_test)[0]):
    prediction = model_bernoulli.predict(text_vector_test[i])[0]
    answer = type_test[i]

    if prediction == 'ham':
        if answer == 'ham':
            bernoulli_TP += 1
        elif answer == 'spam':
            bernoulli_FP += 1
    elif prediction == 'spam':
        if answer == 'spam':
            bernoulli_TN += 1
        elif answer == 'ham':
            bernoulli_FN += 1

bernoulli_accuracy = (bernoulli_TN+bernoulli_TP)/(bernoulli_TP+bernoulli_FP+bernoulli_FN+bernoulli_TN)
bernoulli_precision = bernoulli_TP/(bernoulli_TP+bernoulli_FP)
bernoulli_recall = bernoulli_TP/(bernoulli_TP+bernoulli_FN)

```

```
177
178 print('accuracy, precision, recall')
179 print('Multinomial: ', multinomial_accuracy,
        multinomial_precision, multinomial_recall)
180 print('Bernoulli: ', bernoulli_accuracy,
        bernoulli_precision, bernoulli_recall)
181
182 print(multinomial_TP, multinomial_FN, multinomial_FP,
        multinomial_TN)
183 print(bernoulli_TP, bernoulli_FN, bernoulli_FP,
        bernoulli_TN)
```