INTRODUCTION TO DATA MINING: PROJECT PART 2

Ching-Han Kuo (r0911555)

1 Feature Extraction

1.1 Preprocessing

Firstly, I divide the text into three groups (referred to as "raw text"): subject, content, and full text. In the email text, the one that starts with "Subject:" and ends with a first ending of a line symbol "`\r\n`" will be the subject text, and the rest will be the content text. For the full text, it is the combination of subject and content text. Secondly, I process the raw text into new text groups (referred to as "processed text") by eliminating punctuation and stop-words and lemmatising each word. I assume that the processed text may contain more qualitative information with less noise which can be beneficial for model training. Still, I keep both raw and processed text since information in the raw text, such as punctuation, can be useful.

Table 1: Example of Preprocessing

Text	**Subject: hpl nom for january 9 , 2001\r\n (see attached file : hplnol 09 . xls)\r\n - hplnol 09 . xls**
Subject (Raw)	hpl nom for january 9 , 2001
Subject (Processed)	hpl nom january 9 2001
Content (Raw)	(see attached file : hplnol 09 . xls) - hplnol 09 . xls
Content (Processed)	attach file hplnol 09 xls hplnol 09 xls
Full (Processed)	hpl nom january 9 2001 attach file hplnol 09 xls hplnol 09 xls

1.2 Types of Features

Originally, I would like to extract, for instance, the number of URLs and files in each email as features. However, it turns out that the text of the email was already tokenised (with white space) which makes it hard to extract information. As a result, I drop these types of features. I end up using three types of features: length of text, bag of words, and word similarity.

Length of Text

I use the raw text in this part since I believe that the information of length in the original text may reflect more accurately the message of the email. I count the text with three levels: characters, words, and rows (each "`\r\n`" in the text indicates a row). In addition, I add calculations of punctuation since it can also be a useful feature. Detailed information on features is shown in the table below.

Table 2: Features of Length of Text

	Subject	**Content**
Counts of Characters	✓	✓
Avg of Characters in Each Word	✓	✓
Avg of Characters in Each Row	*	✓
Counts of Words	✓	✓
Avg of Words in Each Row	*	✓
Counts of Rows	*	✓
Counts of Punctuation	✓	✓
Proportion of Punctuation in Characters	✓	✓
Proportion of Punctuation in Words	✓	✓

* Subject only has one row.

Bag of Words

I use the tf-idf calculation for this part since it can not only show the frequency of the word but also catch its importance in the text. I did pilot runs on models that I am going to use using the bag of words extraction on different groups of texts: subject (raw) + content (raw), full (raw), subject (processed) + content (processed), full (processed). It turns out that the accuracy of prediction would be the best (although very similar) if I extract bag of words from subject(processed) and content(processed) separately and then combine them. Therefore, I will approach the bag of words extraction with this method. Extracting bag of words from all the text groups I mentioned and combining them has the potential to make the prediction better, however, it may also cause more running times since the extracted features would be a lot. As a result, I only choose one text group for this project.

Word Similarity

I assume that the connection between the email itself and the spam emails we already know can be important for classification. Therefore, for this type of feature, I compute the similarity (cosine similarity) between the email text with the text that is the combination of emails labelled with “spam” in the training set. Detailed information on this feature is shown in the table below.

Table 3: Features of Word Similarity

Similarity of	**Group of Spam Emails’**
Each Email’s	Subject (raw)
	Content (raw)
	Full (raw)
	Subject (processed)
	Content (processed)
	Full (processed)

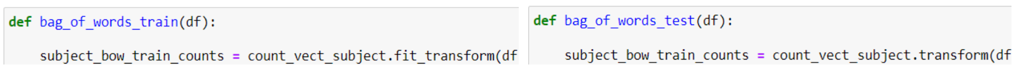
You can notice that I compute the number of all kinds of text groups. Since there are only six figures for each email, I assume that they will provide more information but not cause more running times.

In addition, I also did a trial run for this feature, and it seems that because the computed similarity is between 0 and 1, the difference between each email is not big. The models could not detect it very well and performed poor predictions (almost the same as the dummy baseline). As a result, the figure of this feature needs to be multiplied by 100 before putting them into the models.

2 Model Training

2.1 Feature Extraction from the Test Set

To extract features from the test set correctly, I carefully divide my extraction into two functions for the training set and test set respectively, to make sure I would not accidentally use the fit_transform on the test set.



```
def bag_of_words_train(df):

    subject_bow_train_counts = count_vect_subject.fit_transform(df
```

```
def bag_of_words_test(df):

    subject_bow_train_counts = count_vect_subject.transform(df
```

Figure 1: Extraction Function for Train (left) and Test (right) Sets

2.2 Type of Classifiers

The three types of classifiers I choose are linear classifier (logistic regression), probabilistic classifier (Naïve Bayes), and decision tree. Additionally, since the neural network classifier has logistical activation, I also apply it to see the improvement of it compared to the original logistic classifier. The dummy baseline I choose is to always predict the majority of the training set.

I was going to also choose kNN and Random Forest models, however, they all need an additional step to compute the best parameter (best numbers of neighbours and trees) to explore their best potential. I tried them but it took too long time to run so I ended up dropping them.

Below you can see the predictions of the first 10 emails from each classifier.

Table 4: Examples of Predictions

Real	**Dummy**	**Logistic**	**Naïve**	**Decision**	**Nerual**
ham	ham	ham	ham	ham	ham
ham	ham	ham	ham	ham	ham
spam	ham	spam	spam	spam	spam
ham	ham	ham	ham	ham	ham
ham	ham	ham	ham	ham	ham
ham	ham	ham	ham	ham	ham
spam	ham	ham	ham	ham	ham
ham	ham	ham	ham	ham	ham
spam	ham	ham	ham	spam	spam
ham	ham	ham	ham	ham	ham

3 Performance Evaluation

3.1 Performance of Classifiers

I use five figures to determine the performance of each classifier. Firstly, I use the classifiers to conduct 10-fold cross-validation on the training set and compute the average of accuracy. Secondly, I test the classifiers on the test set and calculate the accuracy, precision, recall, and f1 score. The full result of the performance can be found in the table below.

Table 5: Performance of Each Classifier

	Dummy		**Logistic**		**Naïve**		**Decision**		**Neural**	
	Num	95%CI	Num	95%CI	Num	95%CI	Num	95%CI	Num	95%CI
CV	0.72	0.020	0.86	0.013	0.85	0.013	0.93	0.009	0.97	0.006
Accuracy	0.69	0.019	0.84	0.015	0.85	0.015	0.93	0.009	0.96	0.008
Precision	0.34		0.83		0.85		0.92		0.96	
Recall	0.00		0.63		0.63		0.83		0.94	
F1	0.00		0.71		0.72		0.87		0.94	

* Green/Red indicates it is significantly lower/higher than other models. (t-test)

From the result, we can observe several interesting points:
1. The performance of my classifier performs way better than the dummy baseline, all of the scores of other classifiers are higher than the dummy classifier. This can be evidence that my classifiers can predict something instead of just guessing.
2. The neural network classifier seems to be the best in my experience. All of its scores are the highest compared to other classifiers. Furthermore, if we conduct a t-test on the results of cross-validation, we can see that the average accuracy of the neural network classifier is significantly higher than others.
3. I also conduct a one-sample t-test between the accuracy on cross-validation and the test set, and I see no significant difference, indicating that all classifiers may not have an over-fitting problem from my experiment settings.

To sum up, in my experiment on this project, the neural network model performs the best. However, in my experience, it took way too long time to run compared to other classifiers, and the decision tree classifier also has over 90 percent accuracy. Therefore, I would claim that the decision tree classifier can also be good for practical usage.

3.2 Contribution of Features

I also use cofe to compute the contribution of features, and run the classifer with each type of features individually.

Table 6: Contribution of Features

Logistic		**Naïve**		**Decision**		**Neural**	
Feature	Type	Feature	Type	Feature	Type	Feature	Type
subject_pro	similarity	content_c_count	length	subject_pro	similarity	subject_new	bag of words
subject_c_avg_w	length	content_w_count	length	content_enron	bag of words	content_ken	bag of words
content_w_avg_l	length	content_pun_count	length	content_http	bag of words	content_employee	bag of words
content_c_avg_w	length	content_c_avg_l	length	subject_00	bag of words	content_2004	bag of words
full_pro	similarity	full_raw	similarity	subject_c_avg_w	length	content_question	bag of words
content_pro	similarity	content_raw	similarity	content_c_avg_w	length	content_enron	bag of words
subject_pun_count	length	subject_c_count	length	full_raw	similarity	content_daren	bag of words
content_raw_sim	similarity	content_s_count	length	content_thanks	bag of words	content_let	bag of words
content_c_avg_l	length	content_w_avg_l	length	content_w_avg_s	length	content_pm	bag of words
subject_c_count	length	full_raw	similarity	subject_new	bag of words	subject_picture	bag of words

Table 7: Accuracy of Prediction by Each Type of Feature

	Logistic	Naïve	Decision	Neural
Length	0.76	0.57	0.57	0.82
Bag of Words	0.98	0.92	0.92	0.99
Similarity	0.81	0.70	0.70	0.83

It can be observed that the classifiers with higher accuracy (decision tree and neural network) weighted bag of words more. In addition, if we only use bag of words as features, the performance of each classifier go way higher. In conclusion, the other types of features (length and similarity) may not be good features for spam classification.