Naive Bayes Implementation

– Spam Emails Classification and Filtering

Abstract

The goal of the experiment is to learn Naive Bayes classifiers on spam email filtering.

Spam emails are the majority class(66.67%) in the given dataset. Thus, the data is unbalanced. This bias would affect the classification performances in the later stage.

Here, vectorization based on term counts and on TF-IDF are used to transform textual data into numeric data as a vector space model. There are three Naive Bayes algorithms used from sklearn modules: Multinomial, Bernoulli, Complementary. We tested the overall performances on a dataset with different class proportions, by combinations of different vectorizers and Bayes algorithms.

On the raw data set (majority: spam), the combination of count vectorization and Multinomial Naive Bayes performs the best, with: Accuracy score: 0.9575208422859642, Precision score: 0.970570217199754, Recall Score: 0.9653073113894554, F1 Score: 0.9679301817518213.

On a larger resampled data set (majority: ham, way to resample introduces extra bias), TF-IDF + ComplementNB behaves better. Accuracy score: 0.9516975848792439, Precision score: 0.9200046040515654, Recall Score: 0.9359484777517565, F1 Score: 0.927908056651962.

A generalization model is experimented on another dataset from Kaggle[1]. Associated analysis is attached at the end of the report.

The extended experiment uses weighted class TF-IDF for vectorization.

Best performance is: the combination of content + from + to + date with Multinomial Naive Bayes with accuracy: 0.9762443438914027 precision: 0.9347408829174664, recall: 0.9838383838383838, f1: 0.9586614173228347. However, there exists lots of room for improvement.

1 Introduction

According to statistics, billions of spam emails are sent per day. Therefore, spam filtering is a key function for email service providers to provide. Naive Bayes has always been a widely used algorithm under this implementation, due to its accuracy and calculation speed on dealing with large amounts of data.

Labeled dataset used for this experiment is from Trec06's Chinese spam data set. Kaggle's Email Spam Dataset[1] is used here for extended experiments.

2 Programming Modules

Numpy and Pandas are used for data processing. Matplotlib, wordcloud and shap are used for analyzing extracted features and training result visualization. Both vectorizers and Naive Bayes classifiers are directly called from the sklearn modules. For testing on the Kaggle data set, we also use the python email module to construct parsers for extracting information.

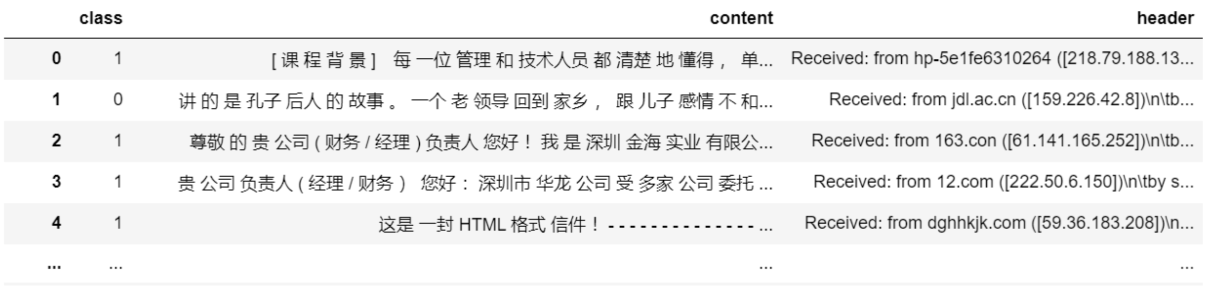
3 Data Analyze

3-1. Data Read and Dataframe Construct

We read the emails that have been cut by terms for constructing a dataframe. There are three types of data we need to collect:

(1) class(label): if the mail is a spam or non a spam
(2) header of an email, e.g. sender, receiver, title
(3) content(body) of an email

The data frame looks like this:



	class	content	header
0	1	[课 程 背 景] 每 一位 管理 和 技术人员 都 清楚 地 懂得 ， 单...	Received: from hp-5e1fe6310264 ([218.79.188.13...
1	0	讲 的 是 孔子 后人 的 故事 。 一个 老 领导 回到 家乡 ， 跟 儿子 感情 不 和...	Received: from jdl.ac.cn ([159.226.42.8])\n\tb...
2	1	尊敬 的 贵 公司 (财务 / 经理) 负责人 您好 ！ 我 是 深圳 金海 实业 有限公...	Received: from 163.con ([61.141.165.252])\n\tb...
3	1	贵 公司 负责人 (经理 / 财务 ） 您好 ： 深圳市 华龙 公司 受 多家 公司 委托 ...	Received: from 12.com ([222.50.6.150])\n\tby s...
4	1	这是 一封 HTML 格式 信件 ！ - - - - - - - - - - - - - - ...	Received: from dghhkjk.com ([59.36.183.208])\n...
...	...	...	...

3-2. General Data Cleanup and Train/Test Set Split

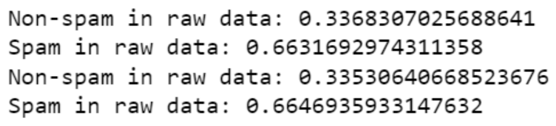
We detected if any duplicates or constant features exist in this constructed dataframe. With the missingno module, we confirm that there exists no missing values in the dataset. Finally, shuffling data is applied so that data fed into the model can be more random; this helps make the model more generalized.

In the experiment on Trec06's, only email contents are used for classification. The train set and test set are therefore both splitted from the class+content columns, without the header information.

3-3. Evaluate Representativity of Data

Inspired by an online project “Building a Spam Filter with Naive Bayes”[2], we evaluate if the data is representative by the proportion of spams and hams in the whole dataset.

In the given set, we have around 66% spams and 34% non-spams(hams). This result reflects that the data is not



```
Non-spam in raw data: 0.3368307025688641
Spam in raw data: 0.6631692974311358
Non-spam in raw data: 0.33530640668523676
Spam in raw data: 0.6646935933147632
```

necessarily representative. As in practice most emails should be non-spam.

This unbalance might bring in bias in the training stage. For example, Complement Naive Bayes is particularly suited to work with imbalanced datasets, whilst Multinomial Naive Bayes does not perform well in this scenario.

4 Content Feature Extraction

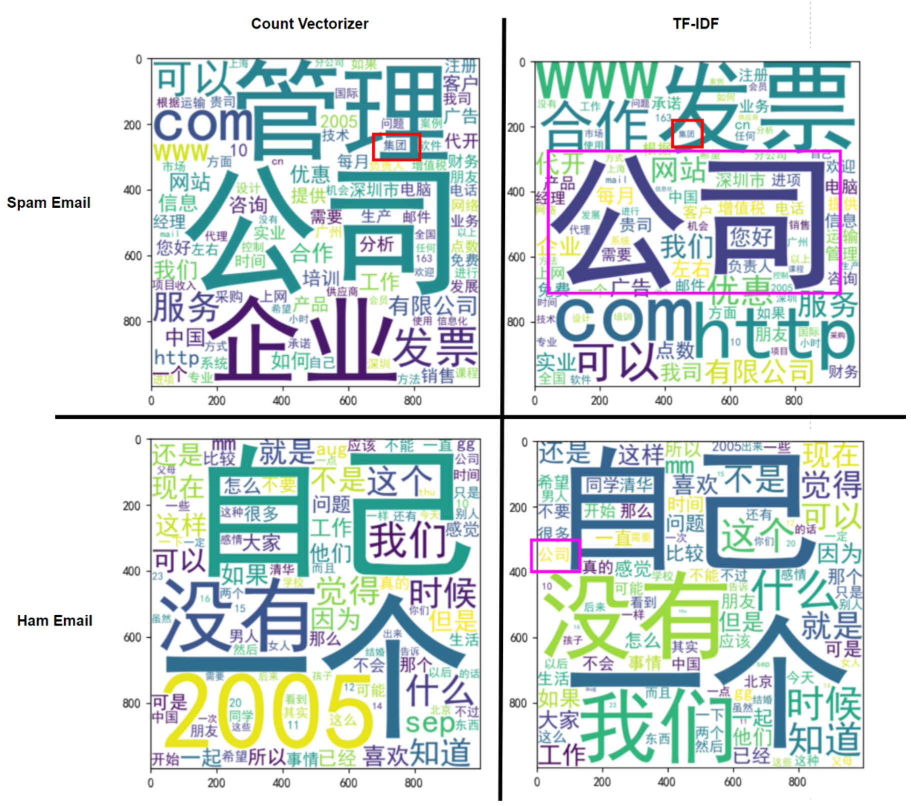
In the previous sections, we clean up the initial texts. Now, we need to transform it into its features to be used for modeling. Document data is not computable so it must be transformed into numerical data such as a vector space model. This process is generally called feature extraction of document data, or Text Vectorization[3].

There are different types of extraction techniques, e.g. Countvectorizer, TF-IDF. TF-IDF is meant for rendering more importance to the rare words. It so happens that if you rely on word counts alone, the unimportant words like ‘the’ , ‘and’ etc. will get more importance because they tend to get used more often.

There is no conclusion to which is better, but from personal use, TF-IDF will usually be stronger in modeled data, e.g. spam classifier. However, the calculation of TF-IDF is not 100% science-proof. The nature of this algorithm will end up with the majority of words chosen being from the majority class in the dataset, and leaving the minority class less represented.

4-1. Vectorization and Feature Analyze

As the given email contents are already splitted by terms, we use (1,1) ngram for this process. Top 100 (most important) features are extracted for hams and spams for compare and contrast.



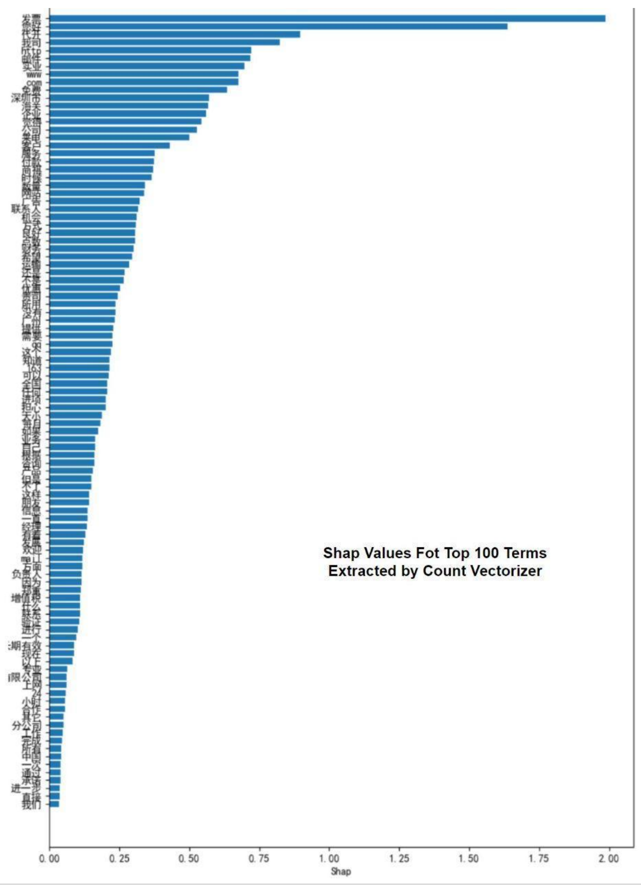
The reason why count and TF-IDF extracts different features from the same contents is that: CountVectorizer simply counts the number of times a word appears in a document (using a bag-of-words approach), while TF-IDF Vectorizer takes into account not only how many times a word appears in a document but also how important that word is to the whole corpus [4].

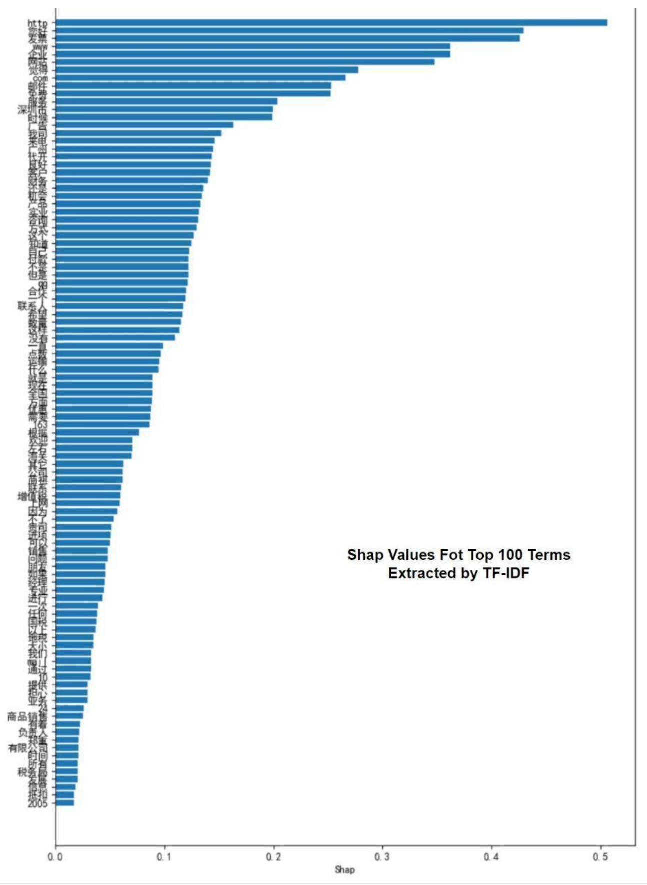
Based on the wordcloud maps above, below observations are made:

(1) CountVectorizer and TF-IDF have main differences on feature extraction results for low-mid frequency(importance) words. The most frequent terms extracted by two methods are identical.
(2) For less-regular terms, the frequencies(importance) calculated by two algorithms are different. For example:
    - On spam emails' contents, "集团" in countvectorizer is more important than in TF-IDF.
    - Numeric terms extracted by the countvectorizer have higher frequencies than in TF-IDF. For example, for English content extraction, words like "it", "is" etc. are seen as highly valuable by countvectorizer.
(3) We need to evaluate if the features extracted can represent the classification characteristics for both spam and ham emails.
    - From the image above, it is easy to see that key feature terms and associated frequencies for two types have wide variations. For example, term "公司" has high importance (repetition) in spam emails, and low in ham emails.
    - Terms that have such "characteristics" may be a "valuable and effective" feature for the spam classifier.

4-2. Explainability and Interpretability

A logistic regression is used to validate the Shapley values of extracted features from email contents.





Shapley values are axiomatically unique in that they satisfy desirable properties to measure the contribution of each feature towards the final outcome[5]. This method is agnostic, consistent, and can handle complex model behavior. Above is the graph for extracted features with top 100 Shapley values.

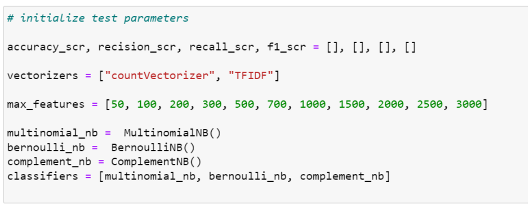
Using the count vectorizer, we can see that the importance of terms drops down massively from the top-2 terms to the less-important ones; whilst TF-IDF has a more gentle downward trend.

In the Shapley graph, "公司" has more interpretability over the whole model built by count vectorizer, compared to TF-IDF. The reason could be: Our data set itself is imbalanced. Thus, by calculating only the count of the term appearances, it suggests a high interpretability as we have more spam emails.

At this point, we could make an assumption that under a balanced data set, TF-IDF may work better. With an unbalanced data set, features extracted by count may have more significance, but can also introduce bias.

5 Naive Bayes Compare and Contrast

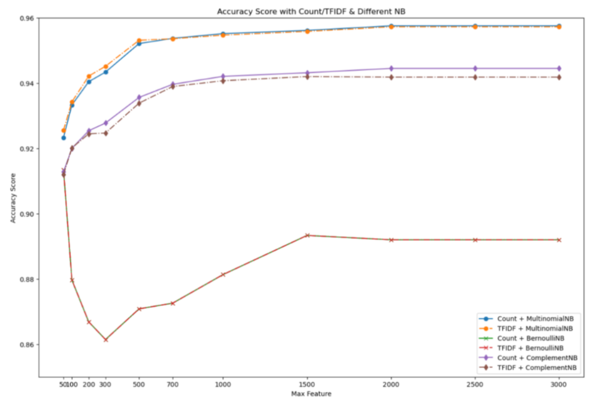
Accuracy, precision, recall and F1 scores are used to evaluate the model performances.

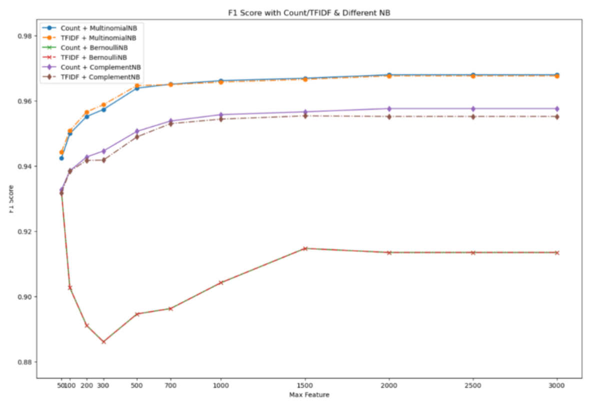


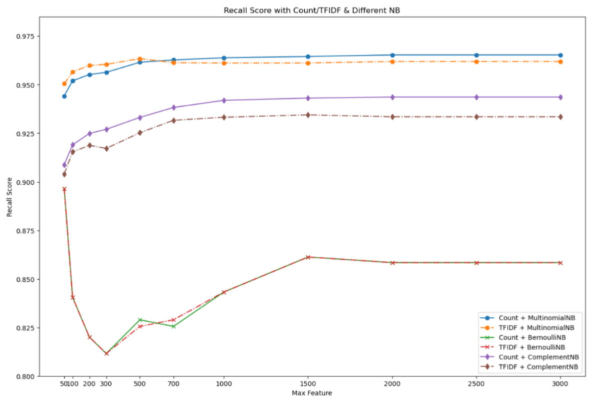
```
# initialize test parameters

accuracy_scr, recision_scr, recall_scr, f1_scr = [], [], [], []

vectorizers = ["countVectorizer", "TFIDF"]

max_features = [50, 100, 200, 300, 500, 700, 1000, 1500, 2000, 2500, 3000]

multinomial_nb =  MultinomialNB()
bernoulli_nb =  BernoulliNB()
complement_nb = ComplementNB()
classifiers = [multinomial_nb, bernoulli_nb, complement_nb]
```

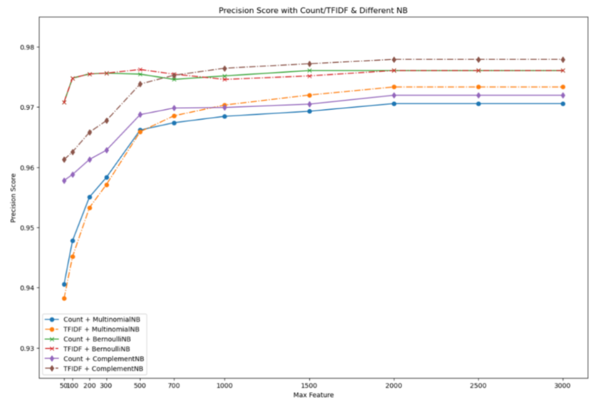
In this section, we analyze the performance of different vectorization and Naive Bayes models on train/validation sets with cross validation mean scores. The validation set is splitted from the original train set.

Above are the parameters and algorithms we run for the test.









5-1 Accuracy Score

Overall accuracy in machine learning classification models can be misleading when the class distribution is imbalanced, and it is critical to predict the minority class correctly. In this case, the class with a higher occurrence may be correctly predicted, leading to a high accuracy score, while the minority class is being misclassified. This gives the wrong impression that the model is performing well when it is not[6].

This means to validate which model performs better on the given data set can not rely only on the accuracy score.

Reading from the accuracy score graph, we could see that Bernoulli Naive Bayes does not have good prediction performance as the other Naive Bayes models. It reaches the lowest accuracy when the max feature amount is around 300. Vectorizing by count or TFIDF works the same for Bernoulli NB.

Both MultinomialNB and ComplementNB accuracy scores have an upward trend as the max feature amount grows up, then trending towards stability. For these two models, vectorized by Count or TF-IDF does not have huge differences over accuracy. But in general, the count method performs slightly better.

MultinomialNB gives out the best accuracy among three NB models, however, as our data set is unbalanced, we cannot make a conclusion based only on accuracy scores.

5-2 Precision Score and Recall Score

Precision can be seen as a measure of quality, and recall as a measure of quantity[7].

Models need high recall when you need output-sensitive predictions. For example, predicting cancer or predicting terrorists needs a high recall, in other words, you need to cover false negatives as well. It is ok if a non-cancer tumor is flagged as cancerous but a cancerous tumor should not be labeled non-cancerous.

Similarly, we need high precision in places such as recommendation engines, spam mail detection, etc. Where you don’t care about false negatives but focus more on true positives and false positives. It is ok if spam comes into the inbox folder but a really important mail shouldn’t go into the spam folder[7].

Based on the math formula, it is fair to say that precision score is a useful measure of the success of prediction when the classes are very imbalanced(here, spam:ham = 2:1). If in our model, the proportion of ham emails increased, then the precision score should drop.

In general, the multinomialNB has the lowest precision scores and highest recall scores, which indicates that most of its predicted labels are correct when compared to the training labels.

The BernoulliNB model has high precision and low recall. This indicates that most of its predicted labels are incorrect.

Precision and recall for ComplementNB have similar trends. Comparatively high precision and recall are desirable. To get a comprehensive understanding of the performance, F1 should also be introduced.

Another conclusion we can get from the precision graph and recall graph is that, with the ComplementNB classifier, vectorization with TF-IDF provides better prediction quality for both ham and spam emails, but count returns better result regarding the task (here to identify spam emails.)

5-3 F1 Score

F1 score is a popular way to measure models as it provides accurate results for both balanced and imbalanced dataset. In plain words, it tells you the model's balanced ability to both capture positive cases (recall) and be accurate with the cases it does capture (precision). High F1 score usually suggests that the model predicts each observation correctly[8].

Based on the F1 score graph, and remains consistent to other graphs, Bernoulli is not a suitable classifier for our task (on our dataset).

Count vectorization still performs slightly better than TF-IDF. As mentioned in Section 3.3, the reason could be: our data set itself is unbalanced. Thus, by calculating only the count of the term appearances, it suggests a high interpretability as we have more non-spam emails. However, if we were to generalize the model to a data set that is more similar to real-life ham/spam proportion, the performance may vary.

5-4 Section Conclusion

BernoulliNB takes, in most of the cases, only binary values. The most general example is where we check if each value will be whether or not a word appears in a document. That is a very simplified model. In cases where counting the word frequency is less important, Bernoulli may give better results. The advantages of BernoulliNB include better performance on some datasets with shorter documents. Due to the imbalance of our dataset, term frequency/importance can have more impact over the classification rule compared to whether a term has occurred in an email or not.

Multinomial Naive Bayes does not perform very well on imbalanced datasets, and Complementary NB on the contrary. Imbalanced datasets are datasets where the number of examples of some class is higher than the number of examples belonging to other classes (see from section 4.2). If we only rely on F1 scores, we may have an opposite conclusion on multinomialNB's behavior. Thus, in the next section, we will try balancing out the dataset and see how the balance of the dataset can affect Bayes learning.

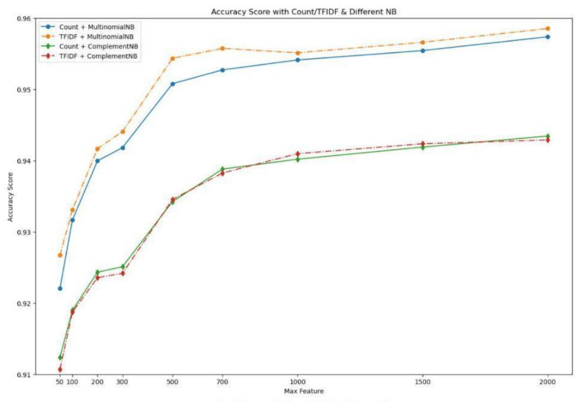
6 Model Generalize on Test Set

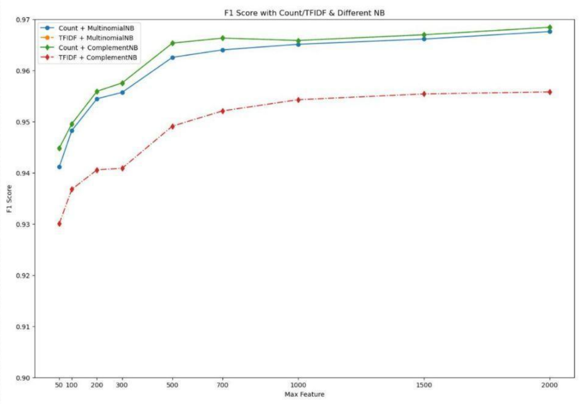
Based on Section5's conclusion, we will focus on testing Complementary Naive Bayes and Multinomial Naive Bayes on the test set.

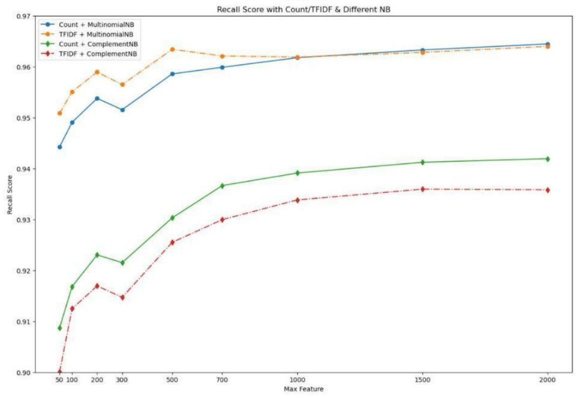
Three datasets are tested separately:

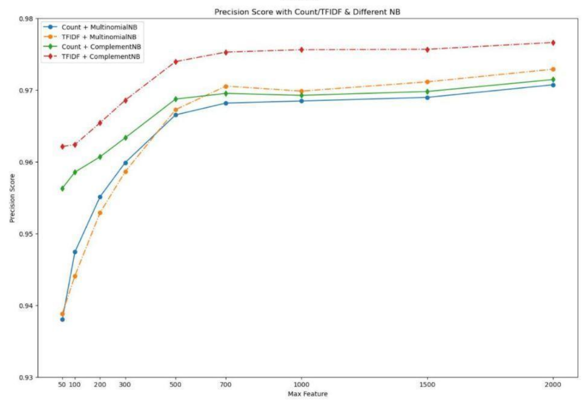
1. The raw dataset we constructed in Section 3-1.
2. Resample the data set with equal amounts of spams and hams (less data compared to raw).
3. Resample the data set so Ham:Spam=2:1.

6-1 Imbalanced Dataset: Spam > Ham

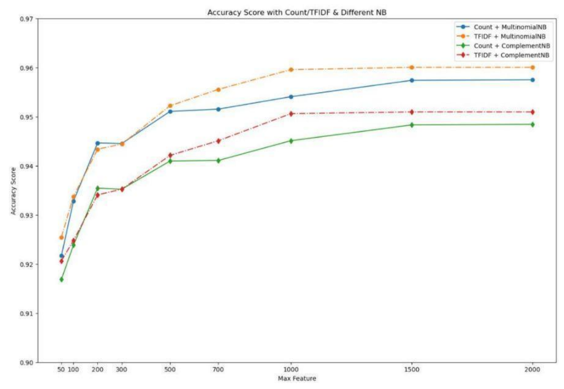


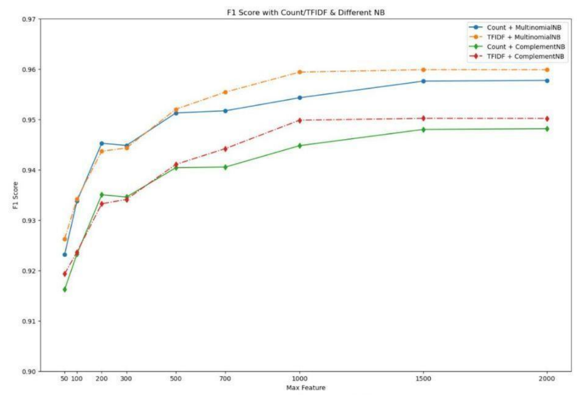


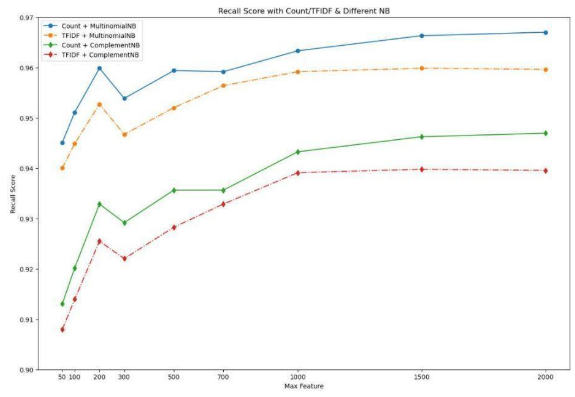


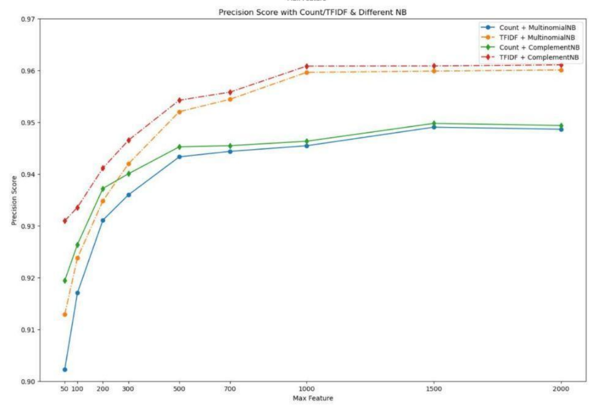


6-2 Balanced Dataset

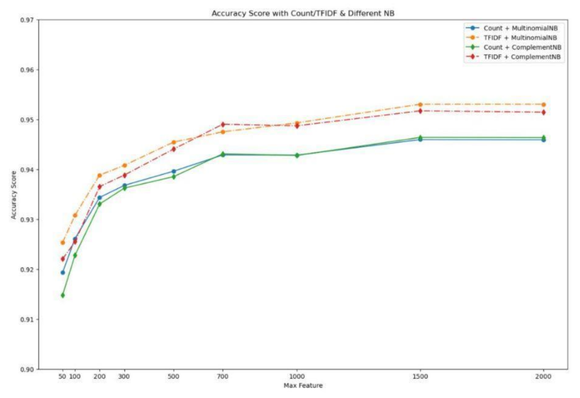


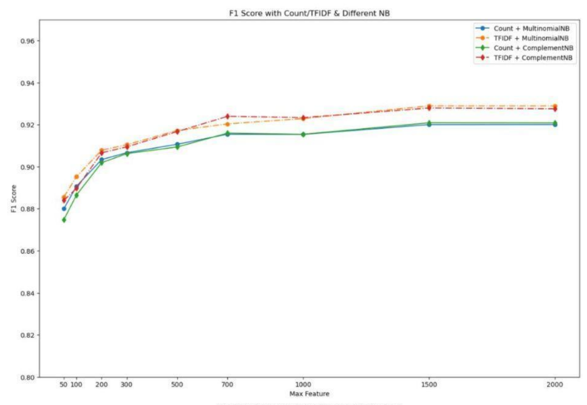


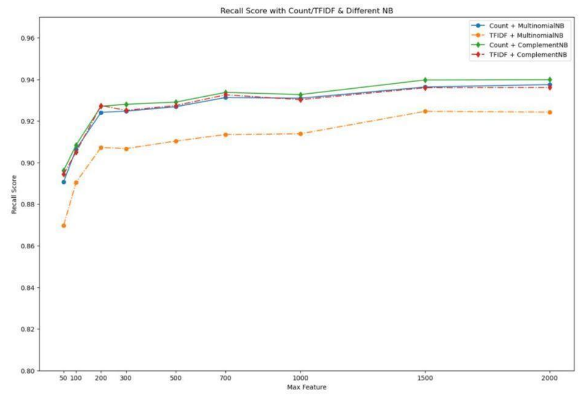


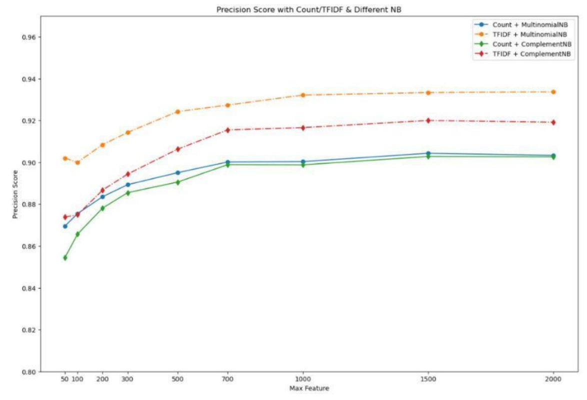


6-3 InBalanced: Ham > Spam



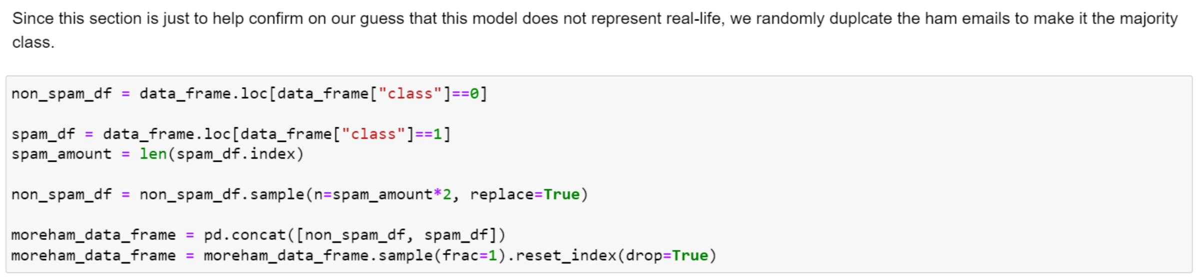






6-4 Analyze

Although the way we re-sample the raw dataset to gradually increase the proportion of ham emails will introduce great bias[image below], we can still get some rough ideas on how models behave on a dataset that has a better representation of real life.



Since this section is just to help confirm on our guess that this model does not represent real-life, we randomly duplcate the ham emails to make it the majority class.

```
non_spam_df = data_frame.loc[data_frame["class"]==0]

spam_df = data_frame.loc[data_frame["class"]==1]
spam_amount = len(spam_df.index)

non_spam_df = non_spam_df.sample(n=spam_amount*2, replace=True)

moreham_data_frame = pd.concat([non_spam_df, spam_df])
moreham_data_frame = moreham_data_frame.sample(frac=1).reset_index(drop=True)
```

For all tested model combinations, the scores remain an upward trend as the max feature amount increases. Compared to an imbalanced dataset with the majority class being spam emails, we can see that the overall performances (all 4 scores) start to drop as the ham emails increase. Vectorization with TF-IDF becomes better than Count, especially when ham emails become the majority class.

Observations when there are more spams:

1. TF-IDF + MultinomialNB has the best prediction accuracy and F1 score.
2. TF-IDF + MultinomialNB and Count + ComplementNB have the best F1 scores(overlapped).
3. TF-IDF + ComplementNB has the worst training result.
4. TF-IDF + ComplementNB has the best precision score and the lowest recall score.

The reason behind this is that, although by definition, ComplementNB should work better on an imbalanced dataset, using TF-IDF will end up with the majority of words chosen being from the majority class(spam). Thus, an extra bias got introduced in.

Observations when spam and ham have equal amounts:

1. TF-IDF + MultinomialNB has the best prediction accuracy and F1 score.
2. Judging by accuracy scores and F1 scores, vectorization with TF-IDF produces more reliable predictions.
3. Count + ComplementNB has the worst training result.
4. Count + MultinomialNB has the highest recall score and the lowest precision score. As the spam and ham amounts get balanced in the dataset, the bias on feature extraction from TF-IDF gets reduced. Observation 3 & 4 is due to the fact of the count vectorizer's inability in identifying more important and less important words for analysis.

In Complement Naive Bayes, instead of calculating the probability of an item belonging to a specific class, we calculate the probability of an item being part of all classes. With the count vectorization and TF-IDF cannot effectively determine the importance of term features, complementNB still produces insufficient results.

Observations when there are more hams:

1. The TF-IDF vectorizer produces better results than the countvectorizer.
2. Accuracy scores and F1 scores for TF-IDF + MultinomialNB and TF-IDF + ComplementNB are close.
3. TF-IDF + MultinomialNB has the highest precision score and the lowest recall score, whilst TF-IDF + ComplementNB has stable performances on both precision and recall.
4. Recall scores with count vectorization are comparatively higher.

The dataset class proportion here reflects the real situation better. Compared to the other two dataset, we have a poorer overall performance on filtering spam emails. The bias analyzed in "Observations when there are more spams” can explain this, as the bias now can mislead the spam emails detected as ham.

In general, TF-IDF + ComplementNB has more stably-accurate behaviors in detecting spams. TF-IDF feature extraction enhances the importance of words for detecting ham emails and complementNB flattens out bias due to there being far less spam emails in the dataset.

In the next sections, we will be looking into the generalization of TF-IDF + MultinomialNB and TF-IDF + MultinomialNB.

7 Experiments on Kaggle Email Spam Dataset

The dataset for this extended experiment is from Kaggle[1].

Around 25.65% of samples within it are labeled as spam emails, and the rest as ham emails. It is fair to say that this dataset is representative as in real life, ham emails would usually be the majority class.

7-1 Email Headers

Information we need to extract for potential future use is:

(1) From
(2) To
(3) CC account amount
(4) Date, e.g. Wed
(5) time, e.g. 19:30:01 (PS. dropped; too many missing values)

However, in later parameter tuning, we noticed that Date, CC, and time do not bring in sufficient improvements. We might need to research for better ways to extract and process features in this information.

7-2 Weighted Class TF-IDF

TF-IDF is selecting features based on term frequency alone and negative words are present in most of the samples. As a result, the minority class gets under-represented. Weighted class TF-IDF is to run TF-IDF for feature selection separately on each class[9].

$$f_{c_i} = \text{max\_features} \times \frac{n_{c_i}}{n_d}$$

Here, n_{c_i} is the number of samples in the i-th class, n_d is the total number of samples and f_{c_i} is the number of features to be selected for the i-th class.

Further, in order to ensure non-overlapping feature-set between both classes, features selected in previous classes are passed as stop-words to the stop_words hyperparameter when running TF-IDF over other classes while stepping through.

Weighted Class TF-IDF runs on: email contents, email “From” accounts, email “To” account

7-3 Cross Validation and Parameter Tuning

We separate the train set into a train set and validation set for cross validation. Performances are evaluated by the mean score from cross validation, including: accuracy, precision, recall and F1.

We try different combinations of features, e.h. Only header information, or “From”+email content etc.

Best Performance so far is: use the combination of content + from + to + date with Multinomial Naive Bayes with scores accuracy: 0.9762443438914027 precision: 0.9347408829174664, recall: 0.9838383838383838, f1: 0.9586614173228347.

.

7-4 Section Summary

Best performance is: the combination of content + from + to + date with Multinomial Naive Bayes with accuracy: 0.9762443438914027 precision: 0.9347408829174664, recall: 0.9838383838383838, f1: 0.9586614173228347.

When the dataset is imbalanced, TF-IDF will end up with the majority of words chosen being from the majority class; and this bias gets strengthened by class weighted (although we avoid the situation in which the minority class is not represented).

We use the terms extracted from one class samples as stop words during extraction for another class; the order to process classes can introduce bias.

What can be better:

(1) Vectorization ngram range hasn't been tested
(2) We could possibly inverse the proportion of classes when doing weighted class TF-IDF. This needs further testing.
(3) The information extracted from email headers, especially Date/Cc Account Amount, does not bring in sufficient improvements. Perhaps we should try other ways to pre-process them.

Bayes Related: This experiment uses all extracted information within one Naive Bayes classifier. Perhaps it would be better if we can separate the training and predicting to: 1. Content Based Filter; 2. Header Based Filter.

8 Summary

The goal of the experiment is to learn Naive Bayes classifiers on spam email filtering.

Spam emails are the majority class(66.67%) in the given dataset. Thus, the data is unbalanced. This bias would affect the classification performances in the later stage.

Here, vectorization based on term counts and on TF-IDF are used to transform textual data into numeric data as a vector space model. There are three Naive Bayes algorithms used from sklearn modules: Multinomial, Bernoulli, Complementary.

We tested the overall performances on a dataset with different class proportions, by combinations of different vectorizers and Bayes algorithms.

On the raw data set (majority: spam), the combination of count vectorization and Multinomial Naive Bayes performs the best, with: Accuracy score: 0.9575208422859642, Precision score: 0.970570217199754, Recall Score: 0.9653073113894554, F1 Score: 0.9679301817518213.

On a larger resampled data set (majority: ham, way to resample introduces extra bias), TF-IDF + ComplementNB behaves better. Accuracy score: 0.9516975848792439, Precision score: 0.9200046040515654, Recall Score: 0.9359484777517565, F1 Score: 0.927908056651962.

In the extended experiment, we also combine email header information into the model, with features extracted by weighted class. However, there exists lots of room for improvement. If time applies, it could be worth trying separating one classifier into 2 filters: content & header, and see how this could affect the overall performances.

9 References

[1] Email Spam Dataset (Extended)
https://www.kaggle.com/datasets/maharshipandya/email-spam-dataset-extended?resource=download

[2] Building a Spam Filter with Naive Bayes: https://amol-kulkarni.com/project/naivebayes/

[3] A Complete Guide on Feature Extraction Techniques:
https://www.analyticsvidhya.com/blog/2022/05/a-complete-guide-on-feature-extraction-techniques/#:~:text=If%20we%20have%20textual%20data,is%20also%20called%20text%20vectorization

[4] What Is The Difference Between TfidfVectorizer And CountVectorizer?
https://enjoymachinelearning.com/blog/countvectorizer-vs-tfidfvectorizer/?expand_article=1

[5] The Shapley Value for ML Models
https://towardsdatascience.com/the-shapley-value-for-ml-models-f1100bff78d1

[6] How to Check the Accuracy of Your Machine Learning Model
https://deepchecks.com/how-to-check-the-accuracy-of-your-machine-learning-model/

[7] Precision and recall — a simplified view
https://towardsdatascience.com/precision-and-recall-a-simplified-view-bc25978d81e6

[8] What is a good F1 score and how do I interpret it? https://stephenallwright.com/good-f1-score/

[9] How to correctly use TF-IDF with imbalanced data
https://www.deepwizai.com/projects/how-to-correctly-use-tf-idf-with-imbalanced-data