SPAM OR HAM

FERRY FNU & BY HOANG CHUNG

Introduction

These days, email spam doesn't seem like a big deal anymore. But that's only due to the hard work of the spam classifiers in most, if not all, email services. If you had to click and read every email in your spam folder, it would be a waste of time at the very least or in the worst case, get locked out of your device after downloading ransomware. To protect users, our project is an email spam classifier that determines whether an email is spam or ham/legitimate based on the email's body.

Originally, we aimed to create a model that could retrieve the title, author, and genre from a book cover with a combination of OCR and Computer vision. However, we could not achieve satisfactory results. Our main challenge involved parsing title and author information from the diverse book cover designs due to varying text placement and irrelevant text like NY bestseller tags, requiring us to use NLP in addition. In the end, we determined that we would not be able to achieve our initial goals due to other commitments. We switched directions, deciding to create an email spam classifier.

Our goal turned to determining the best ML algorithm to train spam classifiers at different levels of available training data.

Methodology

Our project uses the email dataset from [1], which includes 193582 labeled email (ham/spam) samples. We decided on 4 datasets of different sizes, which were sampled from the dataset in [1]. The naming of these sizes is relative to the email dataset. We will use size and variant interchangeably. For each dataset size, we trained several binary classifier models.

Sizes:	Models:
Small: ~500 training samples	Logistic Regression
Medium: ~5000 training samples	Multinomial & Bernoulli Naive
Large: ~50000 training samples	Bayes
Extreme: ~155000 training samples	Random Forest
	SVM*

We preprocessed the data with pandas, setting "spam" as 1 (positive) and "ham" as 0 (negative). For more advanced preprocessing, training, and evaluation, we used scikit-learn. We used their TfidfVectorizer to remove stop words and vectorize the email text. For training, we did hyperparameter tuning with scikit-learn's GridSearchCV to train and choose good hyperparameters for our models based on five-fold cross-validation accuracy.

Finally, we evaluated the models based on accuracy and precision. Precision is especially important as Dada, E. G., et al. [2] claims that flagging a legitimate email as spam (false positive) is much more costly than letting some spam through (false negative).

*Only trained with small and medium datasets, explanation in the discussion section.

References

1. Lilith, M. (2024, March 14). 140k+ SPAM: Ham Email dataset for Classification. Kaggle. https://www.kaggle.com/datasets/marcolilith/140k-spam-ham-email-dataset-for-classification

2. Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: Review, approaches and open research problems. Heliyon, 5(6). https://doi.org/10.1016/j.heliyon.2019.e01802

3. Naive Bayes. scikit. (n.d.-a). https://scikit-learn.org/stable/modules/naive_bayes.html

4. When to avoid random forest?. Cross Validated. (2014, August 16). https://stats.stackexchange.com/questions/112148/when-to-avoid-random-forest

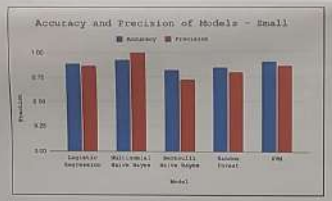
5. Sklearn.svm.SVC. scikit. (n.d.-b). https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

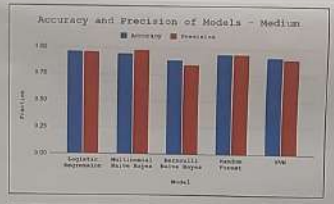
Results

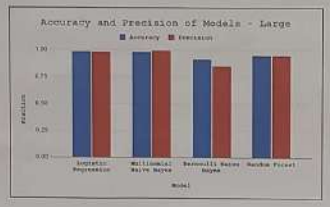
Multinomial Naive Bayes performed extremely well with the highest precision when given the small, medium, and large datasets. Its accuracy steadily rose from 92.86% to 98.44% as the training size increased from small to extreme. This rising trend of accuracy was consistent in all other models except random forest.

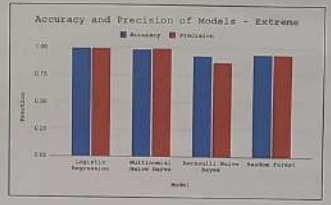
For random forest, the medium-variation accuracy and precision were 95.07% and 95.01%, respectively. These are 10.94% and 16.06% higher than the small variations accuracy (84.13%) and precision (78.95%), respectively. However, the large-variation and extreme-variation random forest models performed slightly worse in accuracy and precision (~1%) than their medium counterpart.

For all models except SVM, they saw a huge improvement (>10%) in accuracy and precision when they received a medium dataset rather than a small dataset. However, when trained on a large or extreme dataset, the models only performed marginally better than their medium variations.









Discussion

We determined that the multinomial Naive Bayes was the best model for our problem for all 4 sizes. The multinomial Naive Bayes had the highest precision for all 4 sizes. When compared to the Bernoulli Naive Bayes, the major difference in performance is likely due to their different assumptions.

The scikit-learn documentation on Naive Bayes [3] states that Bernoulli Naive Bayes assumes the data has a multivariate Bernoulli distribution, while a multinomial Naive Bayes assumes the data has a multinomial distribution. Our TF-IDF vectorized data has a more multinomial distribution than a multivariate Bernoulli distribution, so the multinomial Naive Bayes likely performed better than its Bernoulli counterpart since the data fit the former's assumptions more.

What we found surprising was how random forest didn't outperform all the other models when given the large and extreme datasets. It had a respectable accuracy of 94.19% and a precision of 94.17% with the extreme dataset but these metrics were overshadowed by both logistic regression and multinomial Naive Bayes. Random forest's lackluster performance could be due to the sparsity of the TF-IDF vectorized data. Sycorax's answer in [4] notes that random forests struggle when the data is very sparse since the limited non-zero values make it difficult to find any productive split for its decision tree children to make.

SVM performed well with a small amount of data, but not much better when given a medium amount of data. We attempted to train SVM with the large and even extreme datasets, but we found that they took too long to train. The training of the large variant of the SVM wasn't completed by 14 hours, which is when we decided to abort it. The scikit-learn documentation on SVC, the SVM implementation we used, [5] does warn that it becomes impractical when training tens of thousands of samples (The large dataset includes 50000 training samples). We determined that it was unnecessary to train SVM with large and extreme datasets since it was impractical and was outperformed by other models when trained on small and medium datasets.

Reflections

These are what we believe we could have done better:

- Wider Grid Search for random forest. Bad hyperparameters might've limited it.
- Bag of words instead of TF-IDF vectorizer for Bernoulli NB. Data should be more like a multivariate Bernoulli distribution.
- Train a deep learning model (i.e. transformer). It would've been interesting to see how its performance changes depending on the size of the dataset.
- Included confidence intervals in our metrics. This would give us more information about how reliable our numbers are, especially for the 100% precision we got for multinomial Naive Bayes - small.

Conclusion

All in all, while we were unable to achieve our initial project idea, we were able to solve what we could consider a more serious problem. We found multinomial Naive Bayes to perform the best for all 4 dataset sizes, achieving >90% accuracy and precision in all cases and even a near 100% precision rate and accuracy rate in the extreme case.