Anirudh Rao
BE21B004

DA5400 – Foundations of Machine Learning

Assignment 2

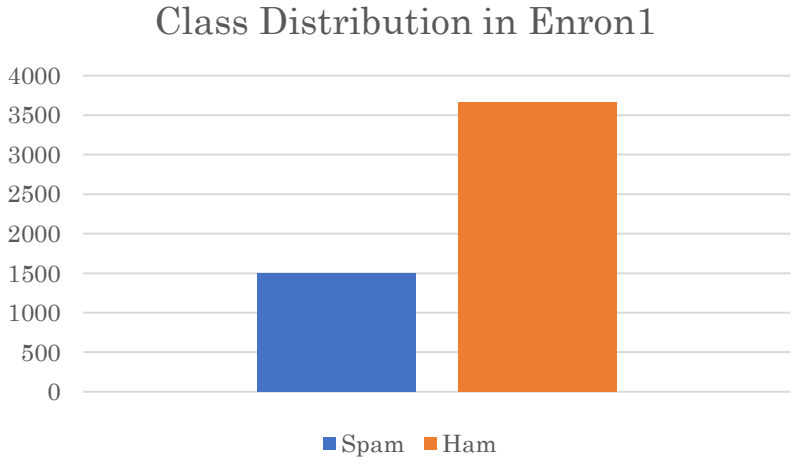
Contents

Dataset

To build the spam classifier, a subset of the Enron-Spam dataset was chosen. This dataset was curated in the conference paper "V. Metsis, I. Androutsopoulos and G. Paliouras, "Spam Filtering with Naive Bayes - Which Naive Bayes?". Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS 2006), Mountain View, CA, USA, 2006". The dataset can be found here.

Enron-Spam consists of 6 sub-datasets that contain different numbers of spam and ham emails. For this assignment, the Enron1 dataset was chosen. It contains 1500 spam emails and 3672 ham emails, with a total of 5172 emails. The class imbalance was adequately addressed during the training of the classifier.

80% of spam emails and 80% of ham emails were used for training. The remaining emails were set aside for testing the classifier.



Preprocessing and Feature Engineering

Each of the training emails was subjected to various preprocessing steps before features were extracted from them to train the classifier. We define the vocabulary as the set of all words that occur across as all emails.

The contents of all emails were made lowercase to ensure that words with the same spellings are treated the same. For example, “Subject” and “subject” will be considered to be the same.

All punctuation and special characters were removed from the emails, allowing only alphanumeric characters. This ensures that words like “emails” and “email’s” will be treated the same.

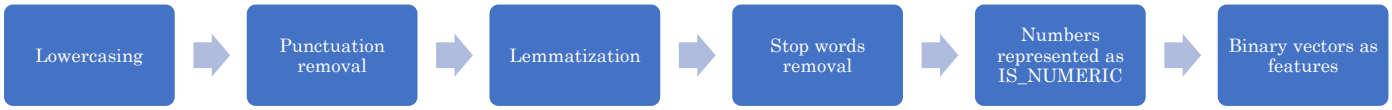
To reduce the size of the vocabulary of the emails, lemmatization was performed using Python’s `spacy` library. This step converts words to their base form, while retaining the meaning of the words. For example, the words “buy”, “bought”, and “buying” are all converted to “buy”.

Common words, known as stop words, were removed from the vocabulary. Examples of these words include “the,” “and,” “is”. This was carried out using the `nltk` library.

All numbers in the emails were represented by “IS_NUMERIC”.

Following this, the size of the vocabulary was 35355 words. To use only the informative words in the vocabulary, low frequency pruning was performed. Words that occurred in lesser than 0.1% of the emails were discarded. This reduced the size of the vocabulary to 6448.

Finally, each of the training emails was converted into a binary vector of length 6448, with the i^{th} entry being 1 if the email contained the i^{th} word in the pruned vocabulary, and 0 otherwise. These binary vectors were used as the features for the classifier model.



The binary features and labels were stored in a Pandas DataFrame. The head of this DataFrame looks like:

	IS_NUMERIC	subject	hi	paliourg	pill	everything	operate	comfort	convenience	prime	...	redir	cqo	jenuwine	berryman	ledger	alamo	quickenloan	mcneill	newland	EMAIL_TYPE
0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
1	1.0	1.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
2	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
3	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1
4	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1

5 rows × 6449 columns

Classifier

To classify the emails as spam (1) or ham (0) using the binary vectors as features, a Naïve Bayes classifier was built. 90% of the training emails were used for actually training the classifier. The remaining 10% were used as a validation set.

Each of the n training emails were represented by their binary feature vector $[f_1 \quad f_2 \quad \cdots \quad f_d]^T$ with $f_i \in \{0,1\}$ depending on whether the i^{th} word in the pruned vocabulary was present in the email or not.

The classifier had the following parameters that were learned during its training – $\hat{p}$, $\{\hat{p}_1^1, \hat{p}_2^1, \ldots, \hat{p}_d^1\}, \{\hat{p}_1^0, \hat{p}_2^0, \ldots, \hat{p}_d^0\}$, where d = size of vocabulary. These parameters were defined as:

$$\hat{p} = \frac{\sum_{i=1}^{n} y_i}{n} = \text{Fraction of points with label } '1'$$

$$\tilde{p}_k^y = \sum_{i=1}^{n} \mathbb{1}\left(f_k^i = 1, y_i = y\right) + \alpha \approx \text{Number of points with label } 'y' \text{that have } f_k = 1$$

$$\hat{p}_k^y = \frac{\tilde{p}_k^y}{\sum_{k=1}^{d} \tilde{p}_k^y}$$

Here, α is the Laplace smoothing factor, a hyperparameter of the classifier. It helps prevent zero probabilities from occurring during test prediction.

The training set was used to learn the parameters. For this a `fit()` function was defined.

Given $x_{\text{test}} \in \{0,1\}^d$, we predict the label as '1' if $P(y_{\text{test}} = 1 \mid x_{\text{test}}) > P(y_{\text{test}} = 0 \mid x_{\text{test}})$ and '0' otherwise. Using Bayes rule,

$$P(y_{\text{test}} = 1 \mid x_{\text{test}}) = \frac{P(x_{\text{test}} \mid y_{\text{test}} = 1)}{P(x_{\text{test}})} P(y_{\text{test}} = 1)$$

$$P(y_{\text{test}} = 0 \mid x_{\text{test}}) = \frac{P(x_{\text{test}} \mid y_{\text{test}} = 0)}{P(x_{\text{test}})} P(y_{\text{test}} = 0)$$

The denominator is $P(x_{\text{test}})$ in both cases and can be ignored.

Suppose $x_{\text{test}} = [f_1 \quad f_2 \quad \cdots \quad f_d]^T$ with $f_i \in \{0,1\}$. Then,

$$P(y_{\text{test}} = 1 \mid x_{\text{test}}) \propto \left(\prod_{k=1}^{d} (\hat{p}_k^1)^{f_k} (1 - \hat{p}_k^1)^{(1-f_k)} \right) \cdot \hat{p}$$

$$P(y_{\text{test}} = 0 \mid x_{\text{test}}) \propto \left(\prod_{k=1}^{d} (\hat{p}_k^0)^{f_k} (1 - \hat{p}_k^0)^{(1-f_k)} \right) \cdot (1 - \hat{p})$$

We predict the label as '1' if $P(y_{\text{test}} = 1 \mid x_{\text{test}}) \geq P(y_{\text{test}} = 0 \mid x_{\text{test}})$ and '0' otherwise.

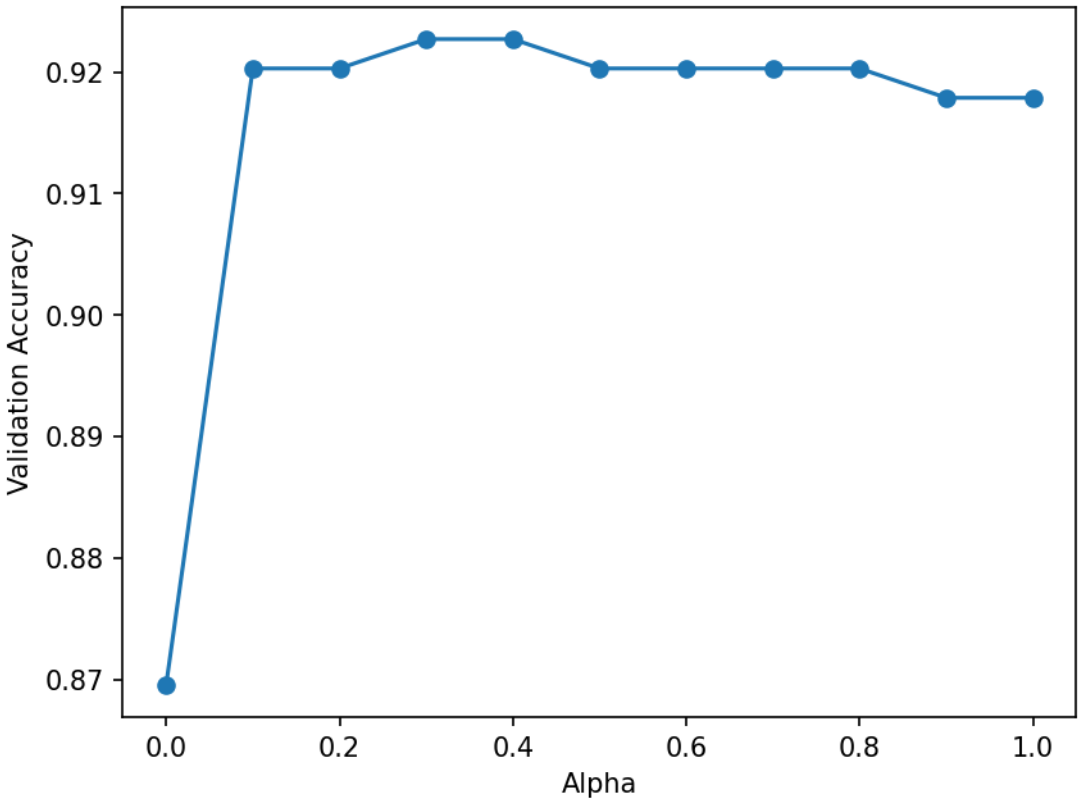
For predicting directly from a binary vector representation, the `predict()` function was defined. To predict from an unprocessed email present in a folder, the `predict_from_path()` function was defined.

Hyperparameter Tuning

To determine the best value of α, different values were used during the training of the classifier. Its performance, on the basis of accuracy, was assessed on the validation set.

It was found that the best value of α to use was 0.3. This achieved a validation accuracy of 92.3%.

α	Validation Accuracy
0	0.869565
0.1	0.92029
0.2	0.92029
0.3	0.922705
0.4	0.922705
0.5	0.92029
0.6	0.92029
0.7	0.92029
0.8	0.92029
0.9	0.917874
1	0.917874



The classifier with $\alpha = 0.3$ had $\hat{p} = 0.29$.

The words with the highest probabilities in the two classes are shown below:

Spam	
Word	**Probability**
subject	0.016563996
IS_NUMERIC	0.012534431
http	0.005386093
get	0.005123895
com	0.004820298
good	0.003523109
please	0.003357511
time	0.003329911
price	0.003247112
e	0.003109113

Ham	
Word	**Probability**
subject	0.020708048
IS_NUMERIC	0.019234593
please	0.008849901
thank	0.008250648
enron	0.008201298
cc	0.007207244
gas	0.00631189
hpl	0.00626254
forward	0.006051039
deal	0.005959389

Test Performance

A Naïve Bayes classifier with $\alpha = 0.1$ was then trained on the entirety of the training data. It was then tested against the 20% of emails left as a test set. Using the `predict_from_path()` function, these emails were directly read by the classifier from their directory locations, processed, converted into binary representation, and predicted as spam (1) or ham (0).

The accuracy of the classifier on the test set was found to be 93.4%, indicating good classification performance.

The confusion matrix for the classifier's prediction on the test set is shown below:

		Predicted	
		Spam	**Ham**
Actual	**Spam**	246	14
	Ham	54	721