Department of Computer Science and Engineering,
Indian Institute Of Technology Madras

CS5691 : Pattern Recognition and Machine Learning

Assignment 3

Spam Email Classifier

Pranit Zope
AE20B046

May 5, 2024

Contents

1 Introduction

The main aim is to build an email classifier where the training data is chosen/created, and then a model is trained. We will dive into the details later. The ultimate fucntioning is wehn given a folder of emails, the model should predict emails with labels +1 for spam and 0 for non-spam, which will be reffered to as "ham" in the entire assignment.

2 Overview of the training Dataset

2.1 Aggregation of Data from different sources

The dataset used for training is a combination of 4 different datasets all taken form kaggle A and is built as follows

Dataset	Total Emails	Ham Emails		Spam	
		Total	Used	Total	Used
Error-Spam-Subset	10000	5000	1000	5000	1567
Spam Assasin	6046	4150	1000	1896	1000
Spam-Ham	5728	4360	1000	1368	1000
Ling-Spam	2605	2172	1000	433	433
Total			4000		4000

Table 1: Overview of the Dataset

This is then converted into a pandas dataframe which looks like this

	Body	Label
976	\r\nOn Mon, 09 Sep 2002 12:05:55 PDT,\r\nRick ...	0
7677	Subject: entrust your visual identity to us t...	1
7118	Subject: notification from sky bank # 6521 - 3...	1
367	empty	0
2245	Subject: discourse conference final call\r\n\r...	0
...	...	...
2858	Subject: sum : ref . on formal models of disco...	0
5956	Subject: new pharm site new great prices saul\...	1
784	On Wed, 2002-09-25 at 13:34, bitbitch@magnesiu...	0
4713	Subject: it is your chance to reduce spending ...	1
1818	\r\nDear friend\r\n\r\nÂ¡Â¡\r\nÂ¡Â¡Â¡Â¡<!--\r\...	1

8000 rows × 2 columns

Figure 1: Initial Dataframe

2.2 Data Preprocessing

Given the complexity of the emails in general, which include various numbers, characters, and formats, we require a standardized format to train our model effectively. We've devised a 14-step preprocessing procedure to convert the data into the desired format.

1. **Lowercasing:** Convert all text to lowercase for equivalence, e.g., "Super" becomes "super".

2. **Contractions Fix:** Expand contractions, e.g., "wasn't" becomes "was not", "don't" becomes "do not".

3. **Removing Numbers:** Replace numerical values with white spaces, e.g., "12" becomes " ".

4. **Replacing URLs:** Replace URLs with a generic term, e.g., "https:www.example.com" becomes "links".

5. **Removing Mail IDs:** Remove email addresses, e.g., "abcd@example.com" becomes "".

6. **Replacing Currency Signs:** Replace currency symbols with a generic term, e.g., "$" becomes "ruppees".

7. **Removing Punctuations:** Remove all punctuation marks, e.g., ".(_ :;/" becomes "".

8. **Replacing Accented Characters:** Convert accented characters to ASCII equivalents, e.g., "ã" becomes "a", "ï" becomes "i".

9. **Removing Multiple Occurrences:** Reduce multiple occurrences of a character in a string, e.g., "arrrrrrrrrrr" becomes "arr", "gggggggggggggghhhhhhhhhh" becomes "gghh".

10. **Remove Common Words:** Remove common words like "subject" and "enron", e.g., "subject" becomes "".

11. **Removing Extra Whitespace:** Remove extra whitespace and newline characters, e.g., newline becomes " ".

12. **Lemmatization and Tokenization:** Convert words to their base form and tokenize them, e.g., "bats" becomes "bat".

13. **Stemming:** Reduce words to their root form, e.g., "programmer" becomes "program".

14. **Removing Stopwords:** Remove common stopwords like "a", "the", "and", etc.

We also can make a wordcloud in order to visualize the words (processed tokens) that are present in the data. The wordcloud generated looks like this

Figure 2: Spam Word Cloud

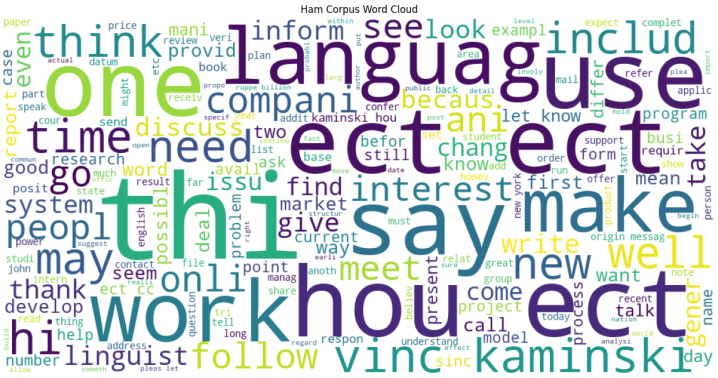

Figure 3: Ham Word Cloud

3 Models Considered

We have considered 3 models for training the dataset, which are Naive-Bayes, Logistic regression and Support vector machine. The procedure for training for each of the models is as follows.

3.1 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks. It's particularly effective in high-dimensional spaces and when the number of features (dimensions) exceeds the number of samples. SVM works by finding the hyperplane that best separates different classes in the feature space. That is the sole reason it is suitable for building this email classifier.

Since the implementation of SVM is quite complex (and as mentioned in the questionnaire, we won't write SVM from scratch but will use the library `sklearn.svm`.

3.2 Naive Bayes Classification

Naive Bayes is a probabilistic machine learning algorithm based on Bayes' theorem. It's particularly popular for text classification tasks like spam email classification. It can

be used for email spam classification due to its efficiency and implementation simplicity. Here is how the Naive Bayes classifier works.

$$P\left(y \mid x_1, \ldots, x_n\right) = \frac{P\left(x_1 \mid y\right) P\left(x_2 \mid y\right) \ldots P\left(x_n \mid y\right) P(y)}{P\left(x_1\right) P\left(x_2\right) \ldots P\left(x_n\right)}$$

which can be expressed as:

$$P\left(y \mid x_1, \ldots, x_n\right) = \frac{P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right)}{P\left(x_1\right) P\left(x_2\right) \ldots P\left(x_n\right)}$$

Now, as the denominator remains c

$$P\left(y \mid x_1, \ldots, x_n\right) \propto P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right)$$

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^{n} P\left(x_i \mid y\right)$$

(Here $P(x)$ is the probability in bernoulli distribution.

3.3 Logistic Regression

Logistic Regression is a statistical method used for binary (or multiclass) classification tasks.It is very interpretable and flexible to use and works well in practise when the data has a clear decision boundary. The optimization problem for Logistic Regression is as follows.

$$\max_w \mathcal{L}(b, w) = \prod_{i=1}^{n} \sigma\left(x_i\right)^y \left(1 - \sigma\left(x_i\right)\right)^{1-y_i}$$

where

$$\sigma(x) = \frac{1}{1 + e^x}$$

and thus we need to solve for w

4 Workflow for Training our models

As discussed earlier, we process our raw data and then make a pandas dataframe out of it. Here is a demonstation of how he process is carried out.

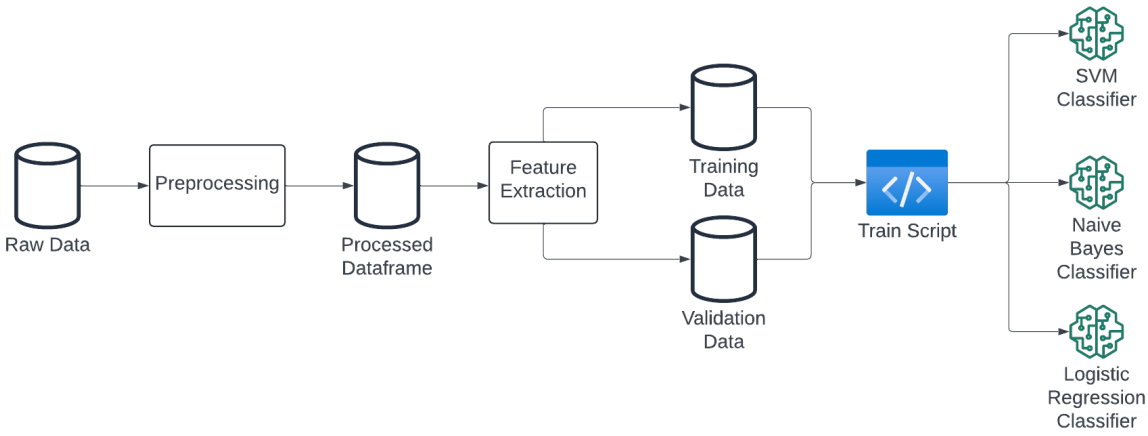

Figure 4: Process for training our Models

A few key parts of the process are as follows

- The features are extracted using a TF-IDF vectorizer which is used to convert the textual content of emails into numerical vectors. It takes care of things like the term frequency (importance of a word in the emai) and inverse document frequency, which is how unique a point is in the entire dataset.
- The data is split into a 80:20 ratio for cross validation where the 80% of data is used for training and the other part is used for validation if the model fits appropriately.
- The models generated will be used for building our final classifier later on.

5 Training

Here we will discuss the parameters used to check the training and the implementation and outcomes of the training of all 3 models.

5.1 Parameters Used

1. **Accuracy** is a measure of how many labels are predicted correctly. This includes both positives and negatives.
2. **Confusion Matrix** is a matrix formed by True Positives and negatives and False positives and negatives in the following way.

$$\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$$

3. **Precision** is the ratio of True positives to the predicted positives

$$\text{Precision} = \frac{TP}{TP + FP}$$

4. **Recall** is the ratio of True positives to the actual positives

$$\text{Recall} = \frac{TP}{TP + FP}$$

5.2 SVM Training

We used linear and rbf kernels with varying regularization constant and the result for the most optimal one on the training data is $c = 3.d28$ and the RBF Kernel. The other stats are shown below.

```
Model: SVM
        Regularization Constant:  3.2857142857142856
        Kernel:  rbf
Confusion Matrix:
 [[779  13]
 [ 21 787]]
Accuracy        :        0.97875
Precision       :        0.9788
Recall          :        0.97875
```

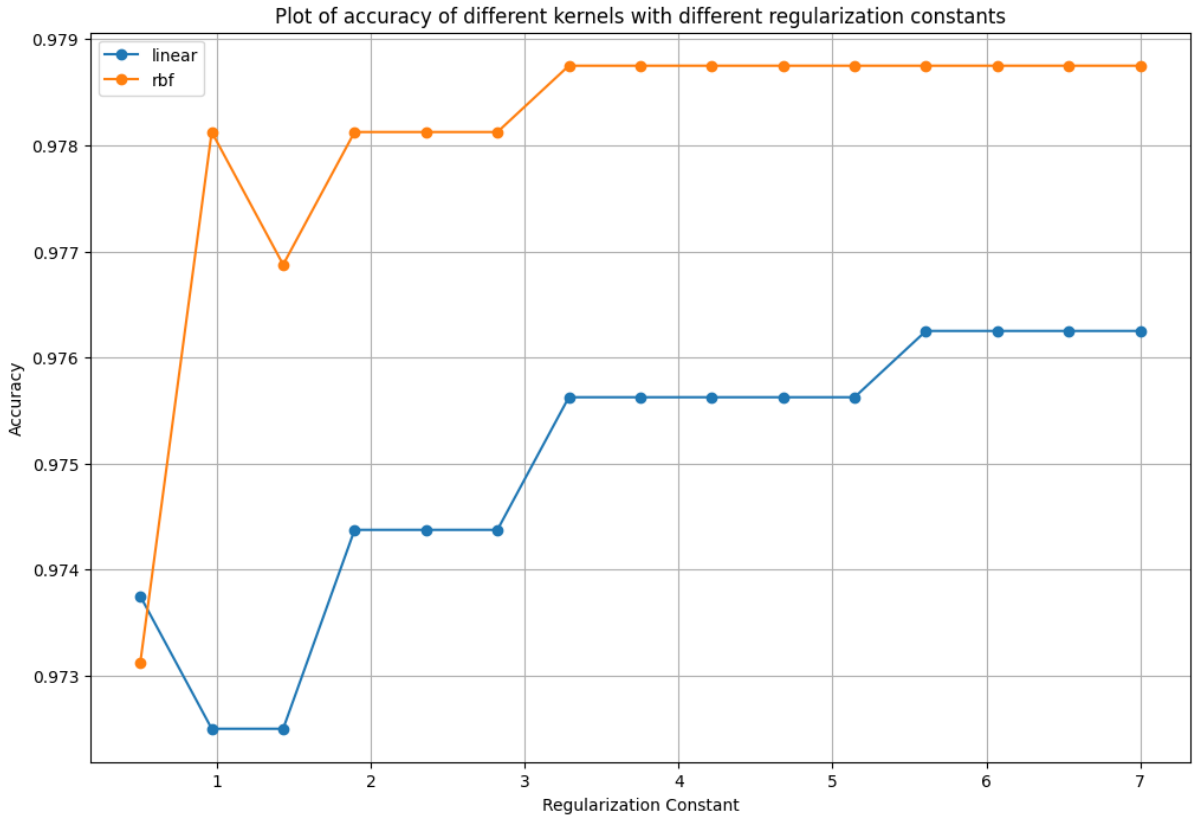

Figure 5: Accuracy of Different Kernels and regularizers

5.3 Naive Bayes Classifier

The model was trained and here are the results on both training data (recalculted to check for overfiting) and testing data.

```
Naive Bayes Model:

Training Accuracy: 0.98375
Confusion Matrix:
 [[3137   41]
 [  63 3159]]
Precision       :        0.9837736328125
```

```
Recall          :       0.98375

Testing Accuracy: 0.97
Confusion Matrix:
 [[771  19]
 [ 29 781]]
Precision       :       0.970078125
Recall          :       0.97
```

5.4 Logistic Regression Classifier

The model was trained and here are the results on both training data (recalculted to check for overfiting) and testing data.

```
Logistic Regression Model:

Training Accuracy: 0.9309375
Confusion Matrix:
 [[2941  183]
 [ 259 3017]]
Precision       :       0.93121953125
Recall          :       0.9309375

Testing Accuracy: 0.933125
Confusion Matrix:
 [[736  43]
 [ 64 757]]
Precision       :       0.93346953125
Recall          :       0.933125
```

6 Building the final classifier

We have several input emails and we need to give them labels 1 and 0 for spam and ham respectively. We will proceed in the following way

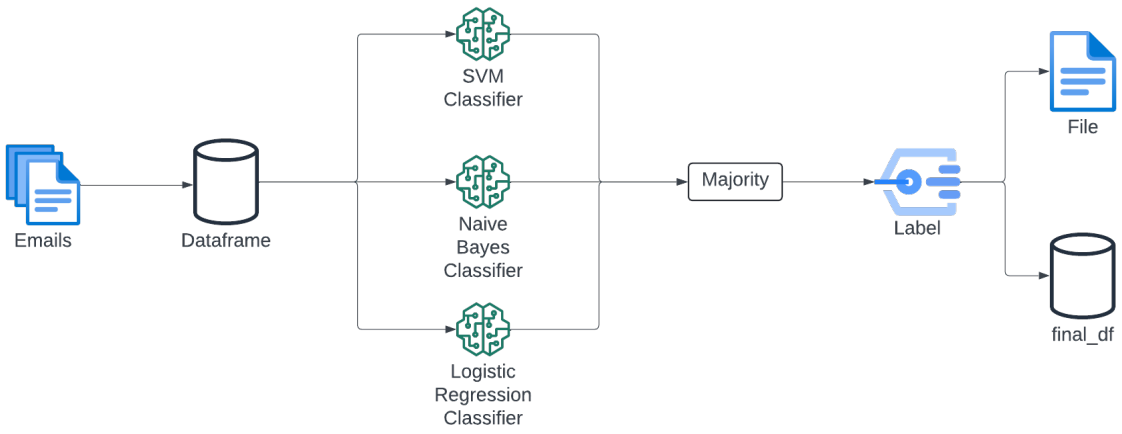

Figure 6: Algorithm for Final Classifier

- Firstly, preprocess the data according to the 14 steps that we followed earlier.

- Extract the features using vectorizer
- Predict the values using all the three trianed models and then to avoid any mis-predictions by any model, we will stack the prediction and take an index wise majority of the prediction to enhance the accuracy of the model.

7 Testing

- To test the model using the code, ensure there is a folder `test` with files name as `email#.txt` in sequence.
- There will be two kinds of outputs, one in a text form and one in a csv form.
- I tested the model on a sample set of 1106 different emails and the results look like this in both formats,

	File	Prediction
0	email0.txt	1
1	email1.txt	1
2	email2.txt	1
3	email3.txt	0
4	email4.txt	0
...	...	...
1099	email1099.txt	1
1100	email1100.txt	0
1101	email1101.txt	0
1102	email1102.txt	0
1103	email1103.txt	1

1104 rows × 2 columns

Figure 7: Result in CSV Format

```
email0.txt      1 (spam)
email1.txt      1 (spam)
email2.txt      1 (spam)
email3.txt      0 (not spam)
email4.txt      0 (not spam)
email5.txt      1 (spam)
.
.
email1097.txt   1 (spam)
email1098.txt   1 (spam)
email1099.txt   1 (spam)
email1100.txt   0 (not spam)
email1101.txt   0 (not spam)
email1102.txt   1 (spam)
```

A Links to Datasets

- Spam-Assassin : www.kaggle.com/datasets/ganiyuolalekan/spam-assassin-email-classif
- Ling-Spam : www.kaggle.com/datasets/mandygu/lingspam-dataset
- Error-Spam-Subset : www.kaggle.com/code/sharmoul/enron-spam-subset-dataset-eda
- Spam-Ham : www.kaggle.com/datasets/bagavathypriya/spam-ham-dataset