# DA5400: Foundation of Machine Learning

# Assignment 2

Nandhakishore C S (DA24M011)

November 3, 2024

## 1 Details regarding the files in submitted *.zip* file

The given submission **Solutions_DA24M011.zip** file contains three main types of files:

- A **Text file** with *.txt* extension, which contains my **details**.
- **Python script** files with ***.py*** extension, where all the **source code** is written. The classes for Data loading, Feature Extraction, Naive Bayes, Logistic Regression and it's corresponding helper functions are written in separate files.
- A **Comma Separated Values** file with **textit(.csv)** extension which contains emails and label is there. When the main python script is invoked, it creates two more *.csv* files - Test and Training files.
- A **PDF file** named ***'Report.pdf'***(i.e.) this file, with the explanation for the solutions for the above question. The latex file for the report can be found here. The end report has other rough work which was done during implementing algorithms for the assignment,
- A directory named 'test' which contains the text files named as *email_1.txt*, *email_2.txt*, ... for testing.
- **To run** the python scripts, keep the ***naive_bayes.py, linear_model.py, preprocessing.py, classification_metrics.py***, files in the same directory and run *python3 main.py* in terminal to see the results.

## 2 Question

In this assignment, you will build a spam classifier from scratch. No training data will be provided. You are free to use whatever training data that is publicly available/does not have any copyright restrictions (You can build your own training data as well if you think that is useful). You are free to extract features as you think will be appropriate for this problem. The final code you submit should have a function/procedure which when invoked will be able to automatically read a set a emails from a folder titled test in the current directory. Each file in this folder will be a test email and will be named 'email.txt' ('email1.txt', 'email2.txt', etc). For each of these emails, the classifier should predict +1 (spam) or 0 (non Spam). You are free to use which ever algorithm learnt in the course to build a classifier (or even use more than one). The algorithms (except SVM) need to be coded from scratch. Your report should clearly detail information relating to the data-set chosen, the features extracted and the exact algorithm/procedure used for training including hyper parameter tuning/kernel selection if any. The performance of the algorithm will be based on the accuracy on the test set.

### Solution

### Exploratory Data Analytics for Enron Dataset

The dataset contains five columns: *Message ID* , *Subject* , *Message* , *Spam/Ham* and *Date.* There are 29780 data points in total stored in a pandas data frame. As columns Message ID, Date and Subject are not required for classifying an email as Spam or not spam (Ham), they are dropped out of the data frame. There are some missing values in the dataset, which are dropped to get clean data for the classification model. After removing the duplicates and missing values, there are 15793 emails labeled as Ham (not spam) and 13987 emails labeled as Spam. It is evident that, the distribution of labels in the dataset are not skewed and nearly balanced. The Messages in the dataset are in English, and have standard separators (punctuations, conjunctions and accent words) in English.

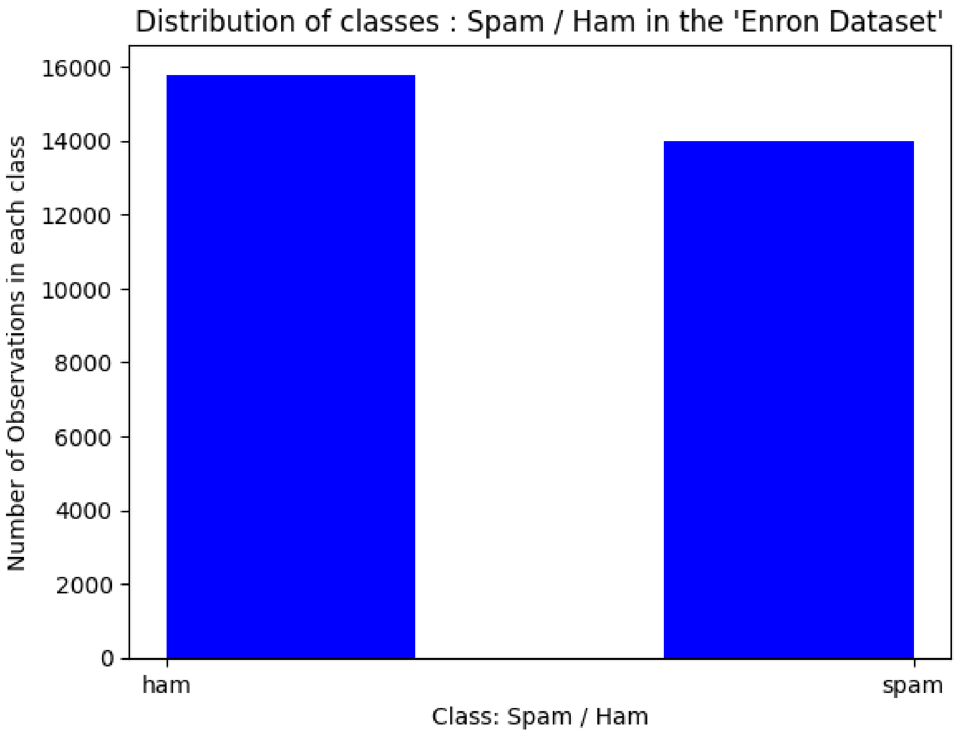

Figure 1: Distribution of data points (emails) in Enron dataset with label Ham and Spam

## Objective:

Given a text from an email, build a classifier which should predict +1 if the email is Spam (or) predict 0 if the the email is Not Spam / Ham.

## Feature Extraction

The Enron dataset comprises of emails from various organizations which are stored as python string literals labeled as Spam / Ham. If we can create a python dictionary mapping all the words in the English language as keys and the values in the dataset as values to the keys, the resulting representation of the dataset is very large and requires a large number of parameters to be trained. To overcome this issue, the text is stored is collected and stored in unordered form known as the 'Bag of words model'. TF-IDF representation is one type of bag of words model, where a text data is represented using Term Frequency (TF) and Inverse Document Frequency (IDF). The common terminologies used in TF-IDF representation is given as follows:

1. **Term Frequency (TF)**: Term frequency is defined as the ratio of the number of times a term occurs in a document to the total number of terms in that document. This is the measure of finding how a word is frequenct in a document.

$$TF(word, document) = \frac{The\ number\ of\ times\ the\ word\ appears\ in\ a\ document}{Total\ number\ of\ words\ in\ a\ document}$$

2. **Inverse Document Frequency (IDF)**: Inverse Document Frequency is defined as the logarithm of the ratio of the total number of documents to the number of documents containing the term. This is a measure of rarity of a given term in the document it is taken from. Goal of calculating IDF is to penalize words which occur in all the documents in a corpus. (A corpus is a collection of documents.)

$$IDF(word, document) = log(\frac{Total\ number\ of\ documents\ in\ a\ corpus}{Total\ number\ of\ documents\ which\ contains\ the\ word})$$

3. **TF-IDF score**: For every word in the document, the TF-IDF score is calculated and the a are stored in a matrix. In general, this matrix is sparse in nature, as not every word in the English dictionary can be seen in a document.

$$TF - IDF(word, document) = TF(word, document) \times IDF(word, document)$$

In implementation the vocabulary of the English language is taken from scikit-learn's pre-processing module. Words which are very frequent and not important enough (e.g.) articles, prepositions, auxiliary verbs are also removed from the dataset. As punctuation does not any significance in TF-IDF representation, they are also removed from the dataset. Before converting text data into TF-IDF format, the words are converted into lower case alphabets as when prediction is done, it doesnot matter weather a spam email is in higher case or lower case.

For this assignment, the TF-IDF vectorising process is implemented from scratch. To remove stop words, scikit-learn's existing english stop word module is used. The punctuation marks are removed using regular expression libraries. Also, the labels Spam and Ham are relabeled as '0' and '1' for the ease of computation. Please refer *preprocessing.py* to see the exact implementation of TF-IDF vectoriser and Label encoder programs.

## Classifier

To solve the spam / ham classification problem using Enron datast, the following algorithms are implemented from scratch.

1. Naive Bayes Classifier
2. Logistic Regression

Apart from the algorithms implemented from scratch, a Support vector classifier built using scikit-learn's Support Vector Machine module using 'linear' and 'rbf' kernel are also done.

## The Naive Bayes Classifier

- Naive Bayes algorithm follows the generative modelling paragigm
- Naive Bayes is built on the following assumption: Given the class label, the value of any feature is independent of the value of any other feature. This is formally defined using 'the Class Conditional function'.

Mathematical formulation of Naive Bayes Classifier is as follows:

Let $X$ denote $y$ denote the feature matrix and the label vector from the dataset. Once TF-IDF vectorisation is done, the transformed TF-IDF matrix will have values zero or more than zero.

$$P(X, y) = P(X|y)P(y)$$

$$where\ y_i \in \{0, 1\}\ \forall i\ and\ X_i \in \mathcal{R}^{29779 \times 155782}$$

Note that, the feature matrix X is sparse in nature. When the data is generated, they are separated into two classes 'Spam' and 'Ham' labeled as '1' and '0'. These features are conditionally independent given the label assigned to it. Let $f$ be the features in the feature matrix $X$, and $p$ be the probability of the class being Spam.

$$P(y = 1) = p$$

This is modeled as a Bernoulli trail. The parameter estimate of the getting heads on a Bernoulli trail is calculated as follows:

$$P(X = [[f_{11}, f_{12}, f_{13}, ..., f_{1m]}]]|y) = \prod_{k=1}^{m} (p_k^y)^{f_k}(1 - p_k^y)^{1-f_k}$$

As we have a parameter which determines the class conditional independence and the $'m'$ parameters for each entry in class, we have to estimate $2m + 1$ parameters modeled as Bernoulli trails.

$$\hat{p} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\hat{p_j^y} = \frac{\sum_{i=1}^{n} \mathbb{1}(f_j^i = 1; y_i = y)}{\sum_{i=1}^{n} \mathbb{1}(y_i = y)}$$

The prediction is done when the following is true (from Bayes Rule)

For a given test vector $X_{test}$, the prediction is done by the following rule:

$$Predict\ 1\ if\ P(y_{test} = 1|X_{test}) > P(y_{test} = 0|X_{test})$$

Refer to *naive_bayes.py* for the implementation.

### Logistic Regression

Given the dataset $\mathbf{D}(X, y)$, the label vector $y$ is modeled using a simple linear classifier. Logistic regression uses the logistic function (also known as the sigmoid function) to map predicted values to probabilities.

$$\sigma(z) = \frac{1}{1 - e^{-z}}$$

Each data-point's label is predicted using the outcome of a Bernoulli trail. The objective function is defined as follows:

$$\mathbb{L}(w, D) = \prod_{i=1}^{n} (\sigma(w^T x_i))^{y_i} (1 - \sigma(w^T x_i)^{1-y_i}$$

$$log(\mathbb{L}(w, D)) = \sum_{i=1}^{n} y_i log(\sigma(w^T x_i)) + (1 - y_i) log(1 - \sigma(w^T x_i)$$

The above function is minimized using gradient descent method. Refer to *linear_model.py* for the implementation.

### Results

The models implemented from scratch (Naive Bayes Classifier, Logistic Regression) along with Support Vector Machines (SVM) using Linear & RBF kernel are ranked based on the accuracy of the prediction. (Refer: Table 1)

By the comparison of values, A support vector machine with rbf kernel would be a good classifier to do spam classification.

| S.No. | Classifier | Accuracy |
|---|---|---|
| 1 | Naive Bayes | 0.9884 |
| 2 | Logistic Regression | 0.9502 |
| 3 | Support Vector Machine (Linear Kernel) | 0.9952 |
| 4 | Support Vector Machine (RBF Kernel) | 0.9981 |

Table 1: Table 1: Results from various spam classifiers

## 3 References

i All the equations and mathematical concepts for Naive Bayes, Logistic Regression and related concepts are referred from the book: *Bishop, Christopher M. Pattern Recognition and Machine Learning. New York :Springer, 2006.*

ii Programming Language used: *Python3 - Van Rossum, G. Drake, F.L., 2009. Python 3 Reference Manual, Scotts Valley, CA: CreateSpace.*

iii Python Libraries used:

- *Numpy - Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2*
- *Pandas - McKinney, W. others, 2010. Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference. pp. 51–56.*
- *Matplotlib - J. D. Hunter, "Matplotlib: A 2D Graphics Environment", Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007*

- *Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.*

iv The dataset used in this assignment is taken from the publication: 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 was last maintained by Dr William W. Cohen and the CMU school of Computer Science, Carnegie Mellon University in 2004.
- The dataset is actually organized in directories with emails in different categories with labels as Ham / Spam. The data loader file to get the *.csv* file can be found here: Enron Spam Dataset.