Ministry of Higher Education and Research
Higher School of Computer Science 08 May 1945 - Sidi Bel Abbes

Second Year Second Cycle - Artificial Intelligence and Data Science

Project Report

EMAIL SPAM CLASSIFICATION USING SEMI-SUPERVISED LEARNING AND NATURAL LANGUAGE PROCESSING TECHNIQUES

Students:

- Abdelnour FELLAH
- Abderrahmene BENOUNENE
- Adel Abdelkader MOKADEM
- Yacine Lazreg BENYAMINA

Supervisor:

Dr. Nassima DIF

Release date: January 14, 2024

Table of contents

List of Figures

List of Tables

1 Introduction

Text classification problems are among the most extensively addressed and widely applied topics in real-life applications, such as spam detection, sentiment analysis, topic modeling in social media, and document classification. This is especially true with the recent breakthroughs in natural language processing techniques, which have significantly eased the handling of text data by machine learning algorithms. This would have been much more challenging without NLP techniques due to the unstructured nature of text data.

The process of building a machine learning model capable of robustly classifying text documents begins with gathering raw text data. Each data point in such a dataset is referred to as a document. Subsequently, this raw data undergoes a preprocessing step, which aims to remove symbols and words that do not contribute to the context of the text. Because machine learning algorithms cannot directly handle unstructured data, such as text, a feature extraction step is necessary. Once the raw documents are transformed into structured, preprocessed data, the problem can be approached like any other machine learning problem.

Since text data differs from the usual tabular structured data, it introduces its own challenges. One of these challenges is the difficulty of labeling different data points, not only because text classification datasets are typically large, and classifying text cannot be done automatically – it requires examination by human experts. but also because, text can be interpreted differently from one human being to another, making labeling ambiguous documents a challenging task. This is why semi-supervised techniques are often applied in document classification tasks, where only a portion of the dataset can be labeled, and ML-based techniques are employed to label the unlabeled data.

In this project we explore natural language processing and semi-supervised learning techniques to build a model that robustly distinguish spam and non spam emails,we primarily tried three different semi-supervised algorithms : Self-Learning,Label spreading and label propagation,full project available on Github via the link :
`https://github.com/Devnetly/email-spam-classification`.

2 Natural Language Processing

Natural language processing (NLP) is a subfield of Artificial Intelligence and linguistics (the scientific study of language). It is concerned with giving machines the ability to understand, interpret, and interact with human language text. NLP involves a range of tasks, including language understanding, sentiment analysis, machine translation, and text generation. This interdisciplinary field combines techniques from computer science and linguistics to enable machines to comprehend and respond to natural language in a way that is both meaningful and contextually appropriate.

2.1 Text preprocessing in NLP

Text preprocessing is an essential in solving any NLP based problem,it aims in removing unnecessary symbols and words that do not contributes in the context of the documents and removing the syntactic differences between the different words in the text by applying a sequence of steps such as : tokenization, stop words removal, capitalization...etc.

2.1.1 Tokenization

Tokenization is a preprocessing technique that breaks a piece of text into a sequence of smaller elements (usually words or sentences) called tokens. For example, consider the following sentence:

> ”Embark on the journey of self-discovery with an open heart and a curious mind.”

After applying tokenization, it yields the following result:

> {”Embark”, ”on”, ”the”, ”journey”, ”of”, ”self-discovery”, ”with”, ”an”, ”open”, ”heart”, ”and”, ”a”, ”curious”, ”mind”}

2.1.2 Stop words removal

Documents are typically filled with words that do not provide additional information about the topic, such as pronouns, commonly used verbs, conjunctions, and many others. These words are usually removed in text classification tasks.

2.1.3 Capitalization

Usually, there’s no difference between words in capital letters or in small letters. To remove this syntactic difference between words with different capitalization, the entire text is transformed to lowercase.

2.1.4 Noise Removal

Documents contain many unnecessary characters, such as punctuation, digits, and special characters. While these elements are important for human understanding and other NLP tasks, such as summary generation, they are typically removed in text classification since they don't carry any meaning.

In some cases,depending on the problem this noise can include URLs in the case of email spam detection or web pages classification or mentions in case of classifying social media posts.

2.1.5 Stemming

In a natural language, a word can appear in different forms. For example, a word can be singular or plural, a verb can be conjugated with different pronouns, or a word can have various suffixes. Stemming is a preprocessing technique that reduces a word to its root (also known as a stem). For example:

Original Word: running

Stem: run

Original Word: jumping

Stem: jump

Original Word: better

Stem: better

2.2 Feature Extraction in NLP

In NLP, feature extraction is the process of converting text data into a structured format that can be processed by machine learning algorithms. Many feature extraction techniques exist, and here we will discuss tow of those techniques which is bag of words:

2.2.1 Bag of words :

The bag-of-words is a technique that converts a collection of documents into a matrix where the rows represent different documents, and the columns are the distinct words in the entire collection. The value of a cell in the i^{th} row and the j^{th} column is the number of occurrences of the j^{th} word in the i^{th} document. While this technique is simple, it is very efficient and widely used in text classification tasks. However, it can result in a large matrix with millions of columns for extensive datasets (which can be addressed using dimensionality reduction techniques).

2.2.2 Term Frequency-Inverse Document Frequency

Term Frequency-Inverse Document Frequency (TF-IDF) is another frequency-based feature extraction technique that assigns higher weights to words with very high or very low frequency. This is because low-frequency words can also help distinguish different topics. The value of this weight is given by the following formula:

$$w(d,t) = \text{TF}(d,t) \times \log\left(\frac{N}{\text{df}(t)}\right)$$

where:

- d is a given document.
- t is a term (word).
- TF(d,t) the frequency of the term t in the document d.
- N is the total number of documents.
- df(t) is the number of documents that contain the term t.

3 Semi-supervised learning

In semi-supervised learning (SSL), only a portion of the data is labeled, usually with a small fraction, and the unlabeled data is used to improve the model's performance and provide it with better generalization. This makes it ideal for tasks where labeling data points is expensive and time-consuming.

3.1 Self-Learning

In self-training, an initial model is built using only labeled data points and a supervised learning algorithm. Predictions are then made on the unlabeled data, and instances classified confidently by our model are added to the training set. To identify these instances, a certain threshold (a hyperparameter of the algorithm) is fixed. An instance is added to the training set only if : $max\{P(y=0), P(y=1)\} > threshold$, The model is then re-trained on the new training set. This process is repeated until one of the following conditions is satisfied: the dataset is fully labeled, a number of iterations is reached or no unlabeled data points were added to the training set during the previous iteration, or some early stopping criterion is satisfied.

Algorithm 1 Self training algorithm

```
Require: model : a probabilistic model
Require: $threshold \in [0,1]$
Require: $max\_iter$ a positive integer.
  $trainset \leftarrow labled\_set$
  $stop \leftarrow False$
  $n \leftarrow 0$
  while not $stop$ do
      train(model, trainset.X, trainset.y)
      $\hat{y} \leftarrow predict(model, unlabled\_set)$
      $trainset \leftarrow trainset \bigcup unlabled\_set[\text{argmax}\,\hat{y} \geq threshold]$
      $n \leftarrow n+1$
      $stop \leftarrow unlabled\_set[\text{argmax}\,\hat{y} \geq threshold]$ is empty or $n >= max\_iter$
  end while
```

The labels of new data points are then predicted using the last trained model.

3.2 Graph-Based methods

In Graph-Based methods, the dataset is represented as a graph where each data point is a node. The weight matrix w is a matrix used to represent the graph, where $w_{i,j} \neq 0$ if x_i and x_j are neighbors and 0 otherwise. The function used to calculate the weight matrix is called a kernel, which takes as input tow vectors of the same dimension and compute their similarity.

KNN Kernel:
$K(x_i, x_j) = 1$ if x_i is among the k-nearest neighbors of x_j,0 otherwise.

RBF Kernel:
$K(x_i, x_j) = e^{-\frac{(\|x_i - xj\|)^2}{2*\sigma^2}}$.

in both kernels k and σ are considered as hyperparameters.

After calculating the weight matrix, an iterative algorithm can be employed to propagate the labels. In this process, each node propagates its label to its neighbors, and the iteration continues until convergence, typically defined by reaching a certain number of iterations. Two commonly used algorithms for label propagation are **Label Propagation** and **Label Spreading**.

In both algorithms, $\hat{Y}$ is represented as an $m \times k$ matrix, where k denotes the number of distinct labels. The initialization of matrix Y is such that $\hat{Y}_{i,j} = 1$ if the i^{th} data point is labeled with the j^{th} label. At convergence, the resulting matrix is denoted as $\hat{Y}^{\infty}$, which is interpreted as the probability of the i^{th} document being labeled with the j^{th} label. In other words, $\hat{Y}^{\infty}_{i,j} = P(y_i = C_j)$.

3.2.1 Label propagation

Algorithm 2 Label propagation : Zhu and Ghahramani

```
  compute the weight matrix $w$
  compute diagonal matrix $D_{i,i} = \sum_i W_{i,j}$
  initialize $\hat{Y}^{(0)}$
  **repeat**
      $\hat{Y}^{t+1} \leftarrow D^{-1} W \hat{Y}^{t}$
      $\hat{Y_l}^{t+1} \leftarrow {Y_l}^{t}$
  **until** convergence
```

3.2.2 Label spreading

Algorithm 3 Label spreading : Zhou et al

```
**Require:** $\alpha \in [0, 1)$
  compute the weight matrix $w$ with $w_{i,i} \leftarrow 0$
  compute diagonal matrix $D_{i,i} = \sum_i W_{i,j}$
  compute the matrix $L \leftarrow D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$
  initialize $\hat{Y}^{(0)}$
  **repeat**
      $\hat{Y}^{t+1} \leftarrow \alpha L \hat{Y}^{t} + (1 - \alpha) \hat{Y}^{t}$
  **until** convergence
```

Predicting on new data points :

Let x be a new data point,we calculate a vector w of dimension m,representing the similarity between the data point x and all the data points in the dataset used to train the model,calculated using the appropriate kernel,we then calculate the probabilities for each label using the following formula :

$$p = \frac{w \times \hat{Y}^{\infty}}{\sum\limits_{i=1}^{m} w_i}$$

then : $\hat{y} = \underset{c_k \in C}{\operatorname{argmax}}\, p$

4 Dataset

4.1 Dataset description

The dataset used to train models is available through the following link: Spam Mails Dataset SSL. This dataset consists of 4993 rows. After removing duplicates, it has one feature, which is the content of the email. The target class is assigned a value of 0 for either spam,1 for ham, and -1 for unlabeled examples. Approximately 79% of the rows are unlabeled, while 71.22% of the labeled examples are ham (not spam), making this dataset imbalanced,And to address this issue, we primarily attempted two techniques: oversampling and assigning a higher weight to the minority class.

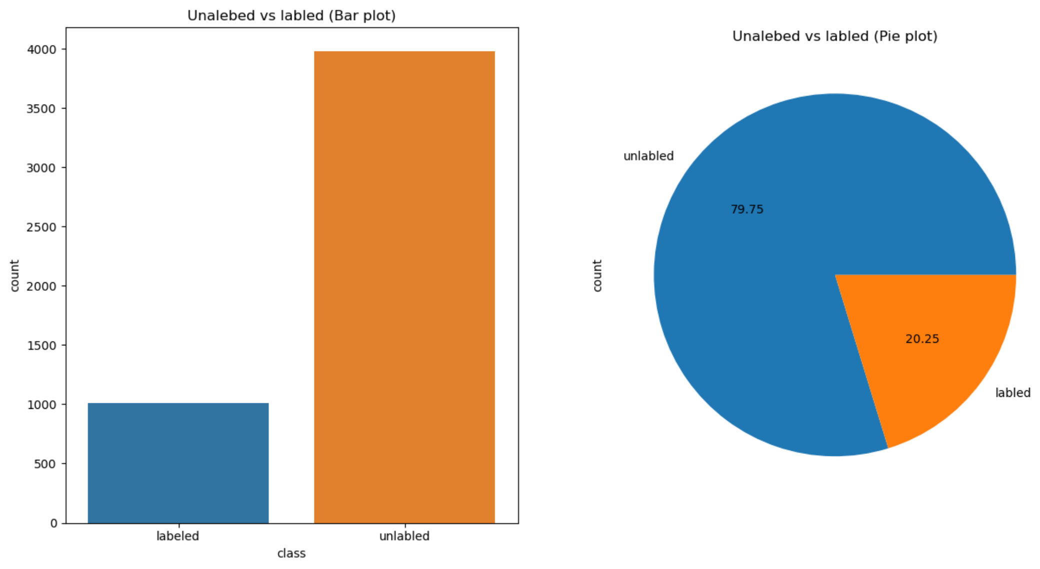


Figure 1: The distribution of the labled and the unlabled documents

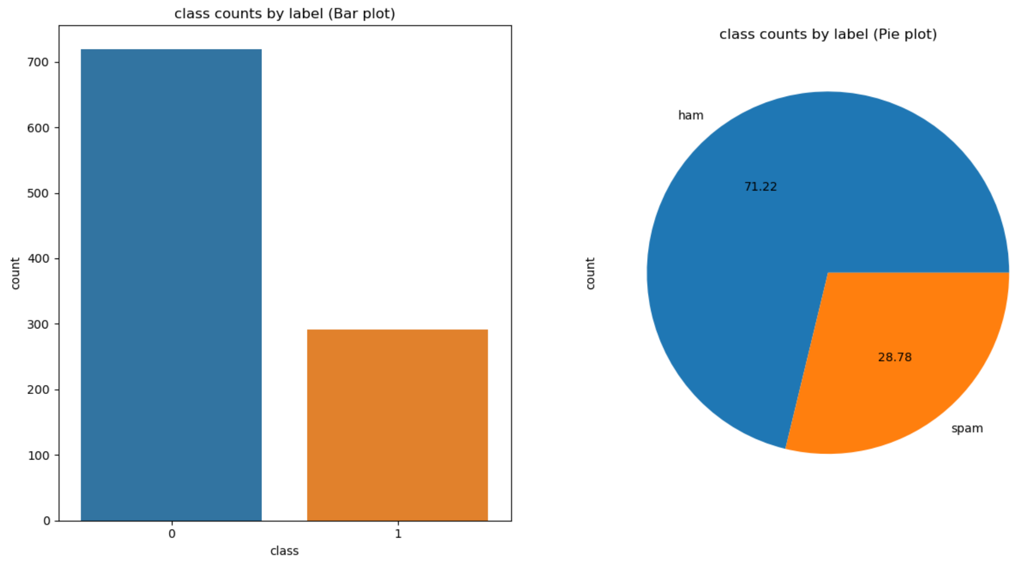


Figure 2: The distribution of the spam and the ham documents

4.2 Feature extraction

In this step, we attempted to extract certain features from the text that can be utilized in conjunction with the features that will be obtained through frequency-based methods (BOW & TF-IDF) applied to the text,and these features are :

- The number of URLs: Many messages may contain URLs that link to external sites, such as shopping sites or potentially harmful sites that could compromise the user's privacy.
- digits counts : Many some emails may also contain many numerical strings (like prices of products).
- whether the emails contains currency symbols or not.
- the length of the text.

4.3 Dataset analysis

The number of URLs turned out to be useless because non of the emails contains URLs (feature with one value).

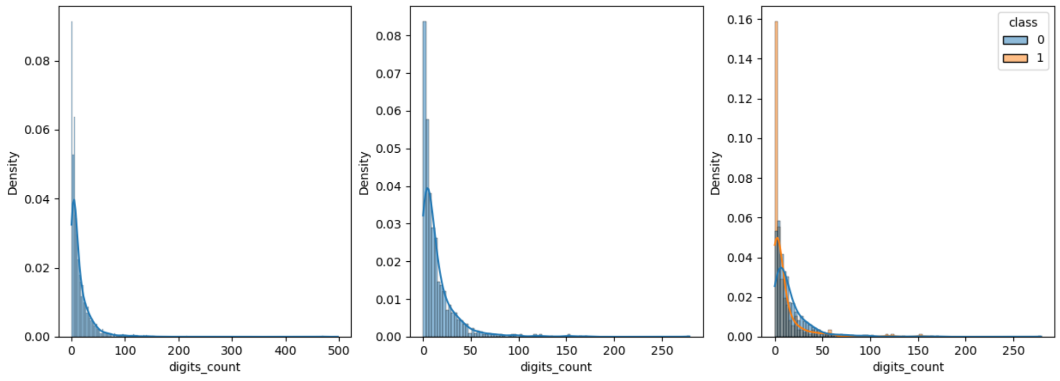


Figure 3: The distribution of the number of numerical strings for the entire dataset, the labeled data points only, and by category.

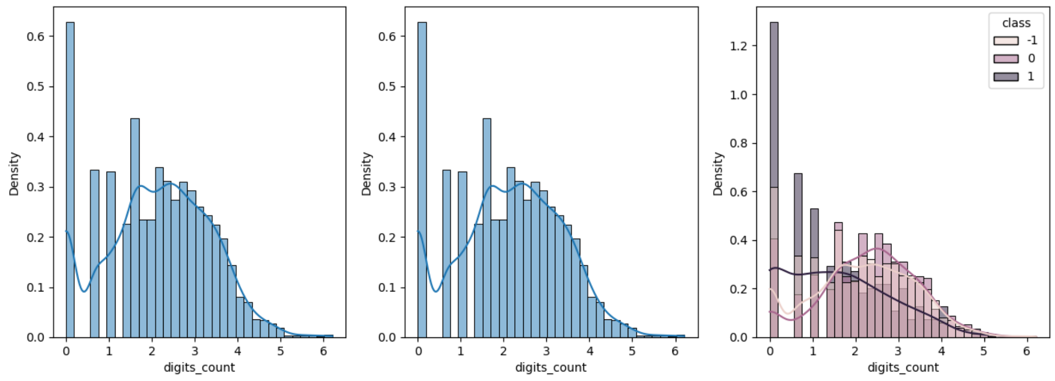


Figure 4: The distribution of the number of numerical strings for the entire dataset, the labeled data points only, and by category. (log scaled)

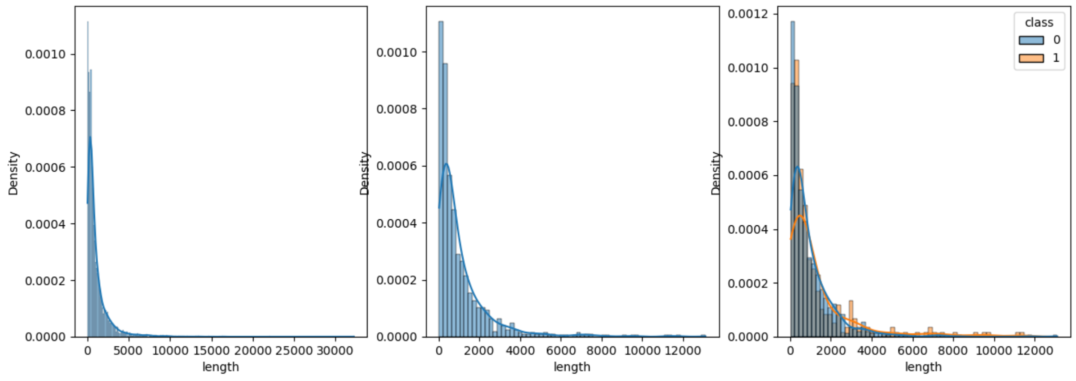


Figure 5: The distribution of length of the text for the entire dataset, the labeled data points only, and by category.

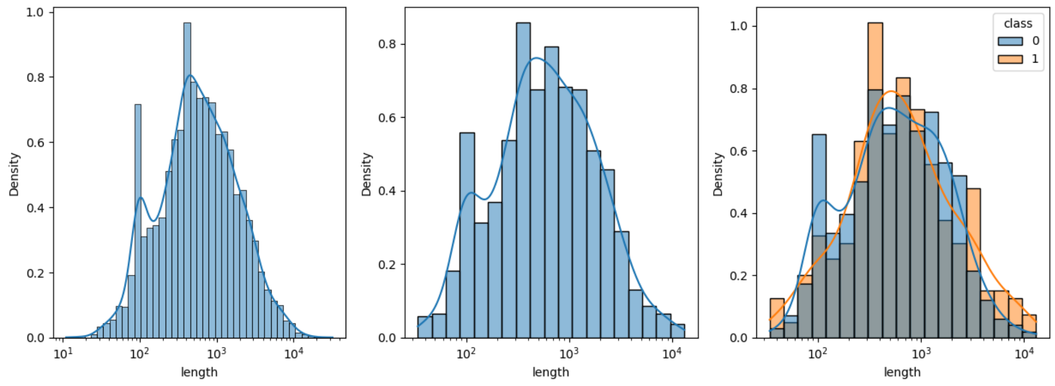


Figure 6: The distribution of length of the text for the entire dataset, the labeled data points only, and by category. (log scaled)

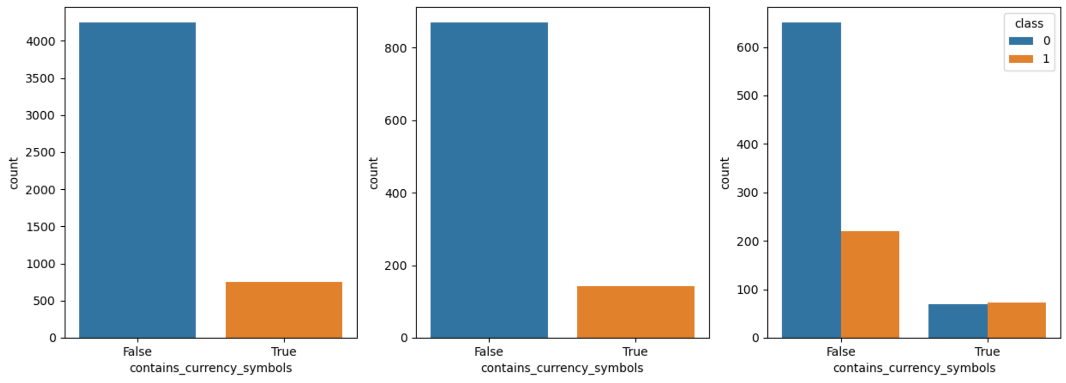
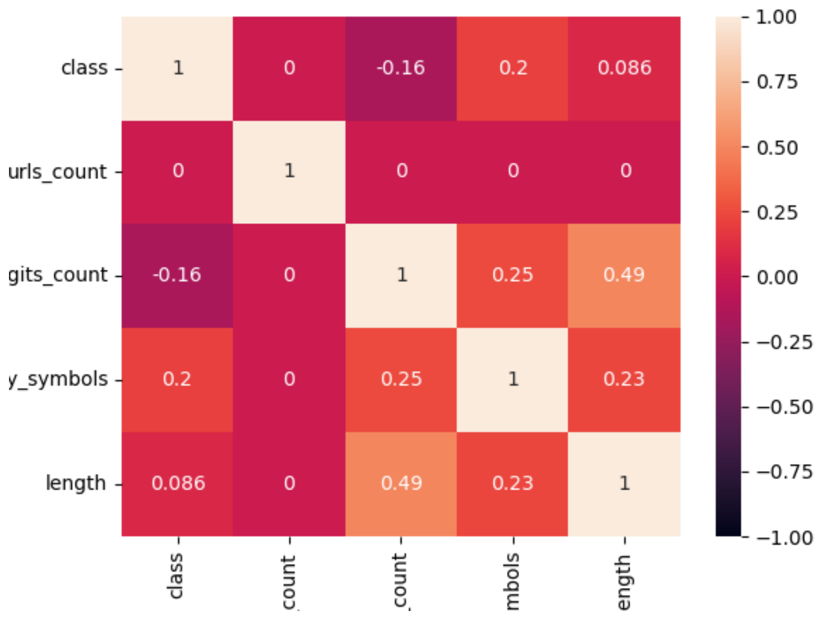


Figure 7: The number of emails that contain currency symbols for the entire dataset, the labeled data points, and by category.



	class	count	count	mbols	ength
class	1	0	-0.16	0.2	0.086
urls_count	0	1	0	0	0
gits_count	-0.16	0	1	0.25	0.49
y_symbols	0.2	0	0.25	1	0.23
length	0.086	0	0.49	0.23	1

Figure 8: Correlation heatmap

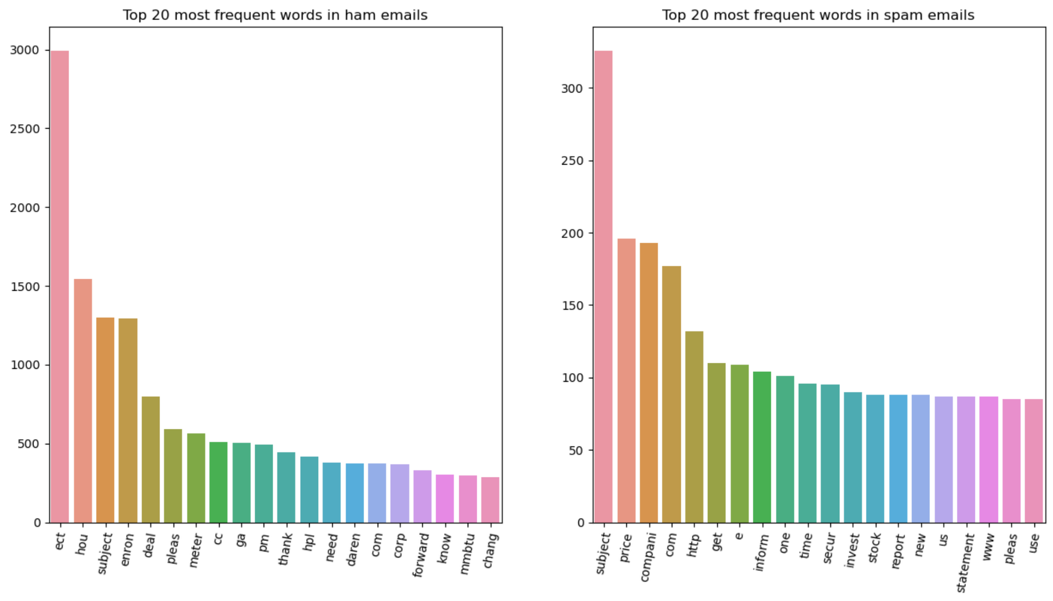


Figure 9: The top 20 most frequent words in ham and spam emails respectively

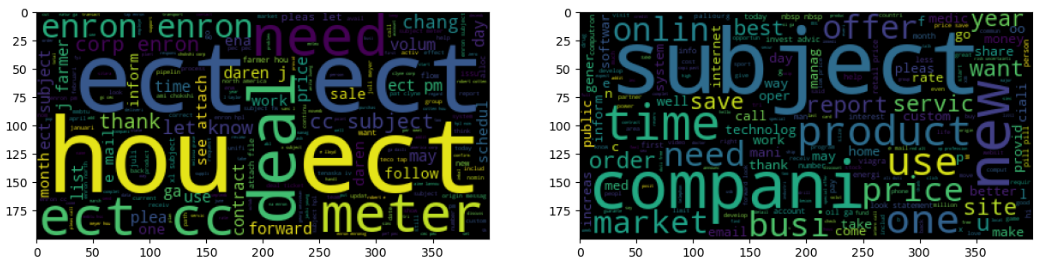


Figure 10: The wordclouds for the ham and spam data points respectively

4.4 Conclusions

- Using a logarithmic scale can help in making features follow a Gaussian distribution.
- By observing the similarity between the plots of the entire dataset and those of the labeled data points alone, we can conclude that the labeled data points are representative of the entire dataset. This is an important hypothesis in semi-supervised learning.
- The extracted features (URLs count, length, etc.) will not be used due to their low correlation with the target feature.
- Based on the word frequencies graphs and word clouds, frequency-based feature extraction techniques (BOW & TF-IDF) appear promising.

5 Training Results

In this chapter, we will explore the obtained results for three algorithms discussed earlier, employing two different feature extraction techniques (Bag of Words & TF-IDF). We will consider variations with and without dimensionality reduction, oversampling, and different hyperparameters. Additionally, for the self-learning algorithm, we have experimented with and compared various base models.

To compare the different models, we've chosen the F1-score as the criteria for selecting the best model. This decision is driven by the dataset's imbalance, making accuracy unreliable. Given the equal importance of spam and ham messages, we opted against criteria that favor one class over another, such as precision and recall.

5.1 Self Learning

We have previously mentioned that the self-training algorithm is a meta-model employing another supervised learning algorithm to iteratively label unlabeled examples in our dataset. To select candidate models for our meta-model, we execute various learning algorithms with different combinations of hyperparameters and employ different feature extraction and preprocessing techniques. The tables below provides a summary of the results obtained for each algorithm.

5.1.1 Comparing the different Algorithms

Logistic regression :
Note : for the models that uses oversampling only the results of the best model are shown.

Feature extraction	Scaling	Dimentiality reduction	Oversampling	best hyper parameters	f1-score
Bag of words	maximum absolute value	no	no	positive class weight: 0.85	0.743
Bag of words	standard scaling	yes	no	positive class weight: 0.6	0.892
Tf-IDF	maximum absolute value	no	no	positive class weight: 0.8	0.890
Tf-IDF	standard scaling	yes	no	positive class weight: 0.65	0.889
Tf-IDF	maximum absolute value	no	yes	positive class weight: 0.5	0.944

Table 1: Logistic regression results for different preprocessing and feature extraction techniques

SGD Logistic regression
Note : for the models that uses oversampling only the results of the best model are shown.

Feature extraction	Scaling	Dimentiality reduction	Oversampling	best hyper parameters	f1-score
Bag of words	maximum absolute value	no	no	positive class weight: 0.9	0.819
Bag of words	standard scaling	yes	no	positive class weight: 0.65	0.887
Tf-IDF	maximum absolute value	no	no	positive class weight: 0.55	0.873
Tf-IDF	standard scaling	yes	no	positive class weight: 0.8	0.874
Tf-IDF	maximum absolute value	no	yes	positive class weight: 0.5	0.890

Table 2: SGD Logistic regression results for different preprocessing and feature extraction techniques

K-Nearest neighbors

Feature extraction	Scaling	Dimentiality reduction	Oversampling	best hyper parameters	f1-score
Bag of words	maximum absolute value	no	yes	k:3,p:2,weights:distance	0.587
Bag of words	standard scaling	yes	yes	k:11,p:2,weights:distance	0.786
Tf-IDF	maximum absolute value	no	yes	k:3,p:2,weights:distance	0.578
Tf-IDF	standard scaling	yes	yes	k:3,p:2,weights:distance	0.597

Table 3: K-Nearest neighbors results for different preprocessing and feature extraction techniques

Naive Bayes

Algorihtm	Feature extraction	f1-score
Multinomial	Bag of words	0.880
Multinomial	TF-IDF	0.446
Complement	Bag of words	0.936
Complement	TF-IDF	0.818
Bernoulli	Bag of words	0.392
Bernoulli	TF-IDF	0.386

Table 4: Naive Bayes results for different algorithms and feature extraction techniques

SVM

Feature extraction	Scaling	Dimentiality reduction	Oversampling	best hyper parameters	f1-score
Bag of words	maximum absolute value	no	yes	C:1,kernel:linear	0.773
Bag of words	standard scaling	yes	yes	C:20,kernel:sigmoid	0.885
Tf-IDF	maximum absolute value	no	yes	C:10,kernel:linear	0.937
Tf-IDF	standard scaling	yes	yes	C:10,kernel:rbf	0.886

Table 5: SVM results for different preprocessing and feature extraction techniques

Decision tree

Feature extraction	Oversampling	best hyper parameters	f1-score
Bag of words	yes	max depth: 20, min impurity decrease: 0.0, min samples split: 2	0.836
TF-IDF	yes	max depth: 50, min impurity decrease: 0.0, min samples split: 2	0.803
Bag of words	no	max depth: 100, min impurity decrease: 0.0, min samples split: 10	0.825
TF-IDF	no	max depth: NA, min impurity decrease: 0.0, min samples split: 10	0.772

Table 6: Decision tree results for different preprocessing and feature extraction techniques

XGBOOST classifier
Note : the scale pos weight parameters controls the weight of the minority class.

Feature extraction	best hyper parameters	f1-score
Bag of words	learning rate: 0.01, max depth: 20, scale pos weight: 1	0.841
TF-IDF	learning rate: 0.01, max depth: 20, scale pos weight: 6	0.831

Table 7: XGBOOST results for different preprocessing and feature extraction techniques

5.1.2 Comparing the different base models

The table below compares the best model for each algorithm in terms of accuracy, precision, recall, F1-score, and Brier score loss.

Algorithm	accuracy	precision	recall	f1-score	brier score loss
SGD Logistic Regression	0.939	0.920	0.867	0.891	0.048
Decision Tree	0.902	0.802	0.871	0.835	0.093
SVM	0.962	0.925	0.946	0.934	0.031
XGBOOST Classifier	0.914	0.822	0.903	0.860	0.097
KNN	0.841	0.650	0.976	0.780	0.122
Logistic Regression	0.971	0.944	0.954	0.948	0.030
Complement Naive Bayes	0.968	0.945	0.944	0.944	0.030

Table 8: Comparing the different algorithms

5.1.3 Calibration graphs & Probabilities distributions

The self-learning algorithm necessitates a well-calibrated model. A well-calibrated model is a probabilistic model capable of generating probabilities (i.e., scores between 0 and 1) that exhibits confidence in its predictions. This confidence is reflected in the predicted probabilities being either close to 0 or close to 1. This property can be visualized using a calibration curve and histogram of the predicted probabilities, and assessed using the Brier score loss.

A calibration curve is a visual representation of the agreement between predicted probabilities and the actual labels of a classification model. It involves binning the predicted probabilities into intervals and plotting the mean or observed frequency of positive data points within each bin. Ideally, a perfectly calibrated model would show a diagonal line (i.e its equation is $y = x$).

Note that a well-calibrated model doesn't necessarily mean a model has good performance; a model can be confident in its predictions and still have low accuracy.

The Brier score loss is a good measure that assesses the model's calibration property without ignoring its accuracy. It calculates the mean sum of the squares of the differences between the predicted probabilities and the actual labels,and its given by the following formula :

$$\text{BS} = \frac{1}{m}\sum_{i=1}^{m}(f_{prob}(x_i) - y_i)^2$$

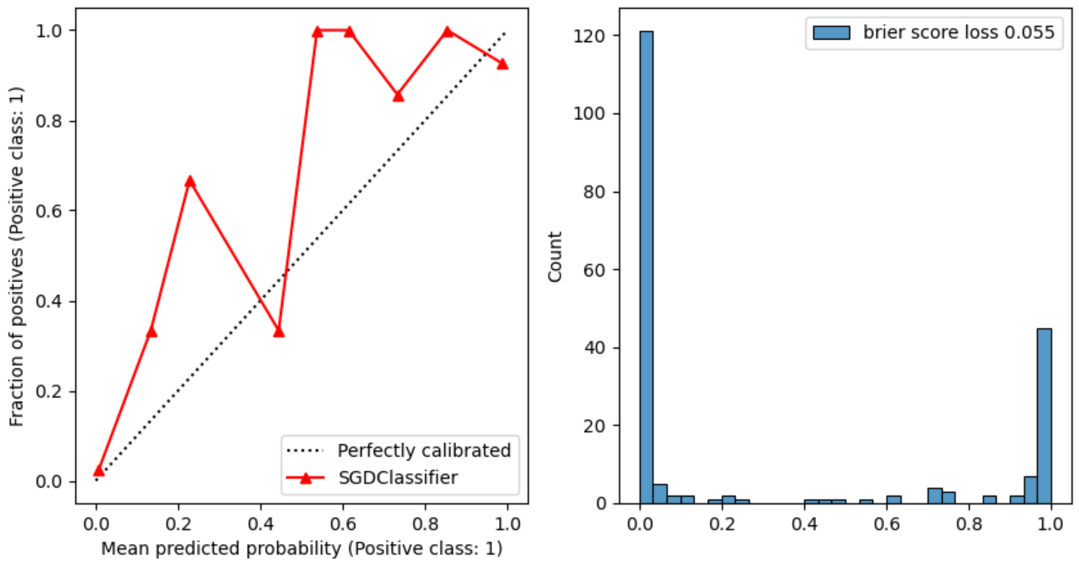


Figure 11: Calibration graph for SGD Logistic Regression

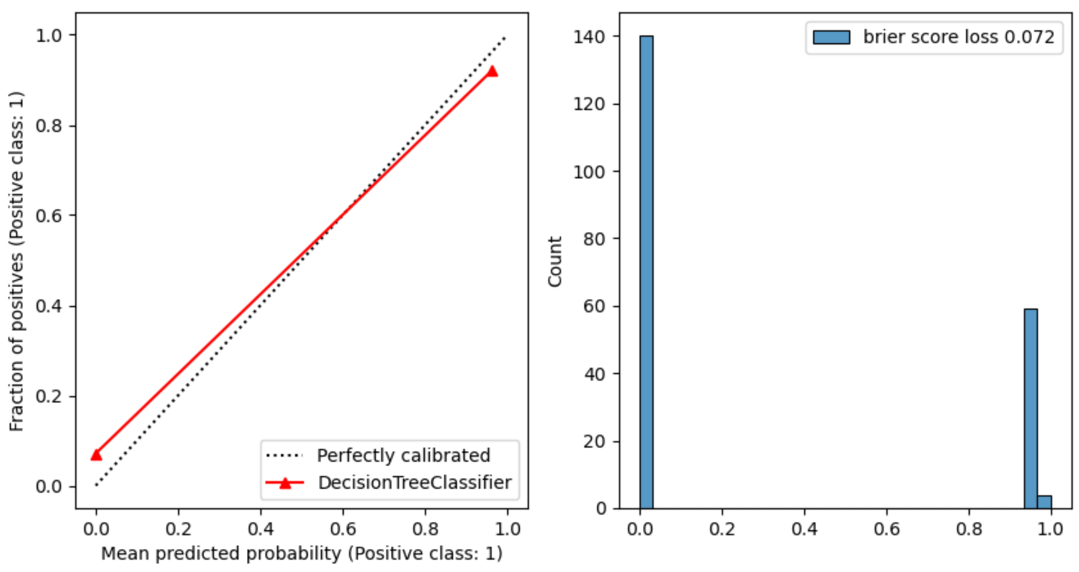


Figure 12: Calibration graph for Decision tree

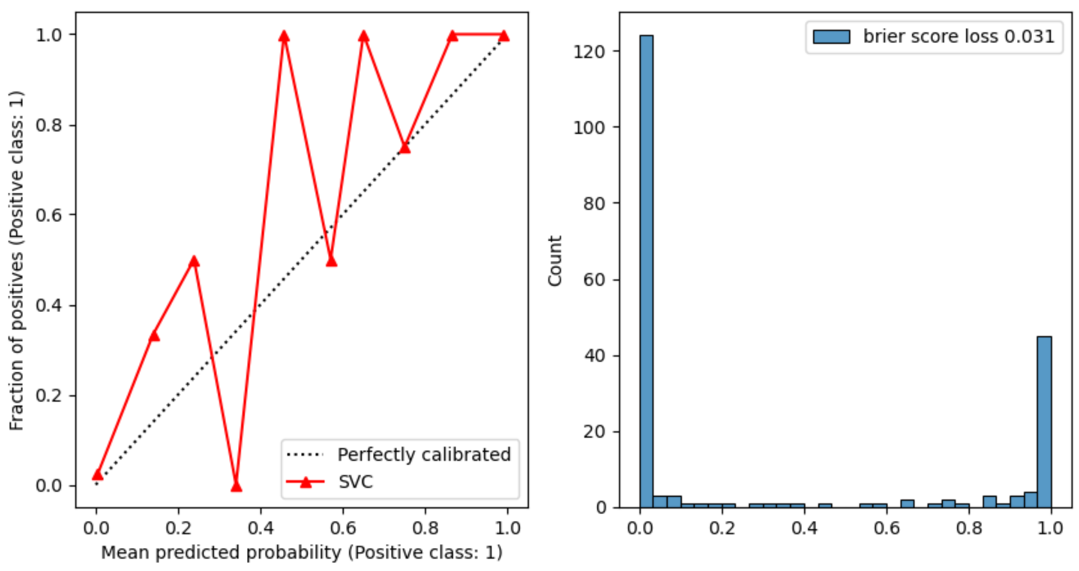


Figure 13: Calibration graph for SVM

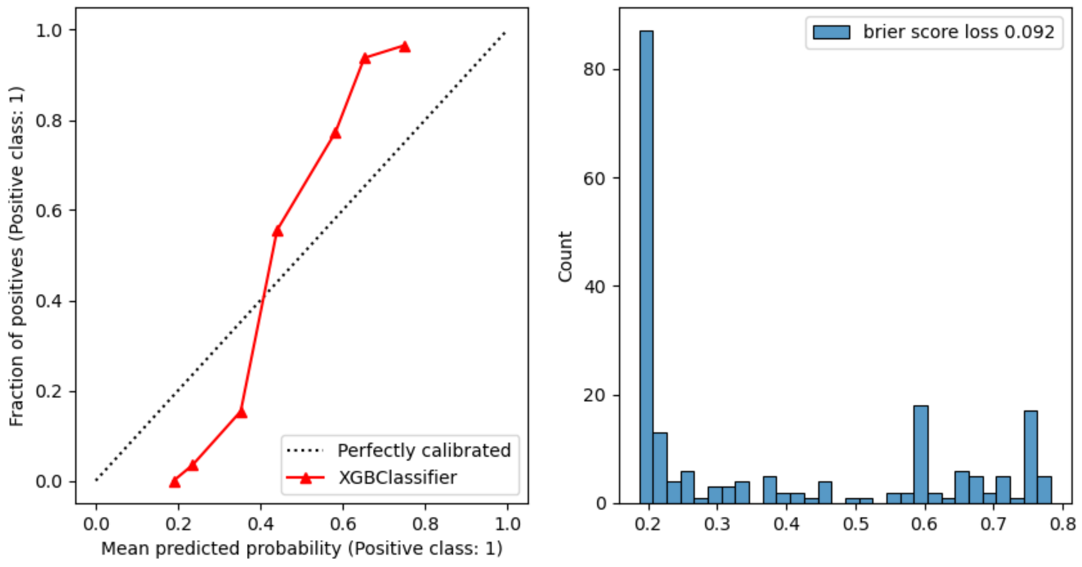


Figure 14: Calibration graph for XGBOOST Classifier

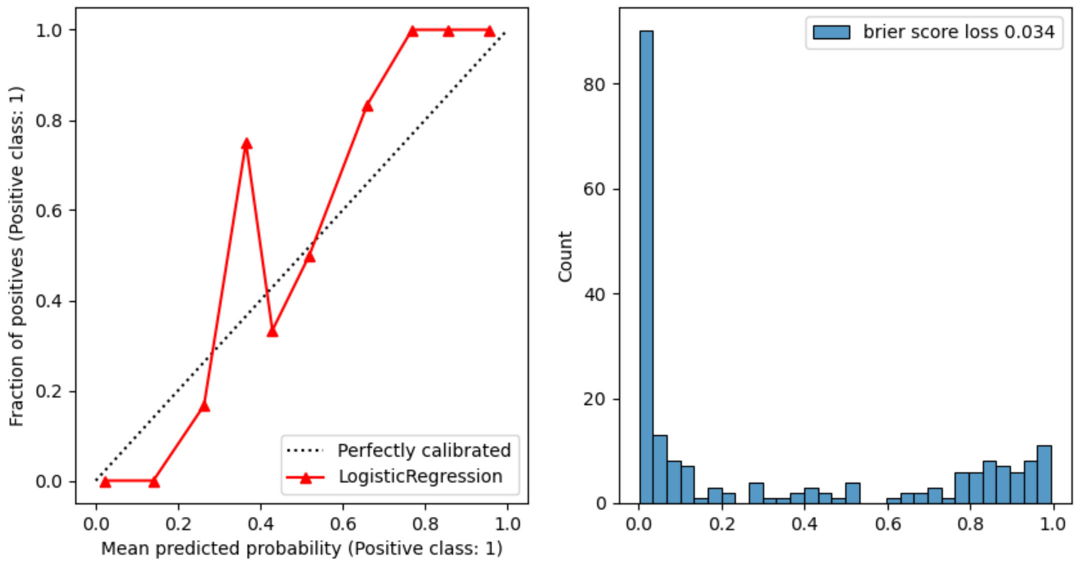


Figure 15: Calibration graph for Logistic Regression

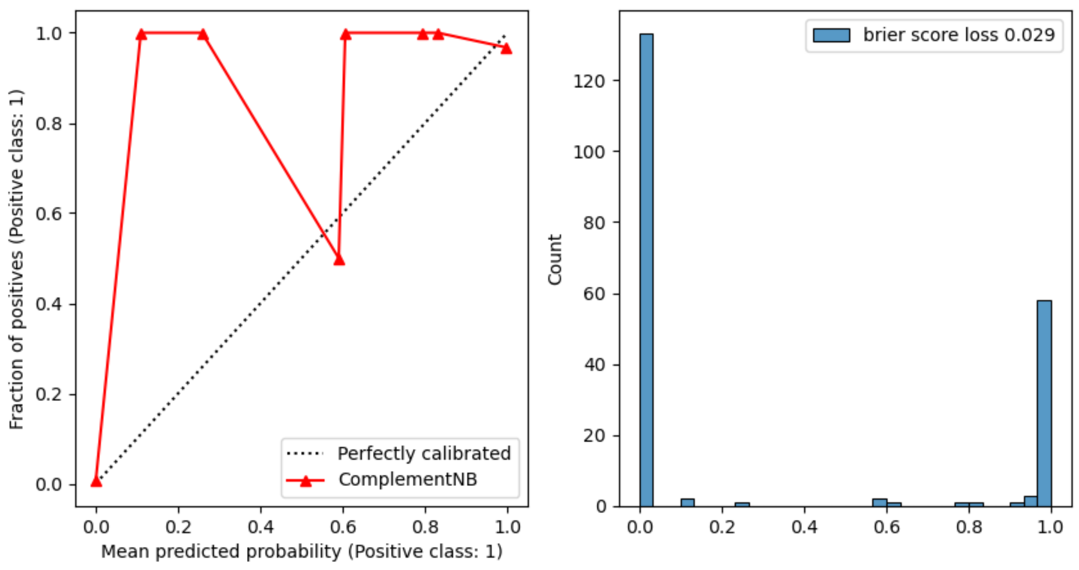


Figure 16: Calibration graph for Complement Naive Bayes

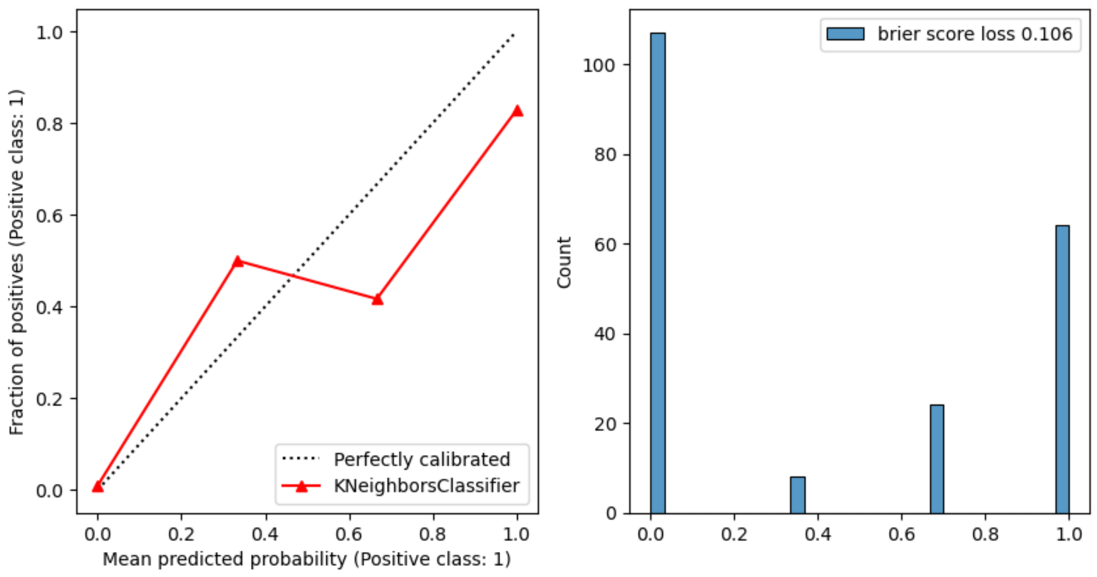


Figure 17: Calibration graph for KNN

From the graphs above, we can observe that Complement Naive Bayes, SVM, Logistic Regression, and Decision Tree models are well-calibrated. However, the Decision Tree serves as evidence that a well-calibrated model does not necessarily guarantee good performance. On the other hand, the other models are not well-calibrated.

To train the self-learning model, we will use as base models the best models of the four following algorithms : SGD Logistic Regression, SVM, Naive Bayes, and we'll also attempt Logistic Regression, despite its lack of good calibration because of its relatively low Brier score.

We will split the dataset into two parts: a training set containing all the unlabeled documents as well as a portion of the labeled examples, and the remaining labeled documents will be kept in a test set for evaluation,the table below compares the different models in terms of accuracy,precision,recall and f1-score.

Base model	accuracy	precision	recall	f1-score
SGD Logistic Regression	0.976	0.988	0.936	0.962
SVM	0.950	0.976	0.863	0.916
Logistic Regression	0.980	0.989	0.947	0.967
Complement Naive Bayes	0.976	0.968	0.957	0.962

Table 9: Comparing the self learning algorithm

5.2 Label spreading

The dataset is split into two sets: a training set containing both unlabeled and labeled data points, and a test set consisting only of labeled data points, which is used for evaluation.

Feature extraction	kernel	Dimentiality reduction	accuracy	precision	recall	f1-score
Bag of words	rbf	no	0.763	0.632	0.364	0.46
Bag of words	rbf	yes	0.845	0.720	0.729	0.725
Bag of words	knn	no	0.789	0.576	0.929	0.711
Bag of words	knn	yes	0.861	0.718	0.847	0.774
TF-IDF	rbf	no	0.907	0.851	0.811	0.831
TF-IDF	rbf	yes	0.657	0.449	1.000	0.624
TF-IDF	knn	no	0.914	0.927	0.752	0.831
TF-IDF	knn	yes	0.805	0.597	0.941	0.730

Table 10: Label spreading's performance for different feature extraction techniques and kernels

5.3 Label propagation

The dataset is split into two sets: a training set containing both unlabeled and labeled data points, and a test set consisting only of labeled data points, which is used for evaluation.

Feature extraction	kernel	Dimentiality reduction	accuracy	precision	recall	f1-score
Bag of words	rbf	no	0.743	0.563	0.364	0.442
Bag of words	rbf	yes	0.819	0.674	0.682	0.678
Bag of words	knn	no	0.730	0.509	0.964	0.666
Bag of words	knn	yes	0.812	0.614	0.882	0.724
TF-IDF	rbf	no	0.891	1.000	0.611	0.759
TF-IDF	rbf	yes	0.592	0.406	1.000	0.578
TF-IDF	knn	no	0.917	0.954	0.741	0.834
TF-IDF	knn	yes	0.756	0.964	0.535	0.689

Table 11: Label propagations's performance for different feature extraction techniques and kernels

5.4 The best model :

According to the chosen metric for this particular problem, which is the F1-score for the reasons discussed earlier, the best model is the Self-Learning classifier with Logistic Regression as the base model.

6 Conclusion

In this project, we explored three different semi-supervised learning algorithms and discussed their implementation. We applied these algorithms to an email spam classification dataset, combining semi-supervised learning and NLP techniques to build a model capable of robustly identifying spam emails from non-spam emails.

7 References

- I. Chapelle, Olivier. II. Schölkopf, Bernhard. III. Zien, Alexander. IV. Series.,"Semi-Supervised learning (Machine learning)",2006
- Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu,Laura Barnes and Donald Brown,"Text Classification Algorithms: A Survey",April 2019
- Sahel Eskandar,"Exploring Feature Extraction Techniques for Natural Language Processing.",Meduim,April 2023
- semi supervised learning user's guide,sklearn.