Sentiment Analysis and Comparison of the Naïve Bayes Classifier and K-Nearest Neighbours for Spam Detection

Purchesta Aria

Queen Mary University of London

School of Mathematical Sciences

A project submitted in partial fulfilment of the Bsc in

Mathematics

7th May 2021

Acknowledgements Page

I would like to thank my supervisor Wolfram Just, for supporting me throughout this project and providing me the foundation knowledge to carry out this research.

I would like to thank my entire family, Behzad Aria, Metra Aria, Avesta Aria, Bejan Aria, Anahita Aria, Qumars Aria, Gulshan Aria, Esfandiar Aria, Farangis Aria and
Arash Aria; fort their support and encouragement.

And, finally I would like to thank my dearest friends Oana Petras, and Jawad Jahangir for supporting me during difficult times.

Table of Contents

Abstract

Over the recent decades, there has been a strong motivation for the study of sentiment analysis. Sentiment analysis is a classification process used to extract the overall sentiment, behaviour, attitude of a given entity or opinion. As human beings, we value other opinions when making decisions and the application of opinions can be seen in a variety of domains, from politics to businesses such reviews. Unsurprisingly the large volume of opinionated data has resulted in opinion spammers who are users that corrupt the system by posting fake opinions to discredit a person. Therefore, through machine learning, spam detection classifiers have been developed over the years, to assist in filtering these spamming data. In this project, I will be exploring spam detection in emails and assessing the performance of the Naïve Bayes classifier and K- Nearest Neighbours in filtering out spam emails from my inbox.

Chapter 1 Introduction

1.1 Sentiment Analysis

The oxford definition for sentiment analysis is "the process of computationally identifying and categorizing opinions expressed in a piece of text, especially to determine whether the writer's attitude towards a particular topic, product, etc is positive, negative or neutral". **(Anon., n.d.)**. Essentially it's the computational study of people's opinions, behaviour, and emotions towards an object. **(Hassan, et al., 2014)**.

Opinions are a perspective or judgment created in the mind about an entity, it can be both factual or fabricated. As human beings, the foundation of our activities is orientated around the opinions we have. These sculpt our actions, beliefs, and behaviour. Furthermore, they are

key for decision making; henceforth we value our peers' opinions as a guide when making choices as it provides an opportunity to share ideas and learn from each other. The prominence of opinions is shown through the variety of applications in every domain from voters wanting to gather information about a political candidate, to businesses requiring public opinion on their services or product **(Liu, 2012)**.

The research of sentiment analysis primarily started over the last few decades and is a growing and active area of study due to the extortionate volume of opinionated data. Since the objective of sentiment analysis is the extract the overall impression of a given opinion, it ties in very well with the field of linguistics and natural language processing. Although, these fields have a long history sentiment analysis provides to be a challenging area of study due to minimal research conducted before the year 2000. **(Liu, 2012)** One of the reasons for this could be due to the lack of opinionated data we had before in digital form. Nonetheless, the proliferation of the web and social media such as Twitter, Instagram, etc has contributed greatly to the abundance of data over the last decades. **(Liu, 2012)**. In addition to this, the real-life applications and the impact it has a variety of domains from computer science to political sciences as well provides strong motivation for this research area. **(Liu, 2012)**.

Filtering and monitoring data provides be an issue and a demanding task for humans. **(Liu, 2012)**. An average human reader will face difficulty finding relevant information and scrutinizing what is of value to them. Being exposed to new thoughts will broaden their horizon and give them new ideas which they would require to find opinions for, leading to an infinite cycle. **(Liu, 2012)** Therefore, sentiment classification models through machine learning have been created and developed over the years to assist us in categorizing information. I will expand on this more in the next chapter.

1.2 Opinion Spamming

The web and social media offer online anonymity for everyone hence providing a platform for users to fully practice freedom of speech and movement by enabling people to speak their mind or watch/buy something without being judged and expressing their own opinions and freedom of movement by users watching or buying something without being judged. **(Terekhov, n.d.)**. Although online anonymity can be beneficial by protecting the users' identity from incurring any backlash which may have affected them in the real world, this aspect of anonymity is costly.

It allows people with malicious intents or hidden agendas to corrupt the system and gives the impression that they are honest members of the public and post fake opinions to discredit an organization, individual, product, or service. **(Liu, 2012)**. Additionally due to the profitable incentive associated with reviews, legitimate companies and businesses exploit the system and write fake reviews or ratings to discredit their fellow competitors. **(Mukherjee, et al., 2013)**. These users are known as opinion spammers and it's a growing issue. (Liu, 2012). It is vital to detect these spamming activities to consolidate the value of the opinionated data on the web.

1.3 Opinion Spamming in Email Systems

Opinion spamming is also evident in email inboxes, which is the form I will be the main focus of this project. These are known as spam emails which are the unsolicited emails a user receives in their inbox. . Nowadays, spam detection systems have been developed through supervised machine learning to automatically filter these out from our inbox. **(Liu, 2012)**

Chapter 2 Machine Learning Classifiers

One of the most common applications of sentiment analysis is classifying text to a given class. As discussed the classes we have outlined were positive, negative, and neutral, however, the application can be seen in spam filtering. Sentiment analysis is used through supervised machine learning algorithms for spam detection. The objective is to identify whether an email is a spam or not.

The formal definition for a classifier is a function, f that maps a given input data point x to one or more set of output classes c ($\in C = c_1.c_2, \dots c_n$) **(Murphy, 2006)** Specifically, in supervised machine learning, the outputs are a fixed set of classes, defined by a labelled training set. Using the training data set, the objective of a supervised learning classifier is to recognisee any patterns to categorize any new input data to a class label. **(Singh, et al., 2016) (Murphy, 2006) (Mesevage, 2020)**.

Supervised classification is generally conducted through two stages. Firstly a classification model is created by applying a supervised algorithm on a training data set. In this training stage, all input points are labelled to the correct class. The labels which are accurately assigned in this training set can be referred to as human-defined labels or golden labels. Once our classification model is created the second stage is testing this against a testing data set. In this stage, the classifier the prior data, to predict the class of the input points in the testing data set. **(Mesevage, 2020)**

Furthermore, a problematic classifier such as Naïve Bayes will go further to return the probability of the input data being in the class. **(Jurafsky & Martin, 2019)**. We can see the

application of classification in a variety of domains' the most common form is labelling an email as spam or not spam which is the domain I will be using. For the purpose of this project, the two classifiers I have selected to compare are the Naïve Bayes Classifier and K-Nearest Neighbours.

2.1 Naïve Bayes

Navies Bayes is a supervised machine learning classifier and belongs to the family of generative classifiers. These construct a model of how a class could generate an observation by returning the most common class for a given input. **(Jurafsky & Martin, 2019)** It is one of the most frequently used methods in extracting sentiments and categorizing text.

The intuition of the classifier is as follows, say we have a text $\boldsymbol{t}$ which we will classify into a class $\boldsymbol{c}$, where $\boldsymbol{c}$… Firstly using the text we create a bag of words which is an unordered set of all the words. The frequency of each word is considered and the position is unimportant. **(Jurafsky & Martin, 2019)**

As discussed briefly as Naïve Bayes is a problematic classifier, for the input text the classifier will assign it to a class label $\boldsymbol{c_x}(\in \boldsymbol{C})$. which returned the highest posterior probability. An equation is shown below. **(Jurafsky & Martin, 2019)**

Equation 1

$$\boldsymbol{c_x} = \max_{\boldsymbol{c} \in \boldsymbol{C}} \boldsymbol{P}(\boldsymbol{c}|\boldsymbol{t})$$

For Navies Bayes, this classifier is inspired by Bayes Theorem which is a method used to calculate probabilities based on observations. **(McNamara, et al., 2006)**. This theorem is also

known as the conditional probability formula which is shown below. **(Jurafsky & Martin, 2019)**

Equation 2: Bayes Theorem/ Conditional Probability

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The intuition of the Bayes Theorem is substituted into the classifier to break down any conditional probabilities into three other probabilities. **(Jurafsky & Martin, 2019)**

Equation 3

$$c_x = \max_{c \in C} P(c|t) = \max_{c \in C} \frac{P(t|c)P(c)}{P(t)}$$

We can leave it as Equation 3 , however in our case this can be further reduced by removing the denominator as we want the most likely class for the same text t, thus must have the same probability $P(t)$.This results to our Naïve Bayes Classifier formula, Equation 4 **(Jurafsky & Martin, 2019)**

Equation 4

$$c_x = \max_{c \in C} P(c|t) = \max_{c \in C} P(t|c)P(c)$$

As we can see from (4), the most likely class is a product of two probabilities, the likelihood of the text, $P(t|c)$, and the prior probability of the class, $P(c)$. **(Jurafsky & Martin, 2019)**

As discussed, the text $\boldsymbol{t}$ can be treated as a bag of words. This can be defined as a set W of all the words contained in $\boldsymbol{t}$. Therefore, without loss of generality, we can further rewrite t as a set of words

Equation 5

$$\boldsymbol{t = w_1, w_2, \dots, w_n}$$

Hence, we can substitute Equation 5 into Equation 4 leaving us with: **(Jurafsky & Martin, 2019)**

Equation 6

$$\boldsymbol{c_x = \max_{c \in C} P(t|c)P(c) = P(w_1, w_2, \dots w_n|c)P(c)}$$

Equation (6 looks like a tedious and difficult formula. We can simplify this by 'naively' taking the product of each of the conditional probabilities in Equation 6, $\boldsymbol{P(w_n|c)}$. **(Jurafsky & Martin, 2019)**. This is referred to as the Naïve Bayes assumption, where the independence of each conditional probability is assumed, resulting in the final derivation of the formula as, **(Jurafsky & Martin, 2019)**

Equation 7: Naïve Bayes Formula

$$\boldsymbol{c_x = \max_{c \in C} P(w_1, w_2, \dots w_n|c)P(c) = \max_{c \in C} P(w_1, w_2, \dots w_n|c)P(c)}$$

Or we can also rewrite it as,

Equation 8 Naïve Bayes Formula (2)

$$c_x = \max_{c \in C} P(c) \prod_{w \in W} P(w|c)$$

The advantage of the Naïve Bayes classifier is that the performance is fast and successful with small amounts of training data. **(Boehmke, n.d.)**. It applies to a majority of domains that require detection software and appropriate for solving multi-class prediction problems. Furthermore, the Naïve Bayes assumptions allow the classifier to perform more efficiently, without requiring large training data. However, this is only in case if the assumption holds to be true, which in some cases isn't true. This assumption establishes the idea that all predictors are independent, which is not the case in real life. Therefore reducing the applicability of it to real-world scenarios. **(Vadapalli, 2021)** Additionally, if a label isn't available in the training set, the classifier is known to output a 'zero probability for the testing data it was unable to classify. **(Vadapalli, 2021)**. Also, the results are not 100% correct in some cases.

2.2 K-Nearest Neighbours

The second classifier I have selected to focus on is K-Nearest neighbours. Similarly, it's a supervised machine learning algorithm that relies on prior labelled data to learn a function to then produce an appropriate output when given new data. K- Nearest neighbours are one of the simplest algorithms, and they can be used both for classification and regression problems. **(Harrison, 2018)**. However, for this project, we will focus on the classification aspect of it.

In KNN objects are classified into a class, based on the modal votes of their 'k' nearest neighbours. During the training phases, the objects being the input data are represented as

data points in an n-dimensional space **(Ananthi & Sathyabama, 2009)**. The neighbours are taken from training data, where the objects have been classified correctly with their class labels **(Ananthi & Sathyabama, 2009)**. The principle behind it is that objects that belong in the same class exist in close proximity **(Harrison, 2018)**. Therefore, it focuses on the similarity and closeness between the data points. To identify the proximity between the objects, the distance is calculated. For this project, the distance I will use is Euclidean Distance however Harmonic or Manhattan is also an option. **(Ananthi & Sathyabama, 2009)**

The general coding implication of the data referenced from Onel Harrison is as follows:

1. Load the training data
2. Initialize K to your chosen number of neighbors
3. For each testing data, calculate distance between that and the training data
4. Add the distances and the index of sample data in a ordered collection
5. Sort from smallest
6. Pick the first K entries form the sorted collection
7. Get the labels of the selected entries
8. Return the mode of the K labels.

The algorithm is computed by selecting a '$\boldsymbol{k}$' value which is the nearest neighbour to the input data waiting to be classified. For instance, if $\boldsymbol{k = 3}$, the algorithm will find 3 nearest neighbors to the data point **(Dwivedi, 2020)**. The point will be assigned to the class with the shortest distance. Selecting the right K value is vital to avoid any errors. If the K values are too small then the amount of meaningless information or noise will affect **(Ananthi & Sathyabama, 2009)**. Thus with a larger K value, the noise is reduced, however, the

boundaries between classes are less distinct and essentially destroy the principles behind the classifier. **(Ananthi & Sathyabama, 2009)** Hence selecting the right K values is achieved through cross-validation, where we can perform the algorithm several times on different K values and observe which returns the most accurate results. The general observation is that if we decrease K, our prediction becomes less stable, hence increasing K to a certain point creates a more accurate prediction **(Dwivedi, 2020)**.

The advantage of KNN is that it is a simple and easy-to-understand algorithm. Furthermore, it's very versatile being used for both classification and regression. Additionally, the accuracy of the algorithm is generally high. However, the disadvantages are that it requires high storage as it stores all the training data. In addition to this, the prediction rates are slow. **(Dwivedi, 2020)**.

Chapter 3 Methodology

This chapter will discuss the methodology, I used for my research, inspired by Olga Belitskaya Project. I have used the programming software Jupyter Notebook and the coding language Python. As discussed the classifiers used are Naïve Bayes Classifier and K- Nearest Neighbours. For this experiment, controlled measures were introduced to ensure that my results are reliable and as fair as possible when comparing these classifiers. First, the emails/data collected in this investigation are all in the English Language, I use the same training data set to train both my classifiers and the same data set to test my predictions.

3.1 Constructing the code

To train my classifiers I required a large database, therefore I resorted to using one online which has 5171 emails filtered as either Spam or ham. This was provided by: **(Kaggle, 2021)**. Of the data set, 1499 were spam and 3672 were non-ham emails. My testing data was the first 50 emails from my inbox, from these I labelled 7 of them as Spam emails and the remaining 43 as ordinary emails.

Pre-Processing my Data

Before applying the classifier to my training set, I created a bag of words which is a set of all the words in the training set. This is achieved through the CountVectorizer code. Each word is treated individually and the frequency of how often it turns up is also recorded. There are three benefits to the CountVectorizer code. Firstly it decapitalizes all the words in the training set. Secondly, a token parameter is in-cooperated which disregards any punctuation marks, and finally, it eliminates any stop words. These are words that frequently appear in the English language such as 'am', 'it'. Therefore 'Happy!' would be treated as 'happy'. **(Belitskaya, n.d.)**

The bag of words is returned as a matrix, where each row is the email in the training and the column corresponding to the frequency of the token (word). Due to the extensive size of the training data set, Jupyter Notebook can return a fraction of the matrix which I have attached below.

Creating the Training Data

After creating my bag of words, I have split up the training data into two subsets:

- X_ Train is the content of the training data
- Y_ Train being the label of the training data.

I have done the same for the Testing Data as follows:

- X_Train is the content of the testing data
- Y_Test is the Label of the testing data

Applying the Classifier on the Training Data.

After splitting the data accordingly, using the bag of words I will convert the training data (X_Train specifically) into a matrix of word frequency. The row corresponds to the emails and the columns to the frequency of each word in the email. The same has been done to the X_test subset. .Now, we can apply the classifiers to the training data. There are two ways in which I could have implemented both of the classifiers, either through scratch or by implementing it using scikit-learn. For this project, I have chosen to conduct it through sckit-learn. **(Belitskaya, n.d.)**

Firstly for the Naïve Bayes classifier, I used the code MultinomialNB code. This code has specifically been constructed for discrete-valued classification problems similar, which is well suited for filtering spam emails. Using the .fit(), will allow me to fit the training data into the classifier and test this against the training set. **(Belitskaya, n.d.)**

This entire process is similar to training my data on the K-Nearest Neighbour algorithm via the KNeighboursClassifer code. Furthermore, through cross-validation, the most suitable K value is 5. **(Belitskaya, n.d.)**

Prediction on my inbox

The final step performed is running the prediction on my testing data set, which includes the first 50 emails from my inbox.

3.2 Visualizing the data

Inspired by Tejan Karmali, using WorldCloud, I was able to create an image that illustrates the most repeated words in the spam messages of both training and testing sets. The modal words in the training set are depicted by the font size in the image. Figure 1 corresponds to the spam messages of the testing data set. As we can see, in the spam training data set words such as ‘free’, ‘call’, ‘prize’, ‘won’ etc. I have done the same for my testing data set shown in Figure to see if there is a clear visual observation of any words they share in common. Unsurprisingly, these emails contained similar words to the prior data. From, this I can hypothesis that the classification models may predict the email with the words these words as spam. **(Karmali, 2017)**

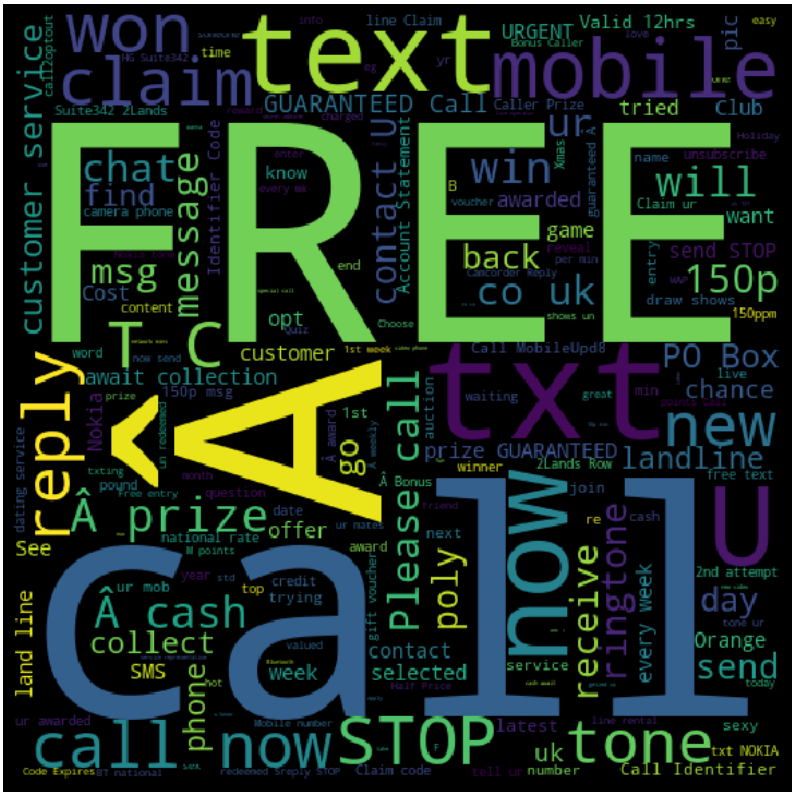

Figure 1 WordCloud of spam words in training data

Figure 2 WordCloud of spam words for testing data

Chapter 4 Results

4.1 Measure to evaluate results

To assess the effectiveness of my model I have evaluated four parameters which are, accuracy, precision, recall, and F1. The objective of this model was to successfully identify unsolicited emails or Spam emails and regular emails and categorise them accordingly. In addition, to this, we need to identify whether the email, the classifier labelled as spam matches with the original labels defined by myself.

To evaluate any detection system, we need to construct a contingency table, where each cell presents a possible outcome. **(Belitskaya, n.d.)**

Table 1 Contingency Table

Contingency Table of possible outcomes		Original label	
		Original Positive	Original negative
Classifier outputs	Classifier positive	True positive	False positive
	Classifier negative	False negative	True negative

The four combinations available are True positives, which correspond to the emails which the classified labelled correctly as spam. Similarly, true negatives correspond to the emails the classifier labelled correctly as non-spam. These match with the original defined labels. Therefore, False positives are the emails the classifier labelled as spam, however, were not. And similarly, false-negative correspond to the emails the classifier labelled as non-spam but were indeed originally defined as non-spam. **(Anon., n.d.) (Belitskaya, n.d.)**

Using the above outcomes defines the parameters I will be using to evaluate my model. These are accuracy, precision, recall, and F1.

Accuracy

Accuracy is the measure of how frequently the classifier makes the correct prediction. This is the ratio of the frequency of correctly labelled predictions and the total number of input data the testing data. Accuracy score is out of 1 (or 100%), therefore the closer the score is to 1, the higher the number or correct predictions. **(Belitskaya, n.d.) (Anon., n.d.) (Joshi, 2016)**

Equation 9: Accurcay

$$Accuracy = \frac{True\ positives + True\ negatvies}{(True\ positive + False\ negatives + True\ negatives + False\ postives)}$$

Precision

Precision solely focuses on spam emails. It defines the frequency of emails that were defined correctly as spam by the classifier against the frequency of emails of all the emails labelled as spam by the classifier, regardless of whether if the prediction was correct or not. **(Belitskaya, n.d.) (Anon., n.d.)**

Equation 10: Precison

$$Precision = \frac{True\ positives}{(True\ positive + False\ positives)}$$

Recall

Recall measures the ratio of the frequency of emails that were correctly labelled spam against all the emails that were originally defined as spam. **(Joshi, 2016)**

Equation 11: Recall

$$Recall = \frac{True\ positives}{(True\ positive + False\ negatvies)}$$

F1-Score

The F1 score measures the weighted average of Precision and Recall. This value takes into consideration both false positives and false negatives. Some may argue that it is more useful than accuracy, especially with a skewed class distribution. This can applies to my case as less than 15% of my inbox are spam labels. **(Joshi, 2016) (Belitskaya, n.d.)**

Equation 12 F1

$$F_1 = 2 \cdot \frac{\text{Preicsion} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{\text{True positvies}}{\text{True Positvies} + \frac{1}{2}(\text{False Positives} + \text{False Negatives})}$$

4.2 Final Predictions

The final results are shown in Figure 3, which displays the full score for each measure. I will evaluate my score as a percentage and to the first nearest decimal place.

Evaluate both of the KNN and Naive Bayes Model

naive_bayes_predictions

In [26]:
```
print('Prediction of Naive Bayes')
print('Accuracy score: ', format(accuracy_score(y_test, naive_bayes_predictions)))
print('Precision score: ', format(precision_score(y_test, naive_bayes_predictions)))
print('Recall score: ', format(recall_score(y_test, naive_bayes_predictions)))
print('F1 score: ', format(f1_score(y_test, naive_bayes_predictions)))
```

```
Prediction of Naive Bayes
Accuracy score:  0.5833333333333334
Precision score:  0.7272727272727273
Recall score:  0.5333333333333333
F1 score:  0.6153846153846153
```

knn_predictions

In [27]:
```
print('Prediction of K nearest Neighbour')
print('Accuracy score: ', format(accuracy_score(y_test, Knn_pred)))
print('Precision score: ', format(precision_score(y_test, Knn_pred)))
print('Recall score: ', format(recall_score(y_test, Knn_pred)))
print('F1 score: ', format(f1_score(y_test, Knn_pred)))
```

```
Prediction of K nearest Neighbour
Accuracy score:  0.625
Precision score:  0.625
Recall score:  1.0
F1 score:  0.7692307692307693
```

Figure 3 Results of classifiers

The Naïve Bayes Classifier has shown a 58% accuracy score. Whereas K- Nearest Neighbours shows a higher accuracy score of 62.5%. However, the precision score for Naïve-Bayes Classifier is 72.5% which is greater than K-Nearest Neighbours being 62.5%. The recall score for the Naïve Bayes Classifier is 53.3%, whereas for K-Nearest Neighbours is recall score is 100%. And finally, the F1 score for Naïve Bayes is 61.5%, while K-Nearest Neighbours produced a higher F1 score of 76.9%

Chapter 5 Conclusion

From my results, I can infer that K-Nearest Neighbours presents to perform better as a Spam detection system, for my testing data. Out of the four measures evaluated, three of them returned a higher percentage. Surprisingly, the Naïve Bayes classifier had underperformed, despite much research concluding it as the well-suited classifier for spam detection.

Chapter 6 Discussion and Closing Remarks

This project explored spam detection systems, through supervised machine learning algorithms to automatically filter out any unsolicited emails from an individual's inbox. The two classifiers I had chosen to compare were K-Nearest Neighbours and Naïve Bayes. With thanks to Olga Belitskaya framework, I was able to create a code to successfully perform these classifiers. The proposed models provided a good theory of how the classification place. Additionally, through my research, there has been supporting evidence for the Naïve Bayes classifier to be the most well suited for spam detection. However, my final results display that out of the two proposed models, K-Nearest Neighbours proved to be a well-suited classifier for the data set I had.

As this is a new and developing field, thus over the years, stronger and more accurate classifiers have been created from a combination of supervised learning classifiers to successfully detect these unsolicited emails.

Bibliography

Ananthi, S. & Sathyabama, S., 2009. *SPAM FILTERING USING K - NN,* s.l.: s.n.

Anon., 2021. [Online]
Available at: https://www.kaggle.com/databeru/spam-classifier-model-comparison-accuracy-97?scriptVersionId=59368486
[Accessed 2021].

Anon., n.d. *Classification: Accuracy.* [Online]
Available at: https://developers.google.com/machine-learning/crash-course/classification/accuracy
[Accessed 2021].

Anon., n.d. *Oxford Dictionary.* [Online]
Available at: https://www.lexico.com/definition/sentiment_analysis
[Accessed 03 2021].

Belitskaya, O., n.d. *Machine Learning Engineer Nanodegree.* [Online]
Available at: https://olgabelitskaya.github.io/MLE_ND_PP0.html
[Accessed 2021].

Boehmke, B., n.d. *UC Business Analytics R Programming Guide.* [Online]
Available at: https://uc-r.github.io/naive_bayes
[Accessed 2021].

Dwivedi, R., 2020. *How Does K-nearest Neighbor Works In Machine Learning Classification Problem?.* [Online]
Available at: https://www.analyticssteps.com/blogs/how-does-k-nearest-neighbor-works-machine-learning-classification-problem
[Accessed 2021].

Harrison, O., 2018. *Machine Learning Basics with the K-Nearest Neighbors Algorithm.* [Online]
Available at: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761
[Accessed 2021].

Hassan, A., Medhat, W. & Hoda, K., 2014. *Sentiment Anaylsis algorthms and applications: A survey,* s.l.: s.n.

Joshi, R., 2016. *Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures.* [Online]
Available at: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/
[Accessed 2021].

Jurafsky, D. & Martin, J. H., 2019. *Speech and Language Processing,* s.l.: s.n.

Kaggle, 2021. *Spam Classifier with MultinomialNB.* [Online]
Available at: https://www.kaggle.com/databeru/spam-classifier-model-comparison-

accuracy-97?scriptVersionId=59368486
[Accessed 2021].
Karmali, T., 2017. *Spam Classifier in Python from scratch.* [Online]
Available at: https://towardsdatascience.com/spam-classifier-in-python-from-scratch-27a98ddd8e73
[Accessed 2021].
Liu, B., 2012. *Sentiment Analysis and Opinion Mining.* s.l.:s.n.
McNamara, J., Green, R. F. & Olsson, O., 2006. *Bayes' theorem and its applications in animal behaviour,* s.l.: s.n.
Mesevage, T. G., 2020. *Machine Learning Classifiers - The Algorithms & How They Work.* [Online]
Available at: https://monkeylearn.com/blog/what-is-a-classifier/
[Accessed 2020].
Mukherjee, A. et al., 2013. *Spotting Opinion Spammer Using Behavioral Footprings,* s.l.: s.n.
Murphy, K. P., 2006. *Naive Bayes classifiers,* s.l.: s.n.
Singh, A., Thakur, N. & Sharma, A., 2016. *A Review of Supervised Machine Learning Algorithms,* s.l.: s.n.
Terekhov, A., n.d. *Online Anonymity: The Pros of Being Anonymous Online.* [Online]
Available at: https://www.hotspotshield.com/blog/pros-anonymous-online/
[Accessed 2021].
Vadapalli, P., 2021. *Naive Bayes Explained: Function, Advantages & Disadvantages, Applications in 2021.* [Online]
Available at: https://www.upgrad.com/blog/naive-bayes-explained/
[Accessed 2021].