Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Morten K. Amundsen

Automated Triaging and Remediation of User Incident Reports with Supervised Machine Learning and Threat Intelligence

Master's thesis in Information Security
Supervisor: Basel Katt, Håkon Olsen
June 2020

NTNU
Norwegian University of Science and Technology

sportradar

Morten K. Amundsen

Automated Triaging and Remediation of User Incident Reports with Supervised Machine Learning and Threat Intelligence

Master's thesis in Information Security
Supervisor: Basel Katt, Håkon Olsen
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Abstract

Malicious emails are increasingly problematic for organizations, and despite several tools being available to prevent many attacks, they are becoming increasingly sophisticated. With all the malicious emails that manage to get past the existing defense mechanisms, an organization relies on the awareness of their employees to prevent a potential catastrophe. When an employee suspects that an email is malicious, they commonly report their suspicions to the information security team, which in turn assesses the incident report and provides feedback notifying the employee of their findings. The task of assessing and acting upon these reports can be automated using machine learning, which will reduce the resources spent on these investigations, allow for rapid feedback to the user potentially increasing their security awareness, and provide metrics for upper management and improvement of security awareness training.

Acknowledgements

First and foremost, I would like to express my sincere gratitude towards my supervisor and mentor at Sportradar, Håkon Olsen, for guidance, interesting discussions, his eagerness to help and ability point me in the right direction. Secondly, I would like to thank my supervisor at NTNU, Basel Katt, for his support, excellent guidance, and feedback during the project work. I would also like to thank my father, Finn, for valuable discussions and helping me stay motivated.

Finally, I would like to thank my family and friends for their continuous support throughout this project. This is for you.

Preface

This Master's thesis was completed during the spring of 2020 at the Norwegian University of Science and Technology (NTNU), Gjøvik. The thesis was initiated by and completed in cooperation with Sportradar AS. It was completed as part of the Information Security program on the Management track. The intended audience for this thesis is anyone interested in email classification, automation and/or machine learning, as well as organizations interested in inspiration on how to automate the process of handling user incident reports.

Trondheim, 31st May 2020

Contents

Figures

Tables

Code Listings

Chapter 1

Introduction

1.1 Topics Covered by the Thesis

Emails with malicious intent have long been a major problem for organizations, where some organizations receive thousands of such emails every day. The information security team, commonly the point of contact for user incident reports, spends significant time and resources investigating these reports, which usually range from spam or attempted fraud, to malware, in addition to false reports. The various types of malicious email are described in Section 2.1.2.

These tasks can be automated using machine learning where reports can be processed and categorized based on entity extraction of the email contents. This will benefit an organization by freeing information security resources and improving security by providing metrics and increasing security awareness among employees. It also has the potential to radically speed up incident response time.

In this project, we will study the effectiveness and efficiency of using machine learning classifiers for automated remediation of user incident reports of malicious emails. We will use a combination of Natural Language Processing, a Machine Learning classification algorithm and threat intelligence for detection and classification of malicious email. We will also look into the effect of rapid feedback on security awareness, investigate whether it is possible to generate metrics representing trends in malicious email, and evaluate the cost-effectiveness of the solution compared to manual labor.

1.2 Keywords

The following keywords refers to the IEEE Computer Society's S1M Taxonomy [1].

- K.6.m.b Security
- I.2.6.g Machine learning
- I.2.7 Natural Language Processing
- I.5.2.b Feature evaluation and selection
- I.5.4.n Text processing

- D.2.8 Metrics/Measurement
- K.4.3.a Automation

1.3 Problem Description

Over the last couple of decades, several approaches to classification of email have been proposed in research. There has been a multitude of proposed approaches to email filtering, including rule-based solutions based on e.g. blacklisted words or the sender's country of origin [2]. In more recent years, solutions applying machine learning classification algorithms as a means of identifying malicious email have become more common, with promising results such as presented in Fang et al. (2019), Akinyelu et al. (2014), and Cohen et al. (2018).

However, a large part of these designs have been developed towards classifying emails as either *good* or *bad*. Thus, these approaches treat harmful phishing emails and emails containing malware equally to relatively harmless spam. Without the ability to determine which attack vector is currently ongoing, and not being able to see the trends in these attacks, an organization will have a hard time identifying what their security awareness training should be targeted towards. Additionally, a significant amount of these solutions have been trained on data sets that are outdated, many of which were collected as far back as the late 1990's to mid 2000's, such as in Song et al. (2009), Li et al. (2015), and Jiang (2010). Malicious emails, and the actors behind them, have become increasingly sophisticated in their methods, and we see that the trends, especially in phishing, fraud, and distribution of malware, changes continuously. A report published by APWG in May 2019 states that 58 percent of phishing sites now uses HTTPS [9, pp. 6], hence making the HTTPS protocol's *green lock* formerly identified as an indication of security and trust a misleading assurance. Accordingly, the email filtering and classification systems need to follow these trends, which requires that the data used to train them are contemporary.

1.4 Justification, Motivation and Benefits

Emails with malicious intent have been around for a long time, and still remains a threat that every organization should be aware of and take action towards. According to the 2019 Trustwave Global Security Report, spam (including malware, phishing and other types) accounted for 45% of all inbound email in 2018. [10, pp. 28]. Further, the report states that one of the most common methods of compromise in 2018 was phishing and other social-engineering techniques, accounting for 46% of the breaches on corporate/internal networks [10, pp. 25], while a report published by IBM Security states that "inadvertent insiders" compromised by phishing attacks or stolen/infected devices account for about a quarter of the breaches. [11, pp. 7]

As stated in [11, pp. 3], the global average total cost of a data breach is USD

$3.92 million, where the average size of a data breach is 25,575 records. Thus, there is a significant financial motivation for decreasing the amount of malicious email successfully reaching employees.

The Cofense Phishing Defense Center (PDC) receives and analyses emails reported by around 2 million users globally. [12, pp. 4–5] states that more than 50 thousand (of around 1.5 million total) of the emails they received and analyzed in 2018 were credential stealing attacks, and further mentions that "[...] it should be remembered that every one of the emails received by the PDC has bypassed some form of automated analysis by a secure email gateway or other inline threat scanning tool." [12, pp. 4-5]. Even though additional measures exist, such as blacklisting of known-bad websites in DNS/firewalls or URL sandboxing, which may provide protection even after a user clicks a link, there is no catch-all solution.

Although current phishing defense mechanisms do a good job at stopping a large amount of the malicious emails received, it's clear that there's still a significant amount that bypasses these defense mechanisms, making an organization depend on the awareness of their employees to detect and report the remaining threats.

Currently, the investigation upon user incident reports takes a significant amount of time and resources for organizations, where issues include slow feedback rates to the reporting users and lack of a good solution to generate metrics (*Key Performance Indicators*) with ease. The goal of this project is to automate this time-consuming task. We want to free important resources in the information security team and save time and costs, as well as generate metrics for management, lowering the costs of work related to the processing of user reports and improving response time and security for the organization.

1.5 Research Questions

1. Will rapid feedback to user incident reports increase the security awareness of employees?
2. Is it feasible to use serverless functions for the automation of these tasks?
3. How does preprocessing impact the accuracy of the machine learning classification algorithms tested?
4. Is it feasible to implement a credibility indicator based on previous reports by the same user, in accordance with the GDPR?
5. Is it possible to automate the generation of metrics representing trends in email attack methods in order to provide relevant security awareness training?

1.6 Planned Contributions

This thesis proposes an approach to automated remediation of user incident reports, achieved through classification of emails with the ability to distinguish between different categories of malicious email. Upon these results, automated actions can be taken based on the confidence level of the output, which will allow for rapid feedback to users reporting incidents, generation of metrics that may be valuable for upper management and provide insights into the current threat landscape related to email, lowering the costs of work related to processing of these reports and improving the security in the organization.

We also hope that the solution will help increase security awareness of employees through the rapid feedback. The project could potentially also be used as a means of mitigating malicious email before it reaches the end-user if implemented as part of the email gateway system.

1.7 Thesis Structure

The thesis is organized as follows:

- **Chapter 2: Background and Theory** – Provides an introduction to the main concepts, technologies, and algorithms that make up the basis of the thesis.
- **Chapter 3: Related Work** – Presents related research and state-of-the-art on the topics.
- **Chapter 4: Methods** – Summarizes the research methodologies used in this thesis, followed by a description of the data set used.
- **Chapter 5: Results** – Presents the results from the research.
- **Chapter 6: Discussion** – Presents our analysis of our findings from the experiments.
- **Chapter 7: Future Work** – Our propositions for future work.
- **Chapter 8: Conclusion** – Concludes the thesis based on the discussion.

Chapter 2

Background and Theory

The purpose of this chapter is to provide the reader with an overview of the main theoretical concepts that are essential for understanding the rest of the thesis. We begin with an introduction to emails in the context of this thesis, before introducing the main concepts of the field of Machine Learning (ML) with a focus on classification. Further, we provide a brief explanation of Natural Language Processing (NLP) and entity extraction. Finally, we describe threat intelligence and its application in this project, and conclude the chapter with a description of how these technologies are combined to make up the project.

2.1 Email

2.1.1 Weaknesses

Email has, like many of the technologies we use today, some design flaws that might increase the risk for users. This is not necessarily just a flaw in the design of the protocol, but also an issue in most email client applications. An example of this is the way that the sender of the email is presented, in that one can specify virtually any email address as the *from* address (the sender of the email).

The protocol is designed so that the sender field consists of either an email address, such as "winston.smith@mail.com", or a combination of a name and an address, e.g. "Winston Smith <winston.smith@mail.com>". Thus, one can specify a name associated with an email address. This name can be anything the sender wishes it to be, such as "Facebook", upon which the recipient will see the sender as "Facebook <winston.smith@mail.com>". The issue, however, is that the protocol also allows one to modify the sender address presented between the "<" and ">" characters, so that one could make an email look like its sent from e.g. "Facebook <no-reply@facebook.com>", despite not owning or having access to the domain "facebook.com". Thus, without further inspection, it would look as though the email is indeed sent from a legitimate Facebook domain. This technique is known as *email spoofing*[1].

[1] `https://en.wikipedia.org/wiki/Email_spoofing`

The reason why this is possible is due to the lack of authentication mechanisms in the core email protocols. Although several potential solutions have been created over the years, including *SPF*, *DKIM*, and *DMARC* (see Appendix A), there has been a lack of traction, and they require correct configuration of all involved systems, including the sending domain and their legitimate mail servers, and the receiving system, in order to work properly [13]. In other words, a domain can be spoofed unless two criteria are met:

- The domain's owner must have an authentication mechanism in place
- The recipient's mail server must be configured to validate the sender's address

Thus, by modifying the *from* field in the email headers, one can impersonate virtually any company or domain, or at least so on the immediate surface visible to the recipient(s). Unfortunately, most email clients (although not all) blindly trust the contents of the *from* header, and presents that value to the user. In order to determine the address and domain of the true sender, one must manually inspect the value of the "envelope-from", which is found in the email headers – in other words, not a task that most people know how to.

Although there were legitimate uses for forged sender addresses in the earlier days of the Internet, email spoofing is now a technique commonly used by actors with malicious intent such as for phishing and business email compromise. With a forged sender address and a well-designed email, there is a good chance that one could trick a potential victim into believing that the email is legitimate.

2.1.2 Types of Malicious Email

This subsection provides a brief introduction to the different types of malicious email that we'll focus on in this thesis project. There are more types and subtypes of each, but this section provides a general overview of each. It should be mentioned that the types are not mutually exclusive, where e.g. an email could be a phishing email containing malicious attachment.

Spam Spam is a rather broad category within the topic of malicious email, where the contents and identifying traits can vary vastly. Spam is generally malicious email sent out in bulk to tens, hundreds, or even thousands of recipients, in the hopes that a portion of the recipients will fall for it. Some examples of spam include the well-known "Nigerian prince" messages, and more recently the *Bitcoin* emails promising recipients wealth and great fortune. Spam is also frequently used to deliver malicious attachments, such as *ransomware* or other types of malware.

Phishing Phishing is one of the more dangerous types of malicious email, where the objective of the sender is to obtain sensitive information from a target, such as credentials (e.g. usernames, passwords) or banking details (e.g. credit card details). This type of *attack* has grown significantly over recent years. [9, pp. 3]

Additionally, we see that malicious actors attempt to take advantage of situations such as the currently on-going *COVID-19* outbreak, exploiting people's health concerns and the increase in employees working from home [15, 16].

Where older attempts at phishing often contained poor grammar and suspicious links, it seems the adversaries have become increasingly sophisticated – including improved spelling, increased use of HTTPS [14], and *typosquatted domains*[2], where a legitimate domain such as "facebook.com" is typosquatted with a domain such as "faceboook.com", making it hard to spot at a first glance.

This type of malicious email is especially dangerous within an organization, where an adversary upon a successful phishing campaign could gain access to a high-privileged account and gain a foothold within internal systems and networks.

Within phishing, there's a few subcategories such as spear-phishing, where the email is highly targeted towards an individual or organization; whaling, where the attacker uses spear-phishing to target high-profile targets or executives within an organization; as well as clone-phishing, where a legitimate, previously delivered email is cloned with its links or attachments replaced with malicious ones.

Fraud Fraudulent emails are a common threat to companies, where some have managed to fraud companies and individuals of significant amounts of money. This type of email is commonly themed around falsified invoices from a contractor of a company, such as described in [17]; or gift card scams where the adversary impersonates e.g. an executive in a company, asking an employee to purchase gift cards, as in [18].

Malware The last type we'll discuss in this subsection is the malware email type, which includes a link to a file or an attached file. This file is commonly disguised as a legitimate file, such as a financial report, but hides malware intended to e.g. infect a user's computer. This type of malware also poses a great risk, as an infected machine within an organization could be capable of causing great harm.

2.2 Machine Learning

Machine Learning (ML) is a part of the field of Artificial Intelligence (AI), and is in its base form an algorithm which builds a mathematical model based on sample data, known as "training data". By providing the algorithm with a data set, it can learn to recognize/detect patterns, and thus be able make predictions or decisions without the need of explicitly programming it to do so.

Within the field of Machine Learning, there are several types of learning algorithms. Two types commonly used are *supervised learning* and *unsupervised learning*. The main difference between these two types of learning algorithms is that supervised learning utilizes pre-labeled data, while unsupervised learning

[2] `https://en.wikipedia.org/wiki/Typosquatting`

does not. By pre-labeled data, we mean data that is labeled with a desired outcome, where the task of labeling the *training data* usually is performed manually by a human-being. With supervised learning, the algorithm calculates (predicts) an outcome, compares the result with the desired outcome, before finally adjusting or correcting its prediction. Unsupervised learning, on the other hand, takes a set of unlabeled data, where it attempts to identify commonalities in the data and group them.

2.2.1 Classification

Classification is a subgroup of supervised learning where the goal is to assign a classification label to some input. The classification labels are a set of two or more defined labels. Classification can be divided further into two types; binary classification (i.e. either *zero* or *one*), and multiclass classification (i.e. more than two classification labels). For example, binary classification can predict whether an email is spam or not (*yes* or *no*), while multiclass classification can have multiple outputs. For instance, it can be used to determine the type of email (e.g. *legitimate/solicited*, *spam*, *phishing*, etc.).

In order to train the classification algorithm (also called a *classifier*), a data set consisting of several values along with a classification (known as a "label") of each is used. In the context of this thesis, a sample could be the contents of a phishing email along with a label representing the type *phishing*, as presented in Table 2.1 below. The category representing each label has been added for reference.

Table 2.1: An example of a data set used for training a multiclass classifier.

Label	Content	Category
0	hi helen, thanks for reaching out! [...]	*legitimate*
1	free bitcoin today [...]	*spam*
2	click this link to log in to facebook [...]	*phishing*
3	please order gift cards [...]	*fraud*
4	check out the attached report [...]	*malware*

By using the data presented in the table above as input (granted we have a higher number of instances per label), we can train the classifier to predict the category of a new email. The classifier solves this problem through statistics, where it attempts to identify which category has the highest probability of containing the words found in the new email. For instance, if the words "free" and "bitcoin" were found in the training data and labeled as spam, it is likely that any new email containing these two words is spam as well.

There are several well-known classifiers, including:

- Naïve Bayes Classifier[3]
- Support Vector Machines[4]

[3] `https://en.wikipedia.org/wiki/Naive_Bayes_classifier`

[4] `https://en.wikipedia.org/wiki/Support-vector_machine`

- Decision Trees[5]
- Neural Networks[6]

In this thesis, we will be using the Naïve Bayes Classifier in order to classify emails using a multiclass algorithm, where the classification labels are *legitimate*, *spam*, *phishing*, *fraud*, and *malware*. In the following subsection, we will explain the process of classification.

2.2.2 The Process of Classification

This section describes a high-level process that is common for all classification problems, as shown in Figure 2.1 below.

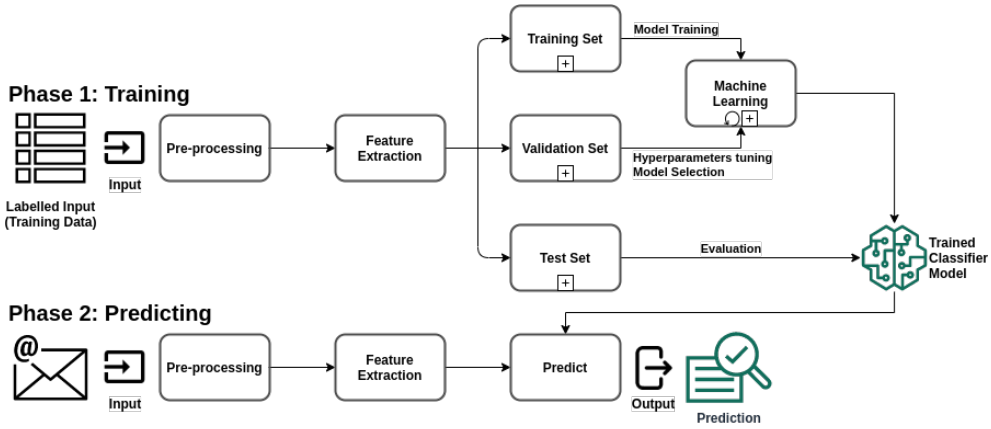


Figure 2.1: The classification process.

Phase 1: Training

The initial phase in the classification process is the task of training a classifier, i.e. to provide the classifier with input (the training data) so that it can learn to perform a classification task. In general, the training data may come in a multitude of different formats, such as plain text, an image, or virtually any digital data. Since this project uses the *supervised learning* technique, we need to input some data with accompanying labels which represent the expected output class. In this project, we use email messages as the input, where the labels range from 0 through 4, each representing a category as presented in Table 2.1 above.

Pre-processing The first step of this phase is to perform *pre-processing* on the input, where the data is "cleaned" in order to produce homogeneous values. For example, the email message files used for this project contains a large amount of values representing the email, including its subject, contents (body), and email

[5] `https://en.wikipedia.org/wiki/Decision_tree_learning`
[6] `https://en.wikipedia.org/wiki/Artificial_neural_network`

headers. Since most emails today utilize HTML formatting, a subtask of the pre-processing process is to remove HTML tags and other non-textual elements in order to get the plain text version of the contents. This is done so that we can perform a word count on the plain text found in the email, to generate statistics on the words most commonly found for the different categories we want to classify.

Feature Extraction The second step is to perform *feature extraction* on the data set. Feature extraction is the process of transforming raw data into features that are suitable for modeling – in this thesis work, we compare two methods: word vector counts and term frequency–inverse document frequency (TF-IDF)[7]. Word vector counts calculates the frequency of each word in a document, while TF-IDF is a statistical method used to assess the importance of a word to a document in a collection. This step builds derived values (features) from the input, which in this case is the body (text contents) of the emails, before finally outputting a reduced set of features known as a feature vector.

Splitting the Data Set The next step involves splitting the data set into three separate parts: one set for training, one set for validating, and one for testing. The *training set* is used for training the machine learning model, whereas the *validation set* is used to tune and validate the model. The third data set, the *test set*, is used to evaluate the resulting classifier model in order to determine its accuracy.

The former two data sets are fed into the learning algorithm, where each entry in the set increases the experience of the classifier, possibly improving its classification accuracy. The output of this phase is a trained classifier model which is used in the next phase.

Phase 2: Predicting

The second phase of the classification process is similar to the *training phase* presented above. However, instead of using a set of data as input, we use a single email message as the input, i.e. the email that we wish to classify. The input no longer includes a label, as it is the task of the classifier to predict and produce the label for this email. Further, we perform the pre-processing and feature-extraction as described in *training phase*. It's important that we perform these two steps in the same way as they were done in the training phase. If these steps are different, or skipped entirely, the classifier will not perform as intended or may not work at all.

The final step of this phase is to use the trained classifier model (the output from the training phase) to predict the category of the new email.

[7] `https://en.wikipedia.org/wiki/Tf-idf`

2.2.3 Measuring Performance

In order to determine the accuracy and precision of our classifier, we need a method for measuring the performance. Since we will be experimenting with the parameters and options of the classifier, we will compute several metrics which can be used for comparison. To ensure that the metrics collected are consistent across experiments, we will build functions that take care of collecting/generating the measurements. These implementations will be built on the built-in functionality of the *sklearn* Python library.

Learning Curves

A learning curve shows the validation and training score for a classifier. It is used to find out how much we benefit from adding additional training data, and to identify whether the classifier suffers from a bias error or variance error. [19]

Classification Report

– Precision The precision score can be described as the classifier's ability not to label a negative sample as positive. It is the ratio of $tp/(tp + fp)$, where *tp* is the number of True Positives and *fp* is the number of False Positives.

– Recall The recall score can be described as the classifier's ability to find all positive samples. It is the ratio of $tp/(tp + fn)$, where *tp* is the number of True Positives and *fn* is the number of False Negatives. [20]

– F1 The F1 score, also known as a balanced F-score, is a measure of a test's accuracy; it is a weighted average of the precision and recall.

– Support The support is the number of occurrences of each class in the test data set.

Cross Validation

Cross validation is used to assess how the results of a classifier will generalize to an unknown data set – e.g. a real world data set. When training the classifier, we normally split the data set into two parts; a training set and a testing set.

With cross validation, we split the training data *k*-fold, resulting in *k folds*. For instance, we could split a data set using 5-fold cross validation, where we use $k - 1$ folds to train the classifier, while the remaining folds are used to validate the model. The performance measure is then the average of the values computed for each fold in the validation set. [21]

Confusion Matrix

The confusion matrix, also known as an error matrix, is used to visualize the performance of the classifier's predictions. The matrix consists of two axes – *Predicted label* on the X-axis, and *True label* on the Y-axis – which represent the instances in a predicted class and the instances in a true class, respectively.

In other words, the confusion matrix visualizes each class and the predictions, so that we can see the number of correct and incorrect predictions for each class.

2.3 Natural Language Processing

Natural Language Processing (NLP) is another branch of the field of Artificial Intelligence (AI), and is concerned with the interaction between human (natural) language and computers. Examples of uses for NLP is speech recognition, natural language understanding, and natural language generation. [22]

With NLP, the goal is to help computers understand, interpret, and manipulate human language, since computers natively use machine code – a language essentially consisting of one's and zero's. In this section, we'll focus on a subbranch of NLP, known as *Entity Extraction*.

2.3.1 Entity Extraction

Entity Extraction, also known as Named-entity Recognition (NER), is a task that aims to locate and classify named entities from unstructured text, categorizing them into defined categories such as person names, locations, organizations, etc. In this project, we utilize Entity Extraction in order to extract important entities from the textual contents of the emails, so that we can reduce the unstructured text as a part of the pre-processing task of the Machine Learning classification process, as explained in Section 2.2.2.

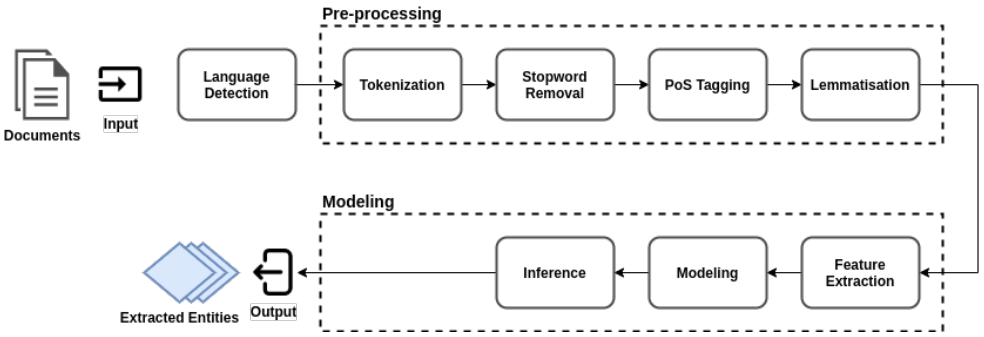


Figure 2.2: The Entity Extraction Process.

The entities that we wish to extract for our project are person names, important nouns, locations, and dates, all of which will form the processed text that is fed into the Machine Learning classifier model.

Pre-processing

Similarly to the classification process described in Section 2.2, we perform pre-processing on the input. The following describes the steps that take place in this task.

Tokenization The first step is to perform *tokenization*, which is the task of *chopping up* a text into pieces called tokens. I.e., splitting a larger body of text into smaller lines or words. In this project, we split full sentences into words so that each word can be processed individually.

Stopword Removal The next step of the pre-processing is to perform *stopword removal*, a task in which we remove *stopwords* (e.g. "the", "a", "for", etc.) and other characters that indicate punctuation, such as "." (period), "!" (exclamation mark), etc.

PoS Tagging PoS tagging, short for Part-of-Speech tagging, is the task of labelling each word in a sentence with its appropriate part of speech. In other words, PoS tagging allows us to tag each word as either a noun, verb, adjective, etc., based on its context and definition. This step is necessary in order to properly perform the next step, which is lemmatisation.

Lemmatization The final step of the pre-processing is to perform *lemmatization*, which is the process of reducing a word to its base form. For grammatical reasons, text will contain different forms of a word, such as *listen*, *listens*, and *listening*. Since our ML classifier relies on the word counts in email's text, we perform lemmatization to reduce each word to its base form.

Stemming Stemming is similar to lemmatization in that its goal is to return a word to its base form; where the difference is that stemming commonly chops off the end of a word, while lemmatization uses a vocabulary and morphological analysis of the word [23, pp. 32].

Once the pre-processing is completed, we perform the modeling of the resulting text, which produces and returns the extracted entities that will be used as input to the ML classifier.

2.4 Threat Intelligence

Threat intelligence, also known as Cyber threat intelligence, or *threat intel* for short, is evidence-based information about threats and threat actors that can be used to inform decisions. There are multiple types of threat intelligence *feeds*,

which provide intelligence on different types of threats, such as known malicious IP addresses or domains, or active APTs (Advanced Persistent Threat).

In this project we take advantage of open source threat intelligence feeds providing information on malicious domains and IP addresses, such as PhishTank[8] and OpenPhish[9].

These threat intel sources are used by a part of the system we develop to check each URL and IP address found in the email to assess whether it is a known threat. By using the threat intel feeds in addition to the classifier's decision, it may increase the overall accuracy of the system's decision on whether an email is malicious.

For this project, we have used the following threat intelligence feeds:

- IPsum
- Collective Intelligence Network Security
- OpenPhish
- Malware Domain List
- Cybercrime Tracker
- PhishTank

Some of these feeds are specific to certain types of malicious activity, such as phishing and malware, while others are more general feeds that provide intelligence on domains, IPs and URLs that are known to conduct or be used for malicious activity.

2.5 System Design

This section presents a high-level overview of the processes involved and the planned architecture for the system, starting with the employee reporting a suspicious email to the final decision and actions taken by the system. We have named this system "PURA", an acronym for "**P**rocessing **U**ser **R**eports **A**utomatically".

As presented in Figure 2.3, there are multiple steps involved in the proposed process of automatically remediating user incident reports. The cycle begins with an employee receiving a suspicious email in their inbox, upon which the regular routine of reporting the event to the security team takes place. This is done by the employee, who sends the email as an attachment to a designated mailbox operated by the security team. It is a prerequisite that a security awareness program has made the employees aware of this practice.

With the system in place, the idea is to handle the remainder of the process in an automated fashion, instead of having a member of the security team perform these tasks manually. Thus, the next step in the process is to receive and analyze the attached email message file. The attachment is downloaded, and the pre-processing of the email's contents begins with extracting the plain text contents from the HTML-formatted message. Furthermore, language detection and

[8] `https://www.phishtank.com/`

[9] `https://openphish.com/`

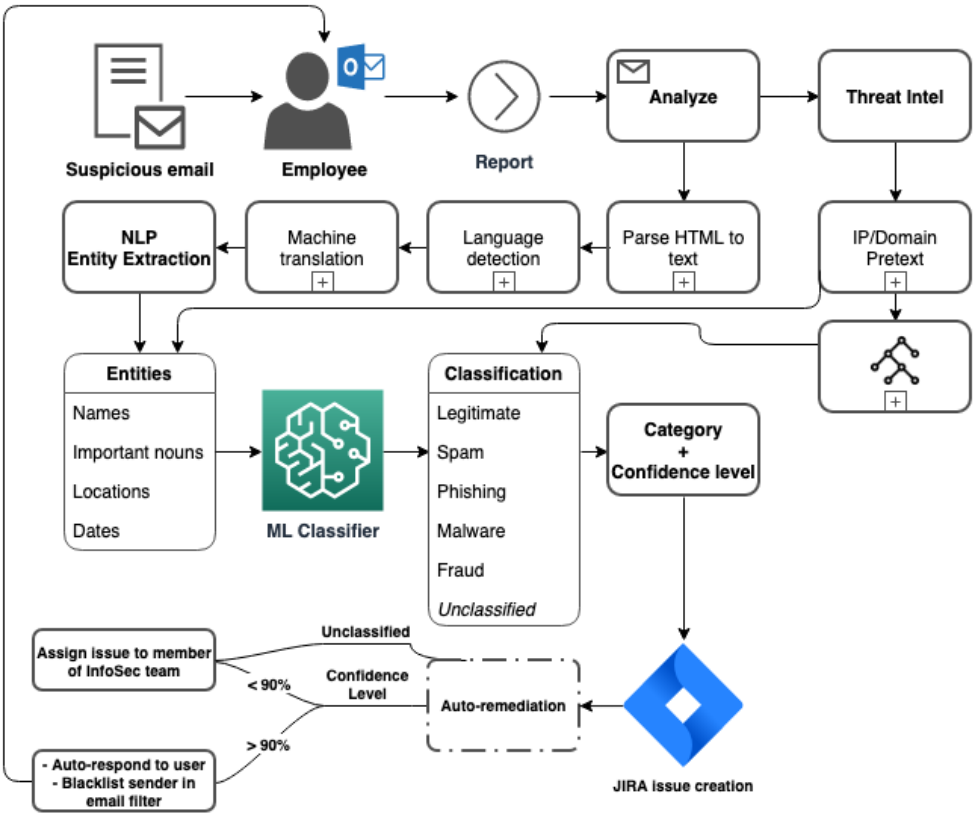


Figure 2.3: The high-level system architecture of *PURA*.

machine translation is performed before the data is fed into the Entity Extraction *module*. This is where entities including names, nouns, locations and dates are extracted. The output from this module is then used as input to the machine learning classification module, which attempts to categorize the type of email. Upon completion, the classifier module outputs the predicted category along with a confidence level which represents the certainty.

At the same time as the sub-processes described above take place, a separate module performs checks against threat intel feeds, checking all domains, IPs and URLs found in the email's contents and headers in order to determine whether any known malicious entities are referenced. The output of this module is used as an addition to the classifier's output.

The next step is related to generating metrics and the assignment of tasks. This step is performed using a separate module with an integration of JIRA[10], an issue tracking product, to create an issue for the specific incident report.

The final step, which is the automated remediation, takes an action depending on the results of the classification – if the confidence level is satisfactory (e.g. above 90 per cent), the system responds to the reporting employee with an action

[10] `https://en.wikipedia.org/wiki/Jira_(software)`

that they should take (e.g. that they should delete the email, that the email is safe, etc.). Additionally, the system could interact with the email filtering system to blacklist the sender, so that no further emails from this particular sender is delivered. However, if the confidence level is too low, the system assigns the issue created in JIRA to a member of the security team so that it can be analyzed by a human-being.

Chapter 3

Related Work

This chapter will look at the currently available research in relation to the defined research questions, and discuss existing knowledge and areas where knowledge is currently lacking.

Will Rapid Feedback to User Incident Reports Increase the Security Awareness of Employees?

The employees of an organization are often seen as the weakest link of the chain, where the compromise of an individual's accounts or devices may result in a threat that is difficult to detect. Many attacks use the human aspect as the initial point of entry, where tactics such as spear-phishing (a targeted phishing attack) is used to trick employees into handing over information such as credentials to an adversary.

In order to act proactively and reduce the risk of employees falling victim for such attacks, security awareness training is a common method used to educate individuals within an organization. When employees spot emails that they deem suspicious, they normally report this incident to the information security team. However, due to several reasons, the response time of the information security team may be slow, which may result in less "awareness gain" for the employee that reported the incident.

We believe that rapid feedback to these reports can increase the knowledge gained from their suspicions, as their reports are either confirmed to be malicious, or they are told that their suspicions were incorrect, almost immediately after the report is sent.

Chen et al. (2006) researched the effect of rapid feedback on student learning in a classroom setting, where they state that "[...] we found evidence that rapid feedback use improved knowledge retention (durability) and knowledge application in a different environment (transferability) [...]" [24, pp. 4]. Furthermore, they found that the treatment (i.e. having rapid feedback) positively influenced student scores on quizzes [24, pp. 4]. Although this research is set in a different context than our thesis project, the results are arguably still relevant, where they concluded that "This confirms the value of providing frequent and rapid feedback

to students." [24, pp. 4].

Katz-Sidlow et al. (2016) conducted a study where students provided feedback to their resident teachers. The objective of the study was to assess attitudes towards receiving rapid feedback, where the residents reported the rapid feedback as highly valuable. As stated in [25, pp. 85], "Ninety-four percent (30/32) of residents rated the rapid feedback process as "very helpful," [...]".

Jansson et al. (2011) conducted a simulated malicious email campaign, where the subjects that reacted to the email were requested to go through with an online awareness program. Upon reacting to the simulated malicious email, the subjects were notified of the exercise, resulting in a form of rapid feedback on their actions. The study concluded that "After the exercise it was concluded that distributing simulated malicious emails on the organization's email system has minimal cost involved, continuously educates the most vulnerable users and can make users more aware of information security in general [...]" [26, pp. 79]

[27] describes an approach to raising security awareness through a browser extension that provides feedback when the user behaves in a way that poses a security risk, such as using weak passwords, browsing vulnerable websites, etc. In the paper, they conclude that "Those who received affective feedback felt it helped to increase their security awareness, and that the feedback encouraged them to learn more about online security, a factor which could potentially improve their security awareness in the future, and modify their behaviour." [27, pp. 156].

Ikhsan et al. (2019) proposed an approach to measuring security awareness through two approaches: (1) a behavioral approach, i.e. a phishing simulation; and (2) a knowledge approach, conducted by distributing questionnaires. Through their experiments, they measured the awareness levels of employees by comparing the results of the simulated phishing campaign and the questionnaire.

Abawajy et al. (2010) conducted a study where they evaluate various delivery methods for providing security awareness training. The methods used included text- and video-based content, as well as simulation-based methods (phishing simulation). In their study, they conclude that "[...] all information security awareness training delivery methods are powerful means of empowering people with knowledge on focused topics." [29, pp. 147], and that their investigation suggests that a combination of methods improves the success of a security awareness campaign.

Assenza et al. (2019) reviews several methods for evaluating security awareness initiatives, where they assess and compare methods such as computer games, questionnaires, interviews, and simulated phishing attacks. In their paper, they present security awareness as a concept consisting of three main components; namely *knowledge*, *attitude*, and *behavior*. Further, they propose a set of measurement methods for each component and discuss the advantages and disadvantages of each. They propose phishing simulation as a method relating to the *behavior* component, along with role games. As stated in [30, pp. 18], "[...] role games and practice simulations present similar performances. They have the advantage to be highly reliable but require more time and investments.".

Dodge et al. (2006) conducted a phishing experiment in which they sent sim-

ulated phishing emails to students at the West Point military academy. In their research, they found that there was a positive increase in the user awareness related to clicking links and opening attachments. They conclude their paper stating that "The phishing exercises served to provide an insight into the awareness levels of our students and help us better focus our IA and awareness training." [31, pp. 459]

Is it Feasible to Use Serverless Functions for the Automation of These Tasks?

Serverless cloud functions, also known as *serverless*, are cloud services that follow a pay-per-use policy, in which the user is charged only for the execution time of the hosted system. Thus, in projects where the frequency of execution varies greatly, a company might achieve reduced operational costs by only paying for the cloud service when needed, compared to paying a continuous price for an always-on system.

Although serverless cloud functions are a relatively new concept, having been available in Amazon's cloud infrastructure since 2014, several researchers have looked into the concept of using these services for various systems.

Asghar et al. (2018) discusses the feasibility of using serverless for hosting a Disaster Management Information System, in which they conclude that serverless is a viable solution allowing for efficient resource management and reduced operational costs. Further, they state that it allows for increased scalability of the system.

Feng et al. (2018) investigates the utilization of serverless runtimes for training neural networks (a type of machine learning model). In their paper, they concluded that there still are some challenges for this task when training large models, which is resource-heavy, but also offers some great opportunities with their proposed design for future serverless runtimes. Further, they showed that serverless architectures are feasible for smaller models.

Wang et al. (2019) proposes a framework for machine learning built upon serverless functions, in which they concluded that the training of Machine Learning (ML) models reduced the job completion time significantly, without reducing the quality.

Deese (2018) investigates the utilization of serverless architecture for implementation of a machine learning algorithm, where the author provides a comparison of the computation speed and cost of running on a traditional PC versus serverless deployment.

How Does Preprocessing Impact the Accuracy of the Machine Learning Classification Algorithms Tested?

Email filtering is a process which requires high accuracy, where false positive rates and false negative rates must be low, while true positive rates must be high. False positives means, in this scenario, that legitimate emails are classified as malicious, resulting in legitimate emails not reaching the intended recipient. Similarly, false

negatives are the emails that are indeed malicious, but have failed to be detected by the system, thus reaching the end-user and increasing the risk for both the company and the individual recipient.

In order to keep both these rates at an acceptable level, the system needs to be accurate. In order to find the optimal technique for preprocessing with classification accuracy in mind, we need to compare several options. In addition to performing comparisons ourselves, we will also look into previous research covering the topic of preprocessing and its effect on classification accuracy.

Krasnyanskiy et al. (2019) researches document classification using machine learning, where they look into preprocessing and its effect on accuracy and classification speed. In their research, they present a process for preprocessing and assess the accuracy of classification and the time required to train the classifier before and after preprocessing, for multiple classification algorithms. The results presented in the paper show that the accuracy improved after preprocessing, while the time needed train was reduced.

Nafis et al. (2019) looks into the impact that preprocessing and feature selection has on text classification problems. In their experiments, they evaluate the effect of various steps in preprocessing (including tokenization, stopword removal, etc.) and compare TF-IDF and BoW (Bag of Words) for feature selection. The paper concludes that the results "[...] clearly revealed that pre-processing activities and feature selection gave a significant impact on the text classification performances." [37, pp. 279], and that the TF-IDF approach to feature selection outperformed the BoW approach.

Uysal et al. (2014) examines the impact that preprocessing has on text classification, where they look at the effect of all possible combinations of widely used preprocessing tasks. Through their experiments, they found that appropriate combinations of preprocessing tasks, dependent on the task, may provide a significant improvement on accuracy – while inappropriate combinations may reduce accuracy. They conclude the paper stating that "[...] there is no unique combination of preprocessing tasks providing successful classification results for every domain and language [...]" [38, pp. 111], and that "[...] researchers should carefully analyse all possible combinations of the tasks rather than completely/individually enabling or disabling them." [38, pp. 111].

Is it Feasible to Implement a Credibility Indicator Based on Previous Reports by the Same User, in Accordance with the GDPR?

Bonatti et al. (2019) discusses the insufficiency of and reduced accuracy of anonymization of personal data in the context of big data and analytics, where they argue that the most flexible and safe legal basis is explicit consent. They illustrate an approach to consent management and compliance with the GDPR. Gruschka et al. (2018) discusses legal regulations, and provides an analysis on different data protection and privacy-preserving techniques related to big data analytics.

Rustici (2018) looks at the different stages of profiling in relation to the GDPR

and discusses important aspects an organization has to consider before performing profiling of individuals, while González et al. (2019) looks at the legal grounds for data processing and profiling under the GDPR. The paper by Kaltheuner et al. (2018) discusses techniques and purposes of profiling, cases where profiling may be harmful, and looks into profiling and automated decision making in relation to the General Data Protection Regulation.

Is it Possible to Automate the Generation of Metrics Representing Trends in Email Attack Methods in Order to Provide Relevant Security Awareness Training?

We want to provide the employees of the company with appropriate and timely security awareness training, where programs should reflect current trends in email attack methods. In order to achieve this, we want to use the automated system to generate metrics representing these trends.

We were unable to find related literature researching this question through our search, possibly due to this research question being quite specific to the rest of the thesis project, including the implementation.

Chapter 4

Method

The thesis consists of several methods to answer the defined research questions. The following subsections will present the methods used.

4.1 Literature Review of Prior Research

Researching the current state-of-the-art is valuable in order to gain an understanding of the possibilities and limitations. To achieve this, a literature review will be conducted where the goal is to gain knowledge of previous and current work in the fields related to the proposed research questions.

This will be done for all of the research questions.

4.2 Implementation of a Proof of Concept (PoC)

The implementation of the system for automated remediation of user incident reports will be written in the *Python*[1] programming language. Python is a versatile programming language which comes with an extensive set of tools, and includes easy access to utilities necessary for the tasks of this project, including parsing of email message files, machine learning libraries, and more. Additionally, there are a lot of free resources available which will make the work of developing a PoC easier.

Before the development of the PoC begins, we will also create a high-level design for the system's architecture to help us identify the necessary components.

4.2.1 Assessment of Best Practices and Review of Existing Solutions

Prior to the implementation stage we will assess previous work done on the relevant topics, which we may use to our benefit. This assessment includes reviewing several different classification algorithms for the Machine Learning (ML) aspect

[1] `https://www.python.org/`

of the implementation, so that we can identify the algorithms that are most applicable to solving this specific problem. In order to assess the performance and accuracy of the implementation, at least two different ML classifiers will be implemented and compared.

We will also assess general best practices for:

- Building a data set that can be used to train and test the classifiers [44], [46, pp. 317–334]
- Implementation of:
    - Machine learning classifiers [44, 45, 49], [46, pp. 315–345]
    - Natural Language Processing [47, 48]

4.2.2 Assessment of Approaches to Preprocessing

We want to gain knowledge on how and to what extent preprocessing impacts the accuracy of the machine learning classifier. The implementation will be built as a modular solution, where specific functionality can be *switched off and on*, so that we can identify the impact that each function has on the overall accuracy. We will test two different algorithms where we evaluate the effect of various methods for preprocessing the input data to the classifier, such as stemming versus lemmatization, and data standardization. We will also evaluate the impact of feature selection methods such as TF-IDF versus word vector counts.

4.3 Evaluation of Security Awareness

In order to gain an understanding of the impact that rapid feedback (through the automated system) has on the security awareness of employees, we will need a method to evaluate the level of security awareness prior to deployment and post-deployment. We want to understand whether, and to what extent, rapid feedback to user reports has an impact on security awareness of employees, which we will measure by performing large-scale simulated phishing attacks and collecting empirical data on the results of these campaigns. As stated by Gardner et al. (2014), "Phishing assessments evaluate the organization's resistance against malicious e-mail content." [50, pp. 54]. We will use an existing, internally developed framework for the simulated phishing campaigns. This evaluation will also include an assessment of weaknesses related to measuring security awareness through simulated phishing campaigns.

Gardner et al. (2014) and Ikhsan et al. (2019) will be used as the primary resources when performing these simulated phishing campaigns.

We will also investigate whether it is possible to use the automated system to generate metrics that represent current trends in email attack methods in order to provide timely and relevant security awareness training.

4.4 Assessment of Cost-effectiveness

Responding to user incident reports is a task that currently requires manual labor, a task commonly performed by the information security team. In a large company where such reports are frequent, the workload for members of the information security team is often significant. We aim to lower the costs of this work by automating these tasks. In order to measure the related costs, we will record the amount of time spent by information security employees on these tasks over a given period of time, and compare it to the costs of running an automated, event-triggered implementation running on serverless workloads.

4.5 Feasibility Study for Credibility Indicators

We will assess the feasibility of implementing a credibility indicator for user reports based on previous reports from the same person, which can potentially be used to improve the accuracy of the classifier by estimating the likelihood that a report from a given user is valid. Since this will be considered profiling under the General Data Protection Regulation (GDPR), we will review strategies based on a literary study, to ensure compliance with the GDPR.

4.6 Data Set

In order to train the Machine Learning classifier, we require a large amount of emails from each category (as presented in Section 2.1.2). In general, it is preferable to use open (public) data from a research reproducibility and transparency point of view, so we will try to locate public sources for these data.

4.6.1 Data Collection

We need to collect relatively recent emails to use as input for our classifier. We have a few approaches to retrieve these data, where the different approaches act as both backup strategies for others in case one does not work out as planned, and as a means to fill any voids left by the other methods.

The first option is to perform searches for publicly available data sets first. For this approach we will use sources such as *Kaggle*[2] and *Google Dataset Search*[3], and expand our search to identify other sources if the aforementioned sources are lacking.

If these searches do not result in appropriate or enough data, the second option is to use emails from Sportradar's email system. An issue with this approach is that it will require that a group of employees voluntarily contribute their emails and give us explicit consent. Additionally, since this data may involve personally

[2] `https://www.kaggle.com/`

[3] `https://datasetsearch.research.google.com/`

identifiable information, we will need to conduct a legal assessment of the privacy aspect of this data collection in collaboration with personnel with qualified legal competence.

In the event that none of the aforementioned approaches provide the data we require, we could potentially use our personal emails as the training data; but this leads to another issue – the classifier will then be trained on personal emails, which may vary in content and sophistication in comparison to the emails an organization may receive. This may impact the classifier's accuracy negatively if deployed within an organization.

We need to acquire a large amount of emails, of both the legitimate and the various malicious kinds. Thus, we may end up collecting the data from several or all of the sources mentioned above. If we acquire our training data from a source such as *Kaggle*, the data may already be labelled and ready for use. However, in most cases, we will need to label each email manually before it's fed into the classifier to train, test and validate the model.

Chapter 5

Results

In this chapter, we present the results of our research based on the methods presented in Chapter 4. During our research and project work, an issue arose which led to us not being able to follow through on all elements of the plan; the *Coronavirus pandemic*[1], also known as the COVID-19 pandemic, led to a health and financial crisis for many organizations worldwide. Due to this unforeseen circumstance, our ability to conduct some parts of the research was affected. Thus, some of the planned research and expected results will be discussed here, and suggested as future work in Chapter 7. This is described in more detail in the affected sections.

5.1 Implementation of PoC

The system has been implemented using a modular approach, where we have tried to separate each part of the system discussed in Section 2.5 into smaller pieces of software (modularization). This has been done for several reasons: in order to simplify maintenance, improve readability of the code, and to separate the logic so that each module can be used individually outside the context of this specific task. [51]

5.1.1 Modules

The implementation consists of 6 separate modules, each designed to solve a specific problem or small set of problems – where the final collection of these modules is the final system PoC named "PURA". The system consists of the following modules:

[1] https://en.wikipedia.org/wiki/COVID-19_pandemic

Table 5.1: The modules that make up *PURA*.

Module	Purpose
emailyzer	Parses .eml and .msg emails into a common format.
juicer	Performs Entity Extraction on text.
katatásso	Handles Machine Learning training and classification.
oionós	Checks IPs, domains, etc. against threat intel feeds.
jira	Handles interaction with the JIRA api.
mailer	Fetches emails from mail servers, and sends responses to reporters.

Emailyzer

The email parsing module, named *Emailyzer*, is built on top of the standard *email* Python library and the third-party libraries *extract-msg*[2] and *mail-parser*[3]. This module is responsible for parsing email files to a common format and interface for further processing, since the emails that will be parsed can be in two formats: the EML format (complies with the RFC-822 standard); and the MSG format (Microsoft Exchange mail document). The module allows us to parse email files of either type, and parses the HTML contents to plain text, extracts IP addresses, hosts and domains, and more.

The source code of *emailyzer* can be found at `https://github.com/mortea15/emailyzer`.

Juicer

Juicer contains the functionality to perform entity extraction on the input, which is done prior to feeding the input to the classifier. The implementation was built using the NLTK Python library (Natural Language Toolkit) [52], a library that provides tools to work with natural language processing.

The source code of *juicer* can be found at `https://github.com/mortea15/juicer`.

Katatásso

The Machine Learning module, *Katatásso* (*GR: classify*), handles the training and classification of the input, where functionality of the *Juicer* module is used for preprocessing. The implementation is based on the Python Machine Learning library scikit-learn (sklearn) [53], which features various algorithms for tasks such as classification.

The source code of *katatásso* can be found at `https://github.com/mortea15/katatasso`.

[2] `https://pypi.org/project/extract-msg/`
[3] `https://pypi.org/project/mail-parser/`

Oionós

Oionós (*GR: Omen*) is the module used to check the URLs, IPs and domains against threat intelligence feeds. The module has a preset list of publicly available feeds, where each host (i.e., URLs, IPs and domains) is checked to see if any matches are found. The module outputs whether the result was found, a confidence level, and the feed that contained the host. The confidence is relatively naïve, and is based on whether the host was a full match (e.g. "www.malware.com/download" was found exactly as is), or a part of the host was found (e.g. the same domain, but with a different path).

The source code of *oionós* can be found at `https://github.com/mortea15/pura/tree/master/pura/modules/threat_intel.py`.

5.1.2 PURA

The complete PoC, named "PURA", is the combination of the modules mentioned in Section 5.1.1, which make up the system for automated remediation of user incident reports. This subsection presents the major versions of PURA produced during the development process,

The source code of *PURA* can be found at `https://github.com/mortea15/pura`.

Version I

For the first version of the PoC, we used a data set with a size of about 7000 samples from the categories legitimate, spam, and phishing to train the classifier. These samples were collected from the public sources presented in Table 5.18. For this version of the implementation, we used stemming for the preprocessing, and a classifier with features extracted using the *word vector counts* technique, where we created a dictionary by counting the N ($N = 5000$) most common words in the collection of documents (i.e., the collection of the text contents from the emails). The data set was then created from the resulting dictionary by counting the occurrence of each word and assigning a label representing the category.

Version II

The second version of the PoC included a new approach to feature extraction, where we decided to implement TF-IDF (Term Frequency-Inverse Document Frequency) for feature extraction, as an alternative to word vector counts. We also implemented a function to allow for standardization of the training data to investigate its impact.

As for *juicer*, the entity extraction module, we implemented a function to perform *lemmatization* for the preprocessing of text, so that we could compare the results of stemming and lemmatization.

Version III

The classifier was built on one of the implementations of Naïve Bayes in *sklearn*, specifically the *MultinomialNB* algorithm – due to it being commonly used in text classification problems. As stated in [56], "[...] implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification". We wanted to use at least two different algorithms for classification in order to determine which implementation achieved the highest accuracy – therefore, we chose to implement another classifier based on *sklearn's ComplementNB* implementation. The Complement Naïve Bayes algorithm is "[...] an adaptation of the standard multinomial naive Bayes (MNB) algorithm that is particularly suited for imbalanced data sets." [57].

We had also identified and retrieved a data set of fraudulent emails to extend our training data set, as well as an additional data set of spam messages to complement the spam data set we had collected previously. With these additions, the spam set increased from ∼500 samples to ∼6000 samples, while the fraud set consisted of ∼4000 samples.

We implemented two algorithms to compare:

- Multinomial NB
- Complement NB

5.1.3 Experiment I

The first experiment was conducted using a Multinomial Naïve Bayes classifier, where we used stemming in the preprocessing process. The data set used is presented in Table 5.18. The goal of this experiment is to evaluate the difference between using Word Vector Counts and TF-IDF for feature extraction.

Word Vector Counts

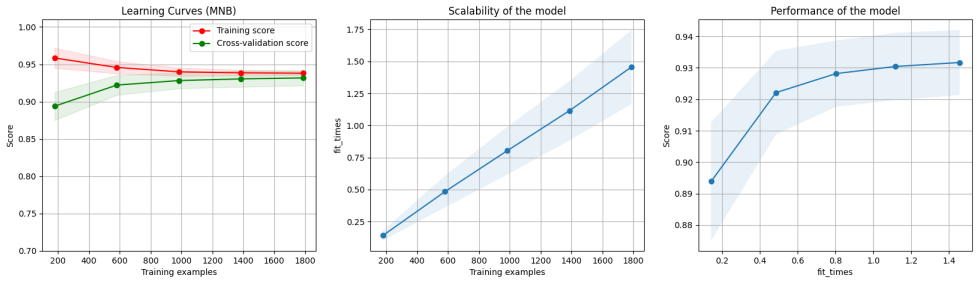


Figure 5.1: Learning curves for Multinomial Naïve Bayes with Word Vector Counts.

As discussed in Section 2.2.3, a learning curve shows the validation and training score of a classifier. On the X-axis, we have the number of samples used to train the classifier. On the Y-axis, we see the scores of the training and cross-validation.

The graph in the second column, *Scalability of the model*, shows the times required by the model to train with various sizes of training data set. On the X-axis, we have the number of samples used to train the classifier, while the Y-axis holds the times required to train.

The third graph shows how much time was required to train the model for each training data set size. The X-axis displays the times required to train, and the Y-axis shows the score.

Table 5.2: The classification report for Multinomial Naïve Bayes with Word Vector Counts.

	precision	**recall**	**f1-score**	**support**
Legit	0.88	0.99	0.93	786
Spam	0.76	0.93	0.84	147
Phishing	0.99	0.89	0.94	1304
accuracy			0.93	2237
macro avg	0.88	0.94	0.90	2237
weighted avg	0.94	0.93	0.93	2237

The values presented in Table 5.2 shows the scores of each category/label and the averages. These values range from 0 (worst) to 1 (best). The precision scores

in the table represents the ability of the classifier not to label a negative sample as positive. The recall scores represent the classifier's ability to correctly identify all the positive samples, while the F1 is a weighted average of the precision and recall.

Table 5.3: Cross validation scores for Multinomial Naïve Bayes with Word Vector Counts.

	precision	**recall**	**f1-score**	**accuracy**
cross validation	94.16%	93.25%	93.36%	93.25%
				93.16%

The cross validation scores, presented in Table 5.3, tells us how the results of a classifier will generalize to a never seen before data set or sample. The accuracy presented in bold on the third row represents the accuracy of the classifier, i.e. the percentage of correct predictions for the data set.

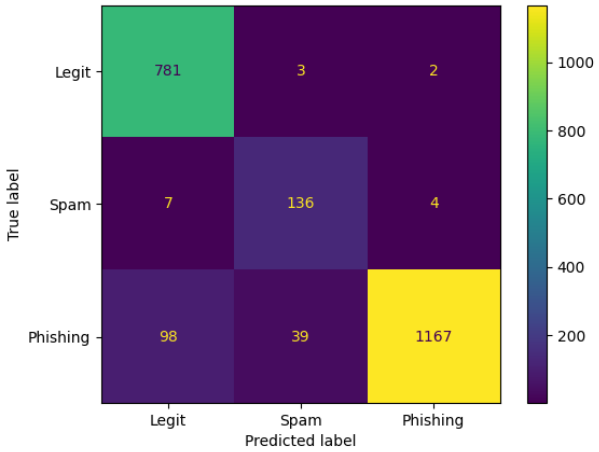


Figure 5.2: Confusion matrix for Multinomial Naïve Bayes with Word Vector Counts.

The confusion matrix presented in Figure 5.2 visualizes the performance of the classifier's predictions, where the X-axis holds the predictions while the Y-axis holds the true labels. In the first column of the first row, we see the number of correctly labeled samples for the category "legit". In the second column of the second row, we see the correctly labeled samples for the category "spam", while the third column of the third row represents the correctly labeled samples for "phishing". This plot provides a quick overview of all categories and which label they were assigned by the classifier. As we can see, of the category "legit", the classifier correctly predicted 781 samples, while 3 samples were labeled as spam and 2 were labeled as phishing. In the confusion matrix, the colors represent the number – as we can see, the third column on the third row presents the number of correct

predictions for Phishing – an issue with the scale in the confusion matrix, occurring due to the imbalance in the data set, is that the colors may be misleading. This is because there are fewer samples of Legit and Spam than there are Phishing samples in the data set. Applying normalization to the data would balance the scales making the colors more representative, however this comes with the disadvantage of not displaying the actual numbers, but rather the ratios. Thus, we chose to use counts for these matrices.

Term Frequency-Inverse Document Frequency (TF-IDF)

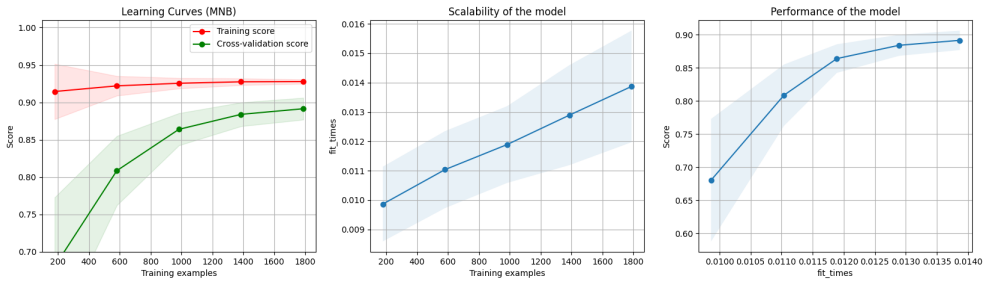


Figure 5.3: Learning curves for Multinomial Naïve Bayes with TF-IDF.

Table 5.4: The classification report for Multinomial Naïve Bayes with TF-IDF.

	precision	**recall**	**f1-score**	**support**
Legit	0.89	0.99	0.93	786
Spam	0.38	0.02	0.04	147
Phishing	0.91	0.94	0.92	1304
accuracy			0.90	2237
macro avg	0.72	0.65	0.63	2237
weighted avg	0.86	0.90	0.87	2237

Table 5.5: Cross validation scores for Multinomial Naïve Bayes with TF-IDF.

	precision	**recall**	**f1-score**	**accuracy**
cross validation	83.69%	89.41%	86.39%	89.41%
				89.62%

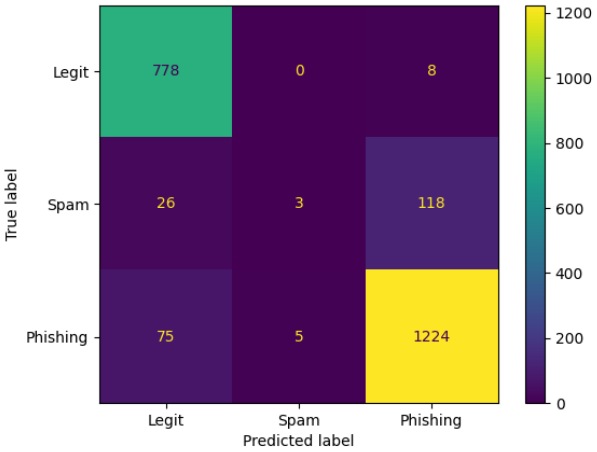


Figure 5.4: Confusion matrix for Multinomial Naïve Bayes with TF-IDF.

5.1.4 Experiment II

Experiment II was conducted using a Multinomial Naïve Bayes classifier. The data set used is presented in Table 5.18. The goal of this experiment is to evaluate the effect of data Standardization on classification scores.

Standardized (Word Vector Counts)

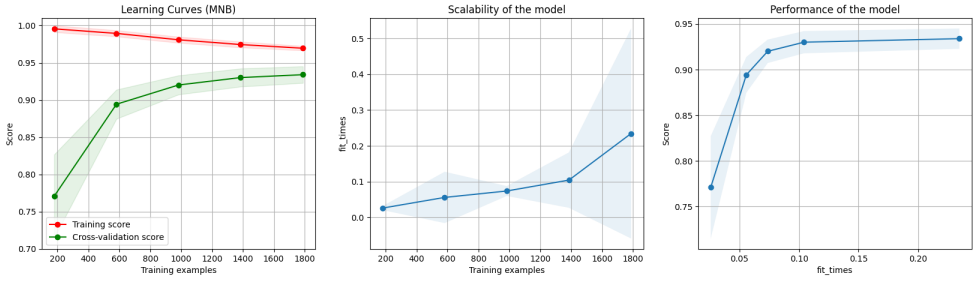


Figure 5.5: Learning curves for Multinomial Naïve Bayes with Standardization (WVC).

Table 5.6: The classification report for Multinomial Naïve Bayes with Standardization (WVC).

	precision	**recall**	**f1-score**	**support**
Legit	0.91	0.97	0.94	786
Spam	0.69	0.89	0.78	147
Phishing	0.99	0.92	0.95	1304
accuracy			0.93	2237
macro avg	0.86	0.93	0.89	2237
weighted avg	0.94	0.93	0.94	2237

Table 5.7: Cross validation scores for Multinomial Naïve Bayes with Standardization (WVC).

	precision	**recall**	**f1-score**	**accuracy**
cross validation	94.21%	93.65%	93.79%	93.65%
				93.33%

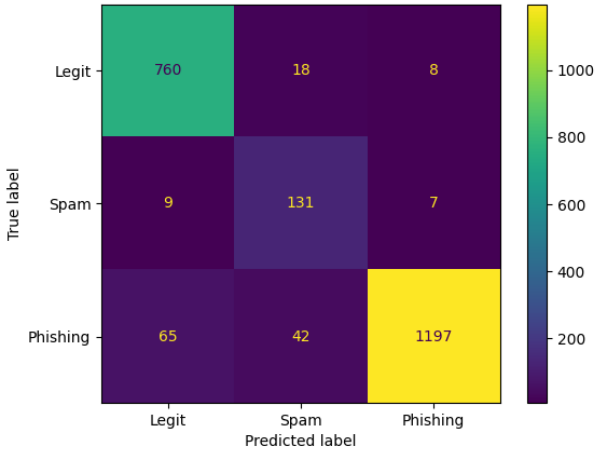


Figure 5.6: Confusion matrix for Multinomial Naïve Bayes with Standardization (WVC).

Standardized (TF-IDF)

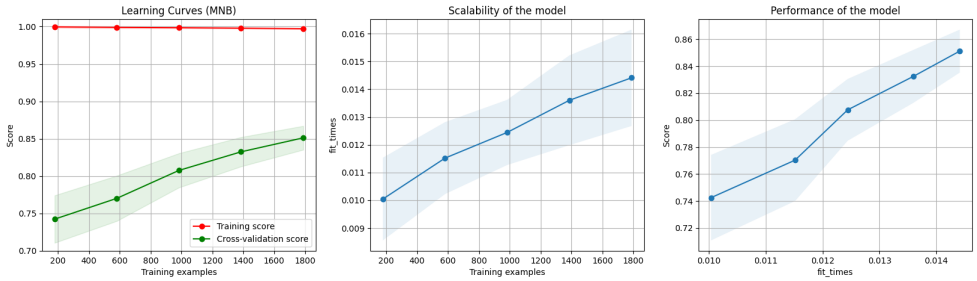


Figure 5.7: Learning curves for Multinomial Naïve Bayes with Standardization (TF-IDF).

Table 5.8: The classification report for Multinomial Naïve Bayes with Standardization (TF-IDF).

	precision	**recall**	**f1-score**	**support**
Legit	0.89	0.99	0.93	786
Spam	0.38	0.02	0.04	147
Phishing	0.91	0.94	0.92	1304
accuracy			0.90	2237
macro avg	0.72	0.65	0.63	2237
weighted avg	0.86	0.90	0.87	2237

Table 5.9: Cross validation scores for Multinomial Naïve Bayes with Standardization (TF-IDF).

	precision	**recall**	**f1-score**	**accuracy**
cross validation	83.69%	89.41%	86.39%	89.41%
				89.62%

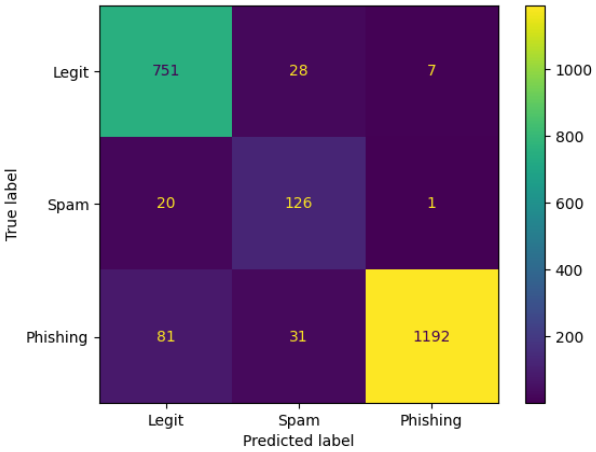


Figure 5.8: Confusion matrix for Multinomial Naïve Bayes with Standardization (TF-IDF).

5.1.5 Experiment III

The third experiment was conducted using a Multinomial Naïve Bayes classifier, using lemmatization for preprocessing. The results of the experiment will be compared against the results from Experiment I (stemming). The data set used is presented in Table 5.18.

Lemmatization (Word Vector Counts)

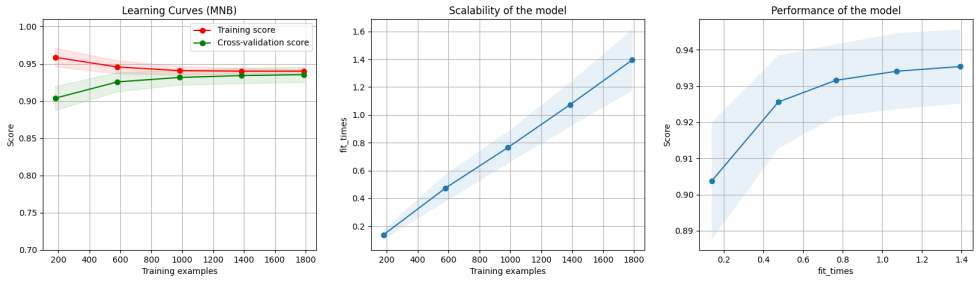


Figure 5.9: Learning curves for Multinomial Naïve Bayes with Lemmatization (WVC).

Table 5.10: The classification report for Multinomial Naïve Bayes with Lemmatization (WVC).

	precision	**recall**	**f1-score**	**support**
Legit	0.89	1.00	0.94	786
Spam	0.79	0.95	0.86	147
Phishing	0.99	0.90	0.94	1304
accuracy			0.94	2237
macro avg	0.89	0.95	0.91	2237
weighted avg	0.94	0.94	0.94	2237

Table 5.11: Cross validation scores for Multinomial Naïve Bayes with Lemmatization (WVC).

	precision	**recall**	**f1-score**	**accuracy**
cross validation	94.50%	93.79%	93.84%	93.79%
				93.60%

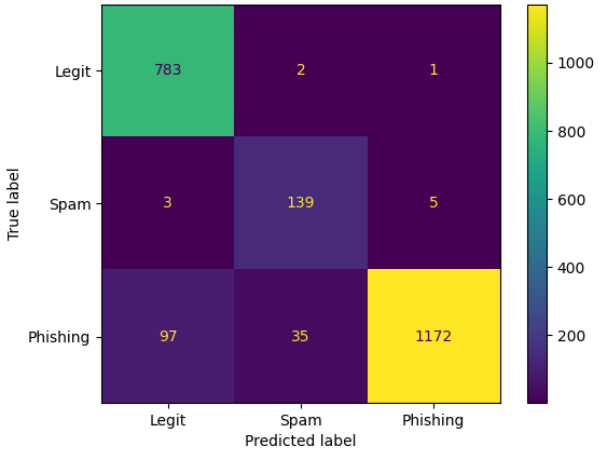


Figure 5.10: Confusion matrix for Multinomial Naïve Bayes with Lemmatization (WVC).

Lemmatization (TF-IDF)

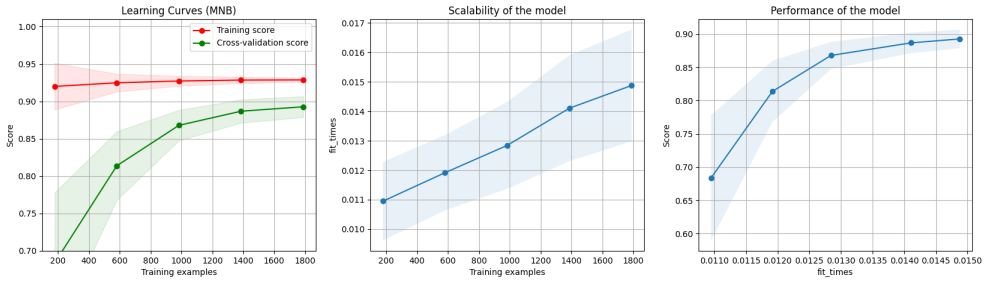


Figure 5.11: Learning curves for Multinomial Naïve Bayes with Lemmatization (TF-IDF).

Table 5.12: The classification report for Multinomial Naïve Bayes with Lemmatization (TF-IDF).

	precision	**recall**	**f1-score**	**support**
Legit	0.89	0.99	0.93	786
Spam	0.50	0.02	0.04	147
Phishing	0.91	0.94	0.92	1304
accuracy			0.90	2237
macro avg	0.76	0.65	0.63	2237
weighted avg	0.87	0.90	0.87	2237

Table 5.13: Cross validation scores for Multinomial Naïve Bayes with Lemmatization (TF-IDF).

	precision	**recall**	**f1-score**	**accuracy**
cross validation	83.96%	89.67%	86.66%	89.67%
				89.76%

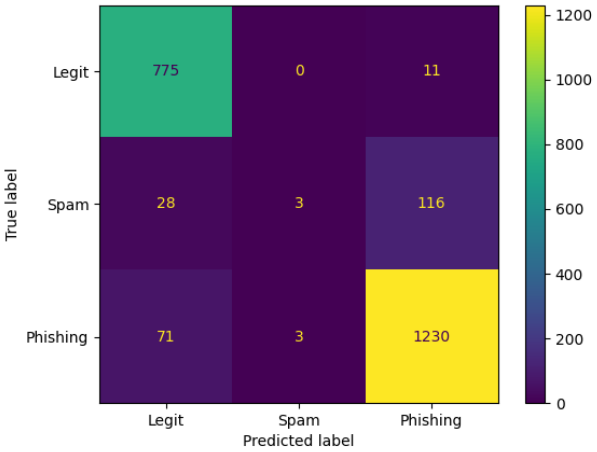


Figure 5.12: Confusion matrix for Multinomial Naïve Bayes with Lemmatization (TF-IDF).

5.1.6 Experiment IV

For Experiment IV, we wanted to compare the Complement Naïve Bayes (CNB) and the Multinomial Naïve Bayes (MNB) algorithms. In this experiment, we used lemmatization for preprocessing, which was chosen based on the results of Experiment IV. The data set used is presented in Table 5.18.

Complement Naïve Bayes (Word Vector Counts)

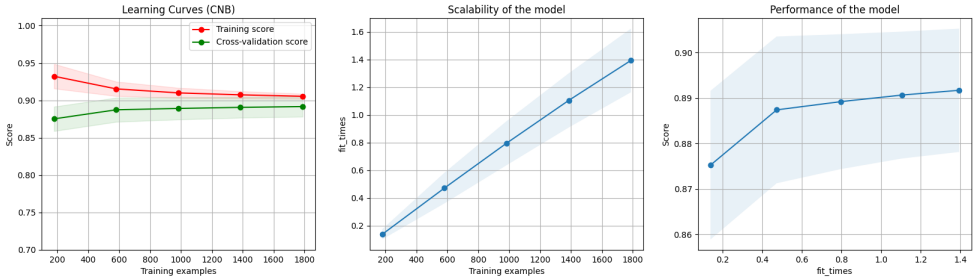


Figure 5.13: Learning curves for Complement Naïve Bayes (WVC).

Table 5.14: The classification report for Complement Naïve Bayes (WVC).

	precision	**recall**	**f1-score**	**support**
Legit	0.79	1.00	0.88	786
Spam	0.92	0.69	0.79	147
Phishing	0.99	0.86	0.92	1304
accuracy			0.90	2237
macro avg	0.90	0.85	0.86	2237
weighted avg	0.92	0.90	0.90	2237

Table 5.15: Cross validation scores for Complement Naïve Bayes (WVC).

	precision	**recall**	**f1-score**	**accuracy**
cross validation	91.26%	89.18%	89.19%	89.18%
				89.76%

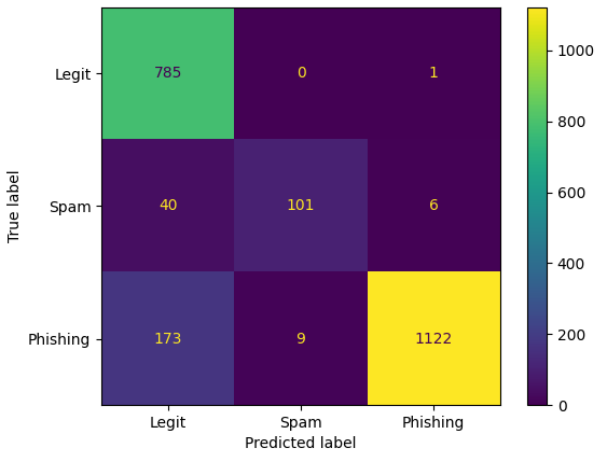


Figure 5.14: Confusion matrix for Complement Naïve Bayes (WVC).

Complement Naïve Bayes (TF-IDF)

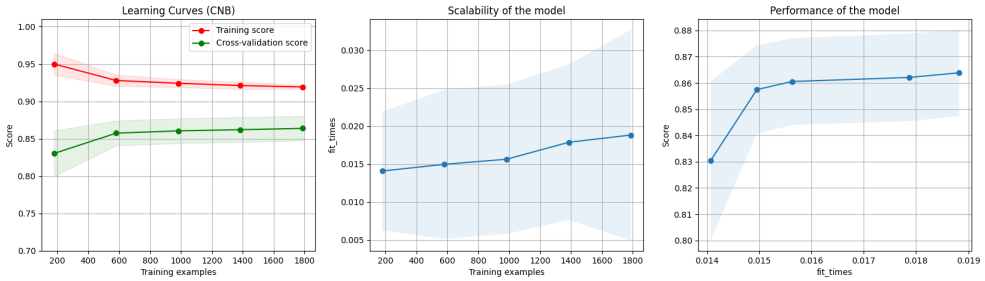


Figure 5.15: Learning curves for Complement Naïve Bayes (TF-IDF).

Table 5.16: The classification report for Complement Naïve Bayes (TF-IDF).

	precision	**recall**	**f1-score**	**support**
Legit	0.80	0.99	0.89	786
Spam	0.78	0.34	0.47	147
Phishing	0.95	0.87	0.91	1304
accuracy			0.88	2237
macro avg	0.84	0.74	0.76	2237
weighted avg	0.89	0.88	0.87	2237

Table 5.17: Cross validation scores for Complement Naïve Bayes (TF-IDF).

	precision	**recall**	**f1-score**	**accuracy**
cross validation	85.70%	86.68%	84.77%	86.68%
				88.06%

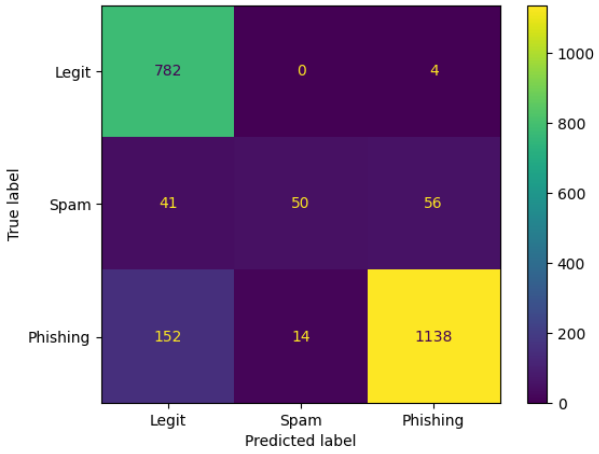


Figure 5.16: Confusion matrix for Complement Naïve Bayes (TF-IDF).

5.2 Effect on Security Awareness

In order to measure the effect that the system has on the security awareness of employees, we referred to previous research and literature on the topic of measuring security awareness. Based on the work from Gardner et al. (2014) and Ikhsan et al. (2019), we chose to conduct simulated phishing campaigns as a method for evaluating the security awareness.

To conduct our phishing campaigns, we used an existing solution for phishing simulations. This system allows us to send personalized emails (e.g. containing the recipient's name) based on templates, where each email opened, link clicked, and login-form filled is logged to collect statistics. Based on the numbers of openers, clickers, logins, and those reporting the email as suspicious, we would be able to see whether these numbers would improve (where the goal is to reduce the former three statistics, and increase the last) after deploying our PoC.

5.2.1 Weaknesses

There are some weaknesses to using simulated phishing campaigns as an indicator of security awareness, where several factors may directly or indirectly affect the results. There are several aspects one needs to consider when conducting a simulated attack:

For instance, we need to evaluate how difficult the phishing email is to spot for the target audience. This is especially challenging, as it depends on human factors such as what background they have, their interests, and more. For instance, it is likely that an employee in a technical role, such as a system administrator, has more insight and knowledge on this topic than a non-technical employee would. For consistency across experiments, it could also be a good idea to use an email

of similar *difficulty*.

We must also consider whether the chosen phishing email is representative for the organization – i.e., that the email is somewhat similar to the real-world phishing attacks that the organization receives.

Additionally, there may be external compounding factors that impact the results of the simulated campaigns. As we discussed in Section 2.1.2, there has been a large amount of emails attempting to exploit the health concerns of people. As a result, phishing and malicious email has received increased attention in media – which may cause people to become more cautious than before. Events like these can also increase the security awareness of employees, at least temporarily. Thus, even if we are able to measure the security awareness, it is difficult to know for certain what caused the *awareness gain*.

5.2.2 Experiment I

Our first simulated phishing campaign took place in January, where we targeted a thousand Sportradar employees spread across the globe. After a week, we *closed* down the campaign and exported the statistics. The metrics collected from this *phishing trip* were then stored for future use, with the goal of comparing them with the results of the final campaign. From this campaign, we collected the following metrics:

- **Number of employees:**
  - Opening the email
  - Clicking a link in the email
  - Logging in to the fake login form
  - Reporting the email
- **Statistics on:**
  - How many of those interacting have reported it
  - How many of those interacting have not reported it
- Organizational positions of those interacting

For each target, we also captured the following:

- **Timestamps of:**
  - Email delivered
  - Email opened
  - Link clicked
  - Login form submitted
  - Email reported
- **Device info:**
  - Device type (phone, computer)
  - IP address (for location, e.g. whether they were at a coffee shop or in the office)

    - Web browser used
- Email address used for login

As for credential capture, we chose not to capture the passwords used. Instead, we captured the email address used to log in and verified that this address matched the address of the email's recipient; so that we could verify if their actual email was used or if they submitted a random value to test. Those that did not submit their actual email were not counted towards the login-statistics.

These results are not included in this thesis, due to their confidential nature with relation to the security of Sportradar.

5.2.3 Experiment II

Due to the situation caused by COVID-19 (as discussed above), we experienced some unexpected challenges with the deployment phase. With an ongoing financial crisis, which resulted in cost reductions including reduced working hours and all employees out-of-office, we did not have access to the necessary resources and time required to deploy the system.

Additionally, the conditions for conducting the second phishing campaign had changed greatly (including less people working, changes in working hours, etc.) from the initial campaign. Thus, even if we had been able to continue as planned, one could argue that the results of the second measurement would be skewed. As a result, we decided to focus our time and attention on other aspects of the project work, and propose this as a part of future work.

5.2.4 Generating Metrics

In Section 4.3, we stated that we wanted to evaluate the possibility of using the PoC to automatically generate metrics that can be used as an indicator of current trends in email attack methods. The idea behind this is to provide timely and relevant security awareness training based on the current threat landscape, as well as having metrics for the C-level executives and other interested parties. These metrics can also be used to determine trends over time.

The system developed throughout this thesis work has a module for JIRA, an issue tracking system, where issues are created for each incident report to help the security team with keeping track of incidents. These issues can also be used to generate metrics, using both built-in JIRA functionality (Reports) and by exporting the statistics to use in other software.

The PoC automatically creates these issues after classification is complete. Figure 5.17 shows an example of such an issue.

By using the built-in Reports functionality in JIRA, we can also create charts representing various aspects – such as an overview of the most common type of malicious email this week; or displaying trends over time. Figure 5.18 is an example of a report generated by JIRA, representing the percentage for each category/class.

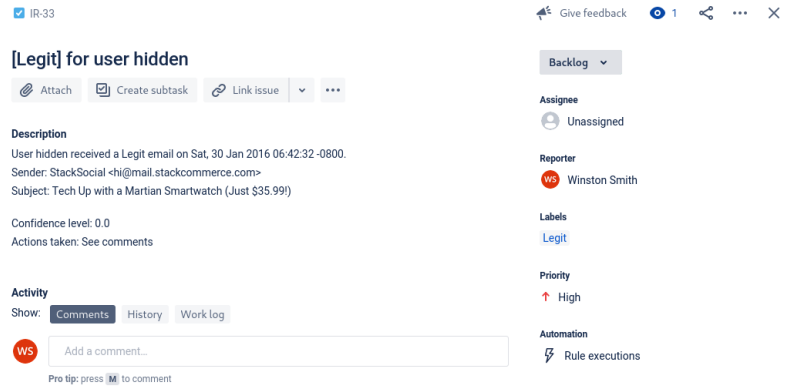


Figure 5.17: An example of an issue created in JIRA by PURA.

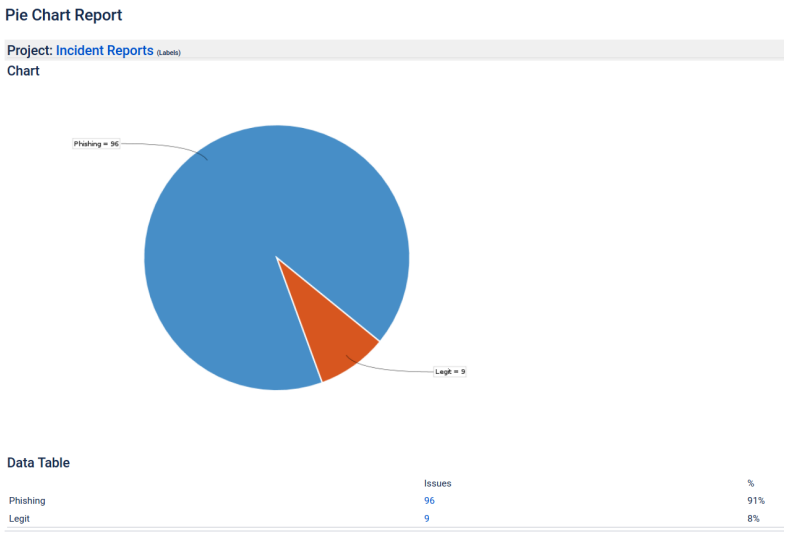


Figure 5.18: An example of a report generated in JIRA.

5.3 Cost-effectiveness

We wanted to measure the cost-effectiveness of the proposed system to see whether it is financially beneficial to have an automated system running on a serverless system compared to the cost of manual labor performed by a human-being. As these tasks are normally conducted by members of the information security team, we were planning to measure the costs by recording the amount of time spent by the team members on these tasks over a given period of time. With the system in place, we would then be able to compare these costs to see whether it could help reduce costs.

Unfortunately, similarly to the experiment on security awareness discussed in the previous section, the COVID-19 pandemic caused some challenges resulting in us being unable to deploy the system. At the same time, due to the situation which as previously mentioned resulted in reduced working hours, we no longer had the resources to record time consumption for the user incident report tasks. Thus, we propose this experiment as a possibility for future work in Chapter 7.

5.4 Feasibility of Credibility Indicators

In Section 4.5, we presented that we wanted to evaluate the feasibility of implementing a credibility indicator. The idea was to use this as an additional layer of support for the decisions being made by the classifier, where the system can check previous reports by the same user to estimate the likelihood of a new report being valid (i.e., correct).

When an employee reports an incident (a suspicious email) to the system, they may add a predefined *tag* specifying the type of email they think it is – such as phishing or malware. Based on the classifier's prediction, we can compare the tag specified by the user with the category predicted by the classifier, to see if their assumption was correct.

Over time, if a user submits multiple reports, we can evaluate the likelihood of the report being valid. However, this involves profiling (automated processing of personal data to evaluate certain things about an individual) under the GDPR. Therefore, we had to explore how to do it in a manner that is compliant with the GDPR.

In order to process these data *as-is*, we may request explicit consent from each user. Thus, any non-consenting user is ignored from the profiling, while allowing us to process the information on the rest. However, this approach comes with another set of considerations, due to the data subject rights of the GDPR – such as the right to deletion and the right to withdraw consent, which means that a consent management system is needed. Another approach was considered, where we could use anonymization techniques: Bonatti et al. (2019) discusses analytics and big data in relation to the GDPR, where they state that anonymous data "[...] are not regarded as personal data, so anonymous data lie outside the scope of the GDPR and can be freely used." [39, p. 7]. Thus, by using anonymous data, we

would be in compliance with the GDPR.

The credibility system essentially requires two pieces of information: (1) a unique identifier for an individual reporting the incident; and (2) the category to which they think the email belongs. The unique identifier could for instance be the user's email address. This is, however, considered personally identifiable information. To deal with this, a solution could be to generate a hash from the email address. By using a strong hash, we can ensure that there is no way to reverse the hash back to the original email address, while allowing us to maintain records of reports to estimate credibility.

5.5 Data Set

This section describes the results of our data collection methodology, the challenges we faced, and how we solved these problems underways.

5.5.1 Data Collection

Ideally, we wanted to use data from publicly available sources as training data, in order to make the research reproducible and provide transparency. However, as we discussed in Section 1.3, and experienced through our online searches, the email data sets that are publicly available are mostly outdated (i.e., collected/-generated from around 2000 – 2010). Using the aged data would most likely impact the classifier's accuracy negatively for classifying modern emails due to them being more sophisticated than those of 10 to 20 years ago. Thus, in order to train a classifier targeted towards the threats seen in emails today, acquiring contemporary data was preferable.

Therefore, we found that the best choice was to go for the second option described in Section 4.6 – gathering emails from employees at Sportradar. This approach would allow us to gather recent emails of a large variety, most of which would also be in an organizational setting rather than a personal – at the cost of reduced reproducibility and transparency.

We began the process by discussing the legal aspect with the company's Data Protection Officer. Based on our discussions, we concluded that the requirements would be:

1. Write a privacy notice
2. Get explicit consent from each volunteer
3. Perform data anonymization on the emails

These steps were decided upon to respect and ensure the privacy of those partaking, where they have the option of opting out and/or having their data removed in the future. We decided that each volunteer should be able to decide whether they wanted to grant us full access to retrieve their emails; or provide a selection of emails to us themselves.

For the next step, we were planning to make an announcement to see if anyone would volunteer. Before we managed to get started with the communications and collection of emails, the aforementioned COVID-19 crisis arose. The situation impacted our ability to conduct the data collection as planned, due to temporary leaves and reduced working hours across the company, as well as a work-from-home solution that made interaction with potential volunteers increasingly difficult.

Thus, the situation at hand resulted in us having to choose the option of using public data for our training, possibly reducing the accuracy of the classifier. It was, however, still a viable option for the proof of concept. Thus, we were back to the public sources identified through our searches. Luckily, the amount of data available was not an issue, so we were able to pick up the work from there.

As mentioned above, we had already identified several sources for various kinds of emails to use as our training data – including legitimate, spam, and phishing emails. Unfortunately, we were unable to locate sources of large collections of emails of the fraud and malware type, but we had enough samples of each for the three other categories to start training the classifier. The data sets we used were taken from the following sources:

Table 5.18: The data set and sources.

Type	Source	Samples
Legitimate	SpamAssassin Corpus [58]	2551 (-4)
Spam	SpamAssassin Corpus [58]	501 (-10)
Phishing	Phishing Corpus [59]	4554 (-308)
Total	*	**7606 (-322)**

As seen in Table 5.18 above, we had identified about 7600 emails from three categories; legitimate, spam and phishing, collected from two separate sources. All of these emails were merged into the data set used for the classification experiments, as presented in Section 5.1. During the processing of the files, some errors occurred with the parsing. The numbers presented inside the parenthesis in the table represent the number of unparsable (unusable) samples.

With Machine Learning, a common challenge is *class imbalance* – that one class (category) has a significantly higher number of samples than others. As we see in Table 5.18, the frequency of spam emails were significantly lower than the other types (thus, spam is a minority class). Although a real world scenario would comprise of more legitimate than malicious mail (so, the real world data is imbalanced), an imbalanced data set could cause the classifier to become biased. Therefore, we looked at strategies to reduce the effect caused by the imbalance.

While conducting the experiments, we discovered another corpus of spam emails to fill out the existing set of ∼500 samples, in addition to a data set containing fraud emails. Thus, with these additions, we had a data set which looked as following:

Table 5.19: The extended data set.

Type	Source	Samples
Legitimate	SpamAssassin Corpus [58]	2551 (-4)
Spam	SpamAssassin Corpus [58], Enron-Spam GP [60]	6069 (-344)
Phishing	Phishing Corpus [59]	4554 (-308)
Fraud	Fraudulent E-mail Corpus [61, 62]	3976 (-456)
Total	*	**17150 (-1112)**

Dealing with Imbalanced Data

To combat the issue introduced by imbalanced data, we found a few strategies:

– Downsampling and Upweighting The first step of this strategy is to *downsample* the majority class. Downsampling means that we use a subset of the majority class as our training data. The next step involves *upweighting*, where we add a weight to the downsampled class, equal to the factor used to downsample [63].

– Adjusting the weights Another technique commonly used in cases of imbalanced data is to adjust the weights of the classes so that the imbalance has a reduced effect on the accuracy. Using this strategy, the classifier weights each class based on the number of samples, which means that the penalty is higher for incorrect classification of minority classes [55, pp. 132].

– Using a subset of the training data In order to avoid imbalance in the data set, another option is to use a subset of the majority class(es) where we select the number of samples based on the number of samples in the minority class. Using this technique, no class would outnumber another in size, as we select N samples from each class. Since the *legitimate* class had the lowest sample frequency, we selected a random subset of each data set with a size of 2500 samples. After extracting the subsets of the classes, we were left with the following sample sizes:

Table 5.20: A subset of the data set.

Type	Samples
Legitimate	2500
Spam	2500
Phishing	2500
Fraud	2500
Total	**10000**

Another benefit of this technique is that we lower the total amount of samples, which reduces the time needed to train the model. Additionally, using too large

data sets may have negative effects, especially on certain algorithms. With this technique, we reduced the total number of samples from ~17000 to 10000.

Since we gathered the training data from multiple different sources, we also faced some challenges with the variety of formatting of these data. The data set acquired from [61], for instance, was a single text-file containing each *EML* (RFC-822) formatted email message. In order for it to be parseable by our email parsing module, we had to perform some processing to extract and store each email in a separate file. We created a simple script to solve this issue, which can be seen in Code listing 5.1.

Code listing 5.1: Python code used to split the fraudulent emails into separate files.

```
import os

fraud_txt = 'fradulent_emails.txt'
out_dir = 'fraud'

if not os.path.exists(out_dir):
    os.mkdir(out_dir)

with open(fraud_txt, 'r', encoding='cp1252') as f:
    emails = f.read()

delimiter = emails[0:8]

emls = [delimiter + eml for eml in emails.split(delimiter) if eml]

for i, eml in enumerate(emls):
    with open(f'{out_dir}/{i}-fraud.eml', 'w') as f:
        f.write(eml)
```

Chapter 6

Discussion

In the following sections, we discuss the results of the experiments presented in Chapter 5.

6.1 Proof of Concept

6.1.1 Implementation

Machine Translation

As presented in Figure 2.3, the plan was to implement the system with language detection and machine translation as part of the process. Due to time restrictions, we did not make time to implement this feature. The resulting PoC is built on a data set in the English language, where it is expected that the input to the classifier is in English. Thus, the classifier will likely not work when used with non-English input. We propose this functionality as future work, where the PoC can be extended to support multiple languages.

6.1.2 Experiments

**All values in the following charts have been rounded up to the nearest whole number.*

Feature Extraction

For experiment I, we compared two approaches to feature extraction: Word Vector Counts and TF-IDF. This experiment was done using a Multinomial Naïve Bayes classifier based on the data set presented in Table 5.18. The data set was imbalanced, where phishing was the majority class, and spam was the minority class.

With TF-IDF (Term Frequency-Inverse Document Frequency), the importance of a term is dependent on the frequency of that word across a collection of documents – so that common words such as "the" have a reduced weight. With word vector counts, we simply count the frequency of a word in a text – each word is weighted equally (meaning that each word is considered equally important). By

using TF-IDF, the weight of a term is based on its frequency across the collection of documents – thus, words that are frequent in one class but less frequent in others give an increased understanding of the importance in that specific collection. TF-IDF is a frequently used technique in information retrieval tasks [55, pp. 296], [48, pp. 116].

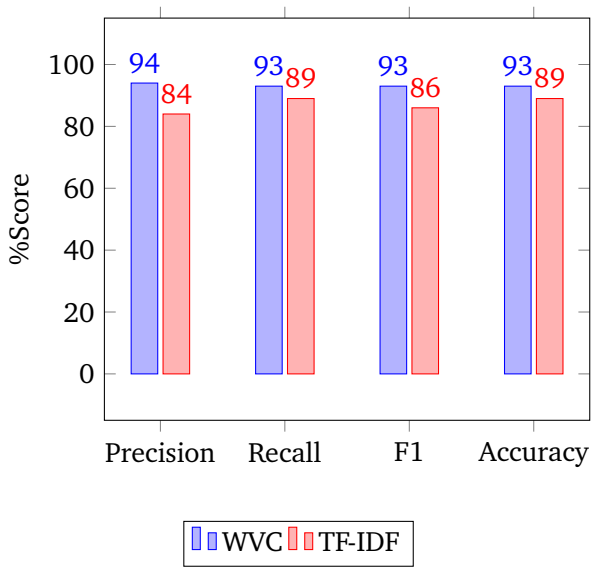


Figure 6.1: Comparison of cross validation scores for Experiment I.

As seen in Figure 6.1, the cross validation scores are high for both methods of feature extraction. Cross validation is used to assess how the classifier will generalize to never-before-seen input – which may help us understand how it will perform on real-world data. This is achieved by splitting the training data k-fold (resulting in k folds), where $k = 5$ for this experiment. Furthermore, $k - 1$ folds are used to train the classifier, while the remaining folds are used to validate the model, where the performance measure is the average of the values computed for each fold in the validation set. The chart in Figure 6.1 tells us that both classifiers achieved high scores overall, but that the WVC-based approach outperforms on all scores for this test set.

Figure 6.2 shows the precision scores for both Word Vector Counts (WVC, left-hand side bar) and Term Frequency-Inverse Document Frequency (TF-IDF, right-hand side bar), for each class label (*Legit*, *Spam*, and *Phishing*). As we discussed in Section 2.2.3, the precision score represents the classifier's ability to not label a negative sample as positive.

For both the Legit and the Phishing class, the scores are relatively similar for both WVC and TF-IDF. The precision score for Spam is, however, significantly lower for TF-IDF than for WVC – where WVC scored twice as high as TF-IDF. If we look at the confusion matrix for TF-IDF in experiment I (Figure 5.4), we see that the

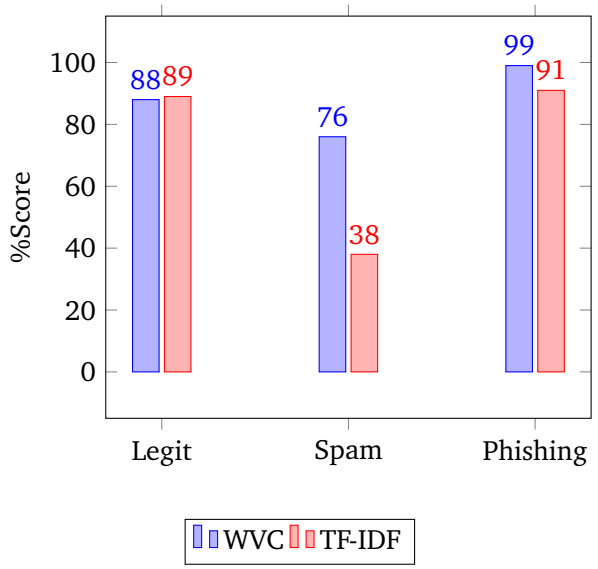


Figure 6.2: Precision scores for Experiment I.

classifier predicted the following for the class Spam:

- 0 *Legit* samples as Spam (False Positive)
- 3 *Spam* samples as Spam (True Positive)
- 5 *Phishing* samples as Spam (False Positive)

The precision is calculated as $tp/(tp + fp)$, where tp is true positive and fp is false positive, which explains why the precision score is low – the classifier did not predict a large amount of non-Spam as Spam, meaning that the number of false positives is low – but it did, however, only predict 3 actual Spam samples as Spam, which means that the number of true positives is low as well.

The cause for the low precision score for the Spam class could be due to it being the minority class, making up ~6.7% of the data set. If we look at the confusion matrix for WVC, seen in Figure 5.2, we see that the classifier predicted the following for Spam:

- 3 *Legit* as Spam (False Positive)
- 136 *Spam* as Spam (True Positive)
- 39 *Phishing* as Spam (False Positive)

As we can see, the WVC-based classifier had a significantly higher amount of True Positives. Thus, it is possible that an imbalanced data set has a stronger impact on TF-IDF, making it biased – or that the features extracted using TF-IDF for Spam are more similar to Phishing, than they are when extracted using WVC. In other words, it may be due to the classifier not learning enough about Spam due to the low amount of samples.

As we presented in 2.2.3, the recall score represents the classifier's ability to

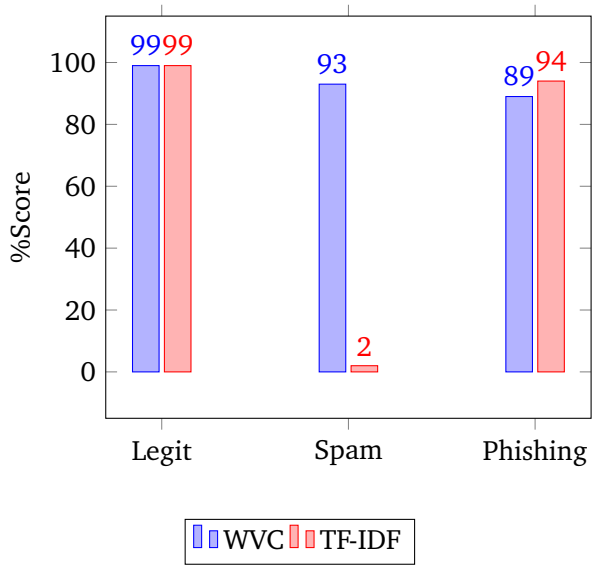


Figure 6.3: Recall scores for Experiment I.

find all positive samples. The chart in Figure 6.3 displays the recall score for Experiment I, for both WVC and TF-IDF.

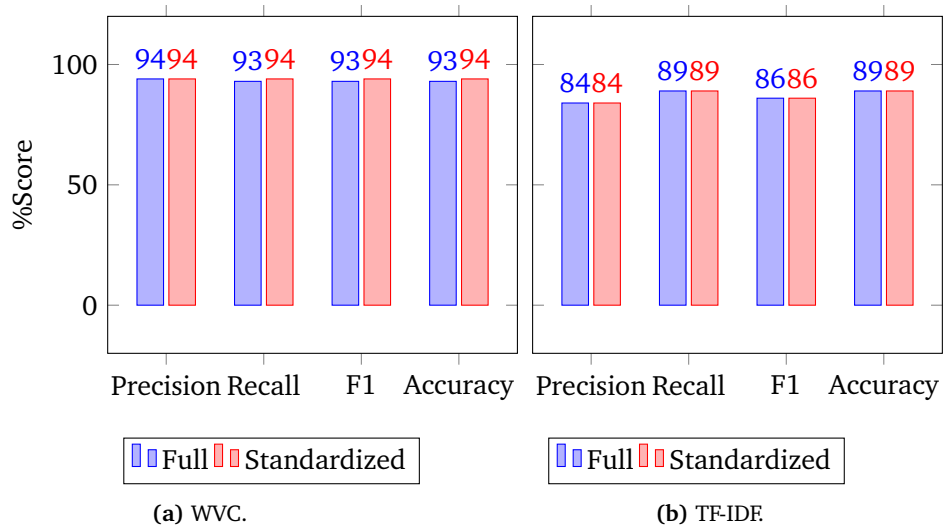
Similarly to the precision score discussed above, the recall score for both the Legit and the Phishing class are relatively similar. For Spam, however, the score is significantly lower for TF-IDF, with a score of 2 compared to 93 for WVC. As we discussed above in regards to the precision score, the TF-IDF method correctly labeled 3 Spam samples as Spam, while 118 *Spam* samples were labeled as Phishing. The large number of false negatives, and the very low number of true positives explains why the score is so low, since the recall score is calculated as $tp/(tp+fn)$ (where fn is false negative).

Standardization

The second experiment was conducted with the goal of discovering the effect of standardizing the data. As presented in Section 6.1.2, the data set was imbalanced – therefore, we considered standardization as a potential solution for dealing with the imbalance. The experiment was conducted using the Multinomial Naïve Bayes classifier, testing both the WVC and the TF-IDF feature extraction methods with standardization. In the following figures, we compare the performance scores for the "full" and standardized data set for both WVC (figure (a); left-hand side) and TF-IDF (figure (b); right-hand side).

The data set we used (see Table 5.18) consisted of ~2500 legitimate, ~500 spam, and ~4200 phishing samples. The data set, as seen from the sample frequencies, was *imbalanced*, and thus likely also biased. In order to deal with the imbalance, we would either need to use a technique to balance the data, or find more data for the minority class (the class where sample frequency is lower than

in others, which was *spam* in this context). Thus, for the second experiment we try to see how standardization impacts the performance.



(a) WVC. **(b)** TF-IDF.

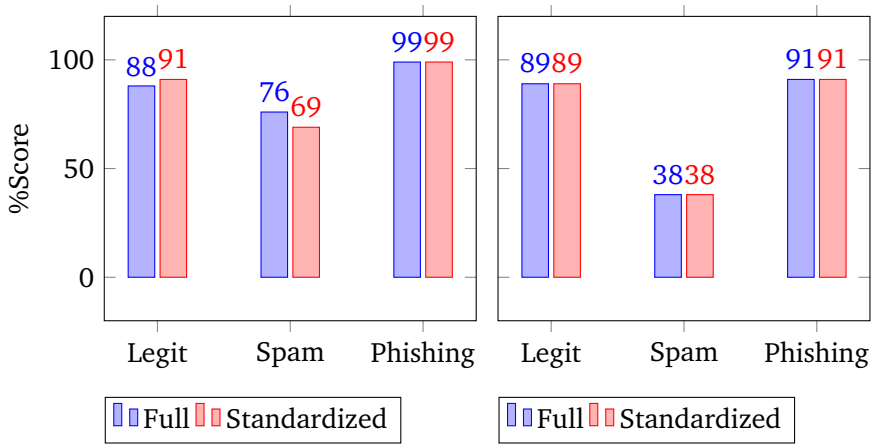
Figure 6.4: Comparison of cross validation scores for Experiment II.

Figure 6.4 presents the results for cross validation scores for experiment II. On the left-hand side, Figure 6.4a holds the scores for *Full* (the original data set) and *Standardized* (where the data set was standardized) for the Word Vector Counts feature extraction method. As we can see, applying standardization did not have a large impact on the scores, where Recall, F1, and Accuracy scores increased by one per cent.

The right-hand chart seen in Figure 6.4b holds the scores for the TF-IDF method, where the application of standardization had no noticeable impact on cross validation scores.

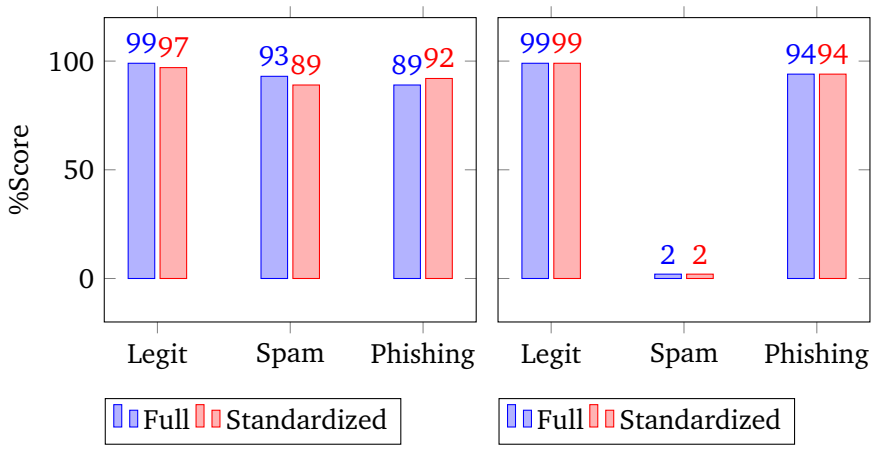
The Precision scores for experiment II are presented in Figure 6.5. As we can see, the standardization had a positive impact on the score for WVC with an increase of ∼3 per cent for the class Legit. For Spam, on the other hand, there was a negative impact from applying standardization with WVC, compared to the original data set – while the score remained the same for the TF-IDF method for all classes. Thus, the results for WVC in Figure 6.5a show that the classifier performs better in terms of not labeling non-Legit (i.e., Spam or Phishing) samples as Legit when standardization is applied, while the opposite is true for the class Spam – it labels more non-Spam samples as Spam.

The recall scores for the second experiment, presented in Figure 6.6, tells us that the classifier performs worse at the task of finding all Legit samples when standardization is applied with WVC. The same is true for the class Spam. For Phishing, however, we see that the classifier's ability to find all Phishing samples



(a) WVC. **(b)** TF-IDF.

Figure 6.5: Precision scores for Experiment II.



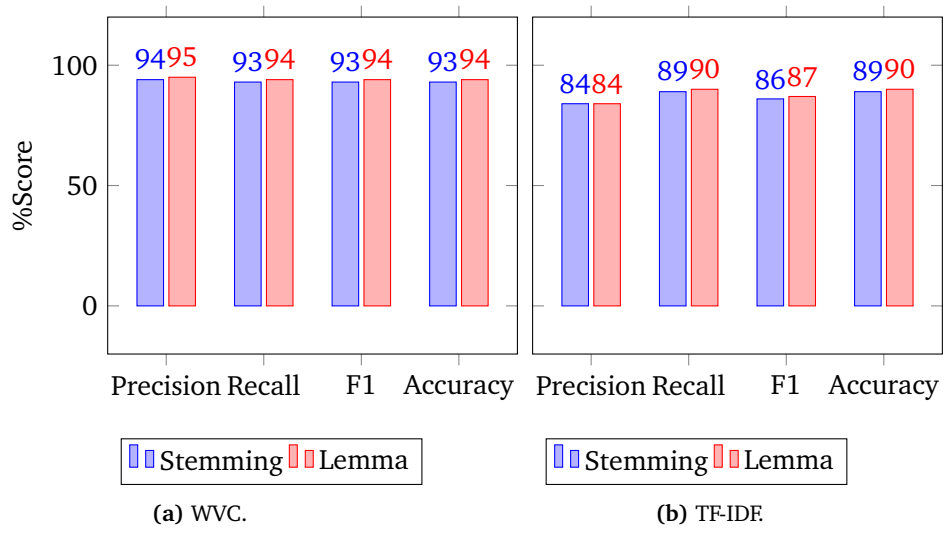
(a) WVC. **(b)** TF-IDF.

Figure 6.6: Recall scores for Experiment II.

has improved by three per cent for WVC with standardization. As for TF-IDF, standardization had no noticeable impact on the scores.

Preprocessing

For the third experiment, we wanted to investigate the impact of stemming and lemmatization, in order to see how the two methods impact classification scores. The experiment was conducted using the Multinomial Naïve Bayes classifier, testing both the WVC and the TF-IDF feature extraction methods with lemmatization, which were compared against the results from Experiment I (where stemming was used). In the following figures, we compare the performance scores for stemming and lemmatization data set for both WVC (figure (a); left-hand side) and TF-IDF (figure (b); right-hand side). The data set used is presented in table Table 5.18.



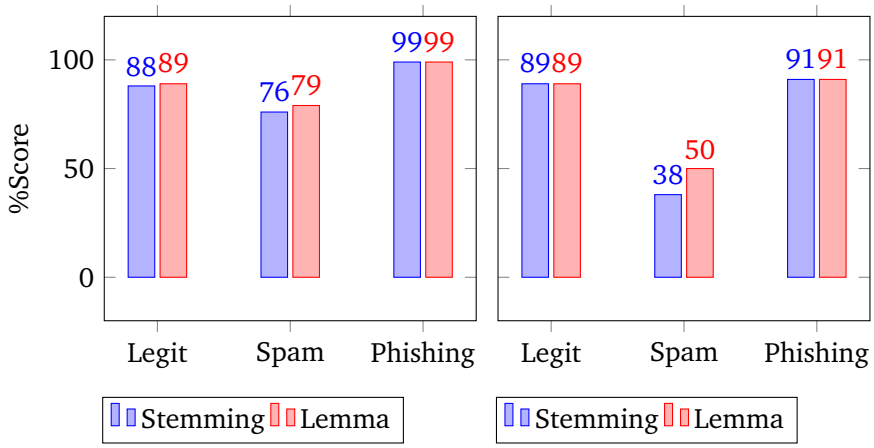
(a) WVC. **(b)** TF-IDF.

Figure 6.7: Comparison of cross validation scores for Experiment III.

As seen in Figure 6.7a, the scores improved slightly for WVC when lemmatization was applied, compared to stemming. The same improvement can be seen in the chart for TF-IDF (Figure 6.7b), except for Precision which remained the same.

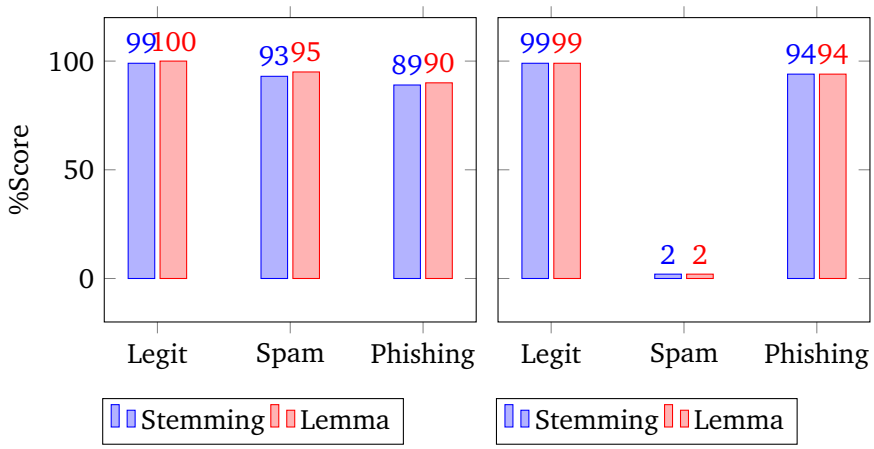
As for Precision scores, the WVC-based classifier's ability to avoid incorrect labeling improved slightly for the class Legit when lemmatization was applied. For Spam, there was a noticeable improvement for both WVC and TF-IDF when lemmatization was used. Based on these results, it seems that the classifier has a lower false positive rate when lemmatization is applied, compared to stemming.

The application of lemmatization improved the classifier's ability to find all positive samples for WVC, as seen in Figure 6.9a. In other words, the true positive rate has increased. For TF-IDF the use of lemmatization had no visible effect



(a) WVC. **(b)** TF-IDF.

Figure 6.8: Precision scores for Experiment III.



(a) WVC. **(b)** TF-IDF.

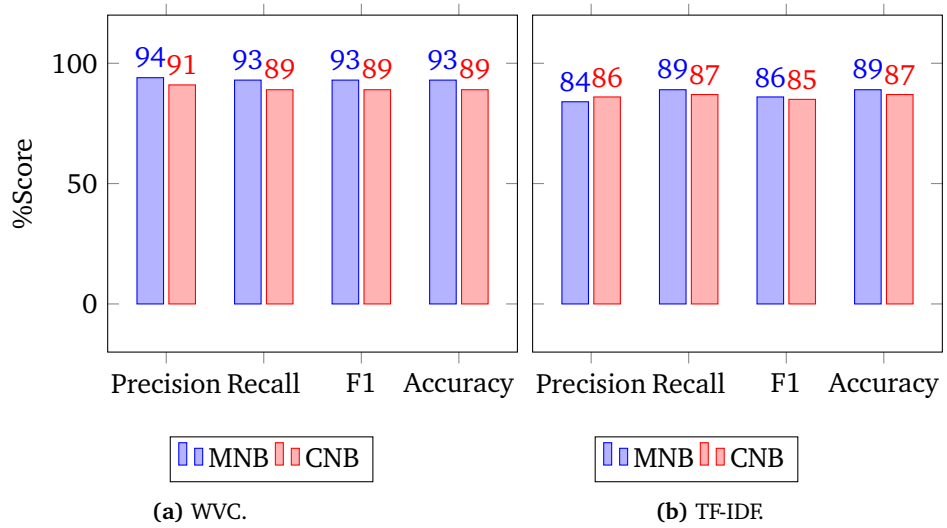
Figure 6.9: Recall scores for Experiment III.

compared to using stemming.

6.1.3 Naïve Bayes: Multinomial vs Complement

The fourth experiment was conducted with the goal of comparing the Complement Naïve Bayes (CNB) algorithm and the Multinomial Naïve Bayes (MNB) algorithm. We used lemmatization for preprocessing (based on the results of Experiment III) for both algorithms, testing both the WVC and the TF-IDF methods for feature extraction. With Complement Naïve Bayes, the weights are computed automatically, which may help reduce the effect of the imbalanced data set.

In the following figures, we compare the performance scores for MNB and CNB for both WVC (figure (a); left-hand side) and TF-IDF (figure (b); right-hand side). The data set used is presented in table Table 5.18.

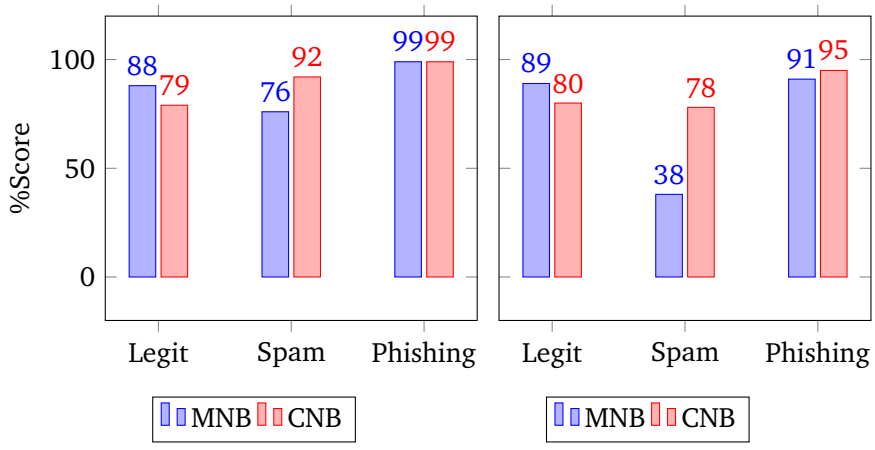


(a) WVC. **(b)** TF-IDF.

Figure 6.10: Comparison of cross validation scores for Experiment IV.

Figure 6.10 presents the cross validation scores for MNB and CNB, where the scores for the Word Vector Counts method are presented in Figure 6.10a, while Figure 6.10b holds the scores for the TF-IDF method. As seen in the charts, the Multinomial Naïve Bayes algorithm outperforms the Complement Naïve Bayes algorithm on all scores, except for the Precision score with TF-IDF.

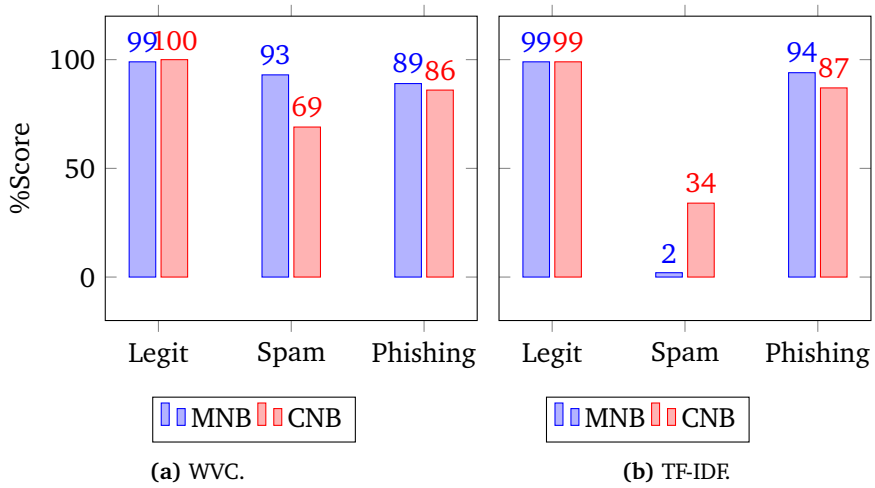
The Precision scores for Experiment IV are shown in Figure 6.11. In the charts, we see that there is no clear *winner* – on some classes, the MNB algorithm performs better, while CNB achieves a higher score on others. As previously mentioned, the Complement Naïve Bayes algorithm is better suited for imbalanced data, where it adjusts the weights automatically. As a result, we see that the CNB algorithm performs significantly better at labeling non-Spam as Spam, i.e. it has a lower false positive rate for Spam. However, it seems that the performance is worse for the two other classes; Legit and Phishing, both of which had much higher sample



(a) WVC. **(b)** TF-IDF.

Figure 6.11: Precision scores for Experiment IV.

frequencies in the data set.



(a) WVC. **(b)** TF-IDF.

Figure 6.12: Recall scores for Experiment IV.

As for the Recall score – the classifier's ability to find all positive samples – we see that for WVC, it performs slightly better at identifying all Legit samples. As seen in the confusion matrix for CNB with WVC in Figure 5.14, the classifier labeled 785 out of 786 Legit samples correctly – with a single sample being labeled as Phishing. For Spam, however, there is a significant decrease in the Recall score when using CNB with WVC, which means that the classifier is worse at finding all the Spam samples. For TF-IDF, however, the opposite is true. MNB with TF-IDF achieved a Recall score of 2 per cent, whereas the CNB algorithm with TF-IDF achieved a score of 34 per cent.

While the automated remediation provided by the system (*PURA*) may reduce the workload required by the information security team, there are some risks that should be considered: Even if one manages to provide a classifier that performs well, and has a high accuracy with a low rate of misclassification, there is no single ML algorithm suited for all problems. Thus, there is always a risk of legitimate email being classified as malicious, or the other way around. If legitimate email is classified as malicious, this can lead to direct loss of business (e.g. sales email), unhappy customers (e.g. customer request over email), etc. At the same time, misclassifying malicious email as benign may cause significant harm as well – for instance, if the system fails to detect an email containing malware, and tells the user who reported the incident that the email is safe, the result may be a malware infection which could cause further harm. Similarly, if a phishing email is classified as legitimate, it may lead to the theft of credentials and more. As the employees may choose to trust the decision of the classifier blindly, it is important that the accuracy is high.

If one were to deploy the system as an email filter, where all incoming email is classified and either allowed/denied depending on the classification result, the consequences of misclassification could be more significant. For instance, if a legitimate email is classified as phishing, it will be blocked by the classifier. This could have a serious business impact; potentially causing business disruption, for instance if the incorrect blocking of the email causes loss of a potential sale or missing out on a customer's request.

Additionally, different organizations may have different priorities for classification. For instance, some might prioritize that legitimate email should never be filtered; others might prioritize that phishing email is stopped from reaching end users while not worrying if some legitimate email is classified incorrectly; and some might need a classifier that performs well over-all. Each implementation used in the experiments in Section 5.1 may be more appropriate for a certain need, where some are very good at correctly finding all legitimate email, while others are better at detecting phishing.

6.1.4 Risks

This subsection will present an evaluation of some risks for an example organization. The organization in this example has a priority of always allowing legitimate email through (no legitimate email should be filtered). They also want to stop as much malicious email as possible, but phishing is considered especially important to block, as they receive a high volume on a daily basis and employees have been compromised through this attack vector in the past. Having invested in security awareness programs, they are confident in the ability of their employees to detect other malicious email such as malware.

Table 6.1: Consequence and probability levels

Level	Description
1	Low
2	Medium
3	High

Identification of Consequences and Risk Estimation

R1: Legitimate sales email classified as malicious An email intended for the sales department is classified as malicious, resulting in the loss of a large sale.

Probability: 2 **Consequence:** 3
Risk: 6

R2: Malware email classified as legitimate An email containing malware is classified as legitimate, and is delivered to the recipient.

Probability: 1 **Consequence:** 2
Risk: 2

R3: Phishing email classified as legitimate A sophisticated phishing email is classified as legitimate, and is delivered to the recipient.

Probability: 2 **Consequence:** 2
Risk: 4

R4: Legitimate customer support email classified as malicious A legitimate email from a customer requesting support is filtered as malicious, resulting in an upset customer and the loss of a sale.

Probability: 2 **Consequence:** 3
Risk: 6

Table 6.2: Overview of the consequences and probabilities of risk scenarios.

Probability Consequence	1 (Low)	2 (Medium)	3 (High)
3 (High)		R1, R4	
2 (Medium)	R2	R3	
1 (Low)			

Risk Mitigation

This section presents some treatment options for reducing the risks.

R1: Legitimate sales email classified as malicious The probability of a legitimate email being filtered as malicious can be reduced if the organization uses a classifier that has an increased accuracy for the legitimate class.

Probability: 1 **Consequence:** 3
Risk: 3

R2: Malware email classified as legitimate The consequence of this risk could be reduced by providing further security awareness training with a focus on malicious attachments. The probability can be reduced by using a classifier with a higher accuracy for detecting malware emails.

Probability: 1 **Consequence:** 1
Risk: 1

R3: Phishing email classified as legitimate To reduce the consequence of phishing email misclassification, the organization could provide security awareness training including simulated phishing campaigns. The probability could be reduced by using the previously received phishing emails as training data.

Probability: 1 **Consequence:** 1
Risk: 1

R4: Legitimate customer support email classified as malicious The probability of this risk could be reduced by training a classifier on a data set containing previous customer support emails the organization has received.

Probability: 1 **Consequence:** 3
Risk: 3

Table 6.3: Overview of the consequences and probabilities of risk scenarios.

Probability / **Consequence**	**1 (Low)**	**2 (Medium)**	**3 (High)**
3 (High)	R1, R4		
2 (Medium)			
1 (Low)	R2, R3		

6.2 Security Awareness

One of the ideas with the system was to assess whether the rapid feedback had an impact on security awareness. Thus, we conducted a simulated phishing campaign and collected results, with the goal of conducting a second campaign post-deployment to see whether it had an impact on (1) number of opens/clicks/logins and (2) number of reports to the security team. Due to the situation explained in Section 5.2, we were unable to conduct the second campaign. Thus, we propose the experiment of measuring *awareness gain* as future work.

Measuring security awareness, and growth in security awareness in particular, is a challenging task to which there has been a large number of proposed methods [28–31]. When measuring awareness, there is no real way to know with certainty what has impacted the awareness levels – the *awareness gain* could be influenced by several factors, including external ones such as a general increase in media attention; or the simulated phishing campaign itself. Regardless of the cause, an increase in security awareness levels is always beneficial.

6.2.1 Metrics

As seen in Figure 5.17, the JIRA issue contains basic information about each event (classification) – including the classification and its confidence level, name of recipient, as well as who sent it and the email's subject. If the confidence level is too low, the issue will be assigned automatically to ensure that the event is handled by a human.

Furthermore, we created an automation in JIRA which takes care of labeling the issue based on the contents of the summary ("[Legit] for user [...]"). We also add a priority level for the issue based on the type of email, ranging from 1 (*Highest*) to 5 (*Lowest*).

As for the creation of reports/dashboards based on the metrics, we found that JIRA's built-in solution works well for this purpose. However, it is also possible to export the statistics for usage in other software.

6.3 Cost-effectiveness

Initially, the plan was to deploy the PoC on a serverless system in the cloud, with the goal of evaluating two aspects:

Feasibility *Is it feasible to deploy PURA on a serverless architecture in terms of performance and reliability?* The goal of the system is to provide subjects reporting incidents with immediate or rapid feedback, where the time from receiving the report to replying to the subject should be as low as possible. Thus, it would be beneficial to run experiments with the system running on both (1) a *traditional* server; and (2) a serverless architecture. With these architectures in place, one could compare the time required to perform the task and whether the serverless

solution is reliable (available when needed, scalable). These assessments could be especially beneficial if the system is deployed as an email filter for all incoming email, where low execution speed or low reliability could be a bottleneck and cause delays for delivery of email.

Cost-effectiveness Additionally, we wanted to assess and compare the cost of:

- Manual labor (no system in place)
- Automated solution running on a serverless system
- Automated solution running on a traditional server

Due to the COVID-19 pandemic (as described in Chapter 5), we were unable to deploy the system during the thesis work – therefore, we propose this as future work.

6.4 Credibility Indicators

As presented in Section 5.4, we evaluated the feasibility of implementing a credibility indicator based on previous reports. Through a literary study, we determined that a viable approach is to anonymize the data. Alternatively, we could use a consent management system and acquire explicit consent from those willing to take part – however, this would require more work to implement, and also result in additional work if the consentees would like to opt out, or exercise other rights related to the GDPR.

Further work could be done in this area, where an idea is to conduct a legal assessment in collaboration with personnel with qualified legal competence to ensure compliance.

An additional benefit of having credibility indicators, related to generating metrics, is that we can use these records to see how the average performs in terms of estimating the category. Furthermore, these data can be used to estimate/evaluate security awareness levels in the organization over time.

6.5 Data Set

As we presented in Section 5.5, we eventually discovered an additional set of spam samples, as well as a set of fraud samples, to extend our data set. The first data set, presented in Table 5.18, was used as training data for the experiments we conducted. Unfortunately, we did not have enough time to conduct the experiments on the extended data set, presented in Table 5.19 – which was more balanced, and contained a significantly higher amount of spam samples, as well as the fraud samples. It would be interesting to conduct the same experiments using the extended data set to see how the results compare to those conducted using the initial data set. Additionally, it would be interesting to compare the performance of the two algorithms with:

- Multinomial Naïve Bayes using the subset (Table 5.20)
- Complement Naïve Bayes using the full data set (Table 5.19)

6.5.1 Data Collection

As discussed in Section 1.3, a large part of previous research on (malicious) email classification using machine learning has used outdated data sets as training data, which may result in a reduced accuracy for detection of more modern and sophisticated malicious email. As we presented in Section 5.5, we discovered several large collections of various kinds of malicious email through our online searches – however, most of these data sets were collected/built years, or even decades, ago. Thus, we wanted to collect contemporary data to use as training data for the classifier where we had planned to retrieve these data from employees at Sportradar willing to contribute. Collecting the data from the organization itself (the same organization as the system will be used by) may have several benefits: (1) the training data more accurately represents the types and contents of emails the organization typically receives; and (2) the classifier will be trained on contemporary data more accurately representing the trends in malicious email today.

As we were unable to complete this part of the research during the thesis work, we propose it as future work.

Chapter 7

Future Work

7.1 Proof of Concept

7.1.1 Analysis of Attachments

An addition to the PoC related to classification of email with malware is to implement a module that performs analysis on attachments in the email. Using machine learning, a classifier could be trained on malware such as obfuscated VBA code (Visual Basic for Applications[1] and VBS (VBScript[2]), a programming language commonly used in malicious document file attachments [64, 65]. The classifier could then be used to classify attachments and potentially improve detection of malware email.

7.1.2 Machine Translation

As stated in Section 6.1.1, we were unable to finish the implementation of language detection and machine translation. Implementing these features could allow for classification of non-English email as well, where the additional steps would involve:

1. Detect the language of the text
2. Translate the text to English

For these steps, one could use existing solutions such as the *langdetect*[3] Python library to detect the language, and perform translation using e.g. the *translate*[4] Python library.

After implementing the functionality, one could conduct experiments to see how machine translation affects the classification.

[1] https://en.wikipedia.org/wiki/Visual_Basic_for_Applications
[2] https://en.wikipedia.org/wiki/VBScript
[3] https://pypi.org/project/langdetect/
[4] https://pypi.org/project/translate/

7.1.3 Confidence Level

We had some challenges with the implementation, where we were unable to provide a confidence level for the classification – thus, since this lays the basis for whether automated action is taken, we propose this as future work.

7.1.4 Using the Full Data Set

As discussed in Section 6.5, we did not manage to conduct the experiments using the extended data set. It is therefore proposed as future work.

7.2 Security Awareness

In order to understand whether the proposed system has an impact on security awareness, the following tasks could be conducted:

1. Conduct an initial simulated phishing campaign
2. Deploy the system, have it running for a given period of time
3. Conduct a post-deployment simulated phishing campaign
4. Compare the statistics of the two campaigns to see if or how the numbers have changed

7.3 Serverless

To determine whether serverless architecture is a viable option, in terms of both cost and reliability, an experiment should be conducted. For example, the main tasks could involve:

- **Deploying the system:**
  - On serverless architecture
  - On a traditional server
- **Comparison of:**
  - Execution time (incl. delay of initiating serverless architecture)
  - Reliability (availability, stability)
  - Running costs

7.4 Data Collection

With the sophistication of malicious email today compared to previous years, it would be useful to see if contemporary training data improves the accuracy of the classifier when applied in the "real world". Thus, we propose the collection of contemporary email is conducted. As this usually involves the collection of personal data, there may be a need to take legal aspects of privacy into account (depending on the setting).

Prioritizing future work We conclude this chapter with a proposal on how the tasks in the sections above could be prioritized.

1. Build a classifier using the full data set (Table 5.19)
2. Implement functionality to provide a confidence level for each prediction
3. Implement machine translation to support multiple languages
4. Collect new training data from the "real world"
5. Deploy the system on serverless architecture
6. Assess security awareness
7. Implement attachment analysis functionality

Chapter 8

Conclusion

In this thesis, we have shown how it is possible to automate the process of triaging and remediating user incident reports using machine learning. We also proved that it is possible to generate metrics representing trends in email attack methods. Although the ML classifier had a poor performance in a real-life setting (the classifier was biased towards phishing), we believe that a more contemporary and balanced data set could improve its performance.

As for the effect on security awareness, we were unable to complete the experiments to measure if the system had any impact; but the previous research presented in Section 4.3 indicates that it is possible.

Bibliography

[1] IEEE Computer Society, 'S1M Taxonomy', (n.d.) [Online]. Available: https://ieeecs-media.computer.org/assets/pdf/taxonomy.pdf.

[2] Wikipedia, *Anti-spam techniques*, Accessed: 25 November 2019, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Anti-spam_techniques.

[3] Y. Fang, C. Zhang, C. Huang, L. Liu and Y. Yang, 'Phishing Email Detection Using Improved RCNN Model With Multilevel Vectors and Attention Mechanism', *IEEE Access*, vol. 7, pp. 56 329–56 340, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2913705.

[4] A. Akinyelu and A. Adewumi, 'Classification of Phishing Email Using Random Forest Machine Learning Technique', *Journal of Applied Mathematics*, vol. 2014, Apr. 2014. [Online]. Available: http://downloads.hindawi.com/journals/jam/2014/425731.pdf.

[5] A. Cohen, N. Nissim and Y. Elovici, 'Novel set of general descriptive features for enhanced detection of malicious emails using machine learning methods', *Expert Systems with Applications*, vol. 110, pp. 143–169, 2018. DOI: https://doi.org/10.1016/j.eswa.2018.05.031. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418303312.

[6] Y. Song, A. Kołcz and C. L. Giles, 'Better Naive Bayes classification for high-precision spam detection', *Software: Practice and Experience*, vol. 39, no. 11, pp. 1003–1024, 2009. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.925.

[7] W. Li and W. Meng, 'An empirical study on email classification using supervised machine learning in real environments', in *2015 IEEE International Conference on Communications (ICC)*, Jun. 2015, pp. 7438–7443. [Online]. Available: https://ieeexplore.ieee.org/document/7249515.

[8] E. Jiang, 'Content-Based Spam Email Classification using Machine-Learning Algorithms', in *Text Mining: Applications and Theory*, John Wiley and Sons, 2010, pp. 37–56, ISBN: 9780470749821. [Online]. Available: https://ebookcentral.proquest.com/lib/ntnu/detail.action?docID=496056.

[9] APWG, *APWG: Phishing Activity Trends Report 1st Quarter 2019*, May 2019. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q1_2019.pdf.

[10] Trustwave, *2019 Trustwave Global Security Report*, Apr. 2019. [Online]. Available: https://securenation.net/wp-content/uploads/2018/12/Trustwave_GSR_2019_Final-.pdf.

[11] IBM Security, *2019 Cost of a Data Breach Report*, 2019. [Online]. Available: https://www.ibm.com/downloads/cas/ZBZLY7KL.

[12] Cofense, *2019 The Role of Automation in Effective Phishing Defense*, 2019. [Online]. Available: https://cofense.com/wp-content/uploads/2019/08/2019_Role-of-automation-eBook.pdf.

[13] Wikipedia, *Email spoofing*, Accessed: 26 April 2020, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Email_spoofing.

[14] E. Volkman, *More Than Half of Phishing Sites Now Use HTTPS*, Accessed: 25 April 2020, Jun. 2019. [Online]. Available: https://info.phishlabs.com/blog/more-than-half-of-phishing-sites-use-https.

[15] J. Birkeland, 'Hundretusener av nordmenn forsøkt koronasvindlet', *Computerworld*, Apr. 2020, Accessed: 26 April 2020. (In Norwegian). [Online]. Available: https://www.cw.no/artikkel/sikkerhet/hundretusener-av-nordmenn-forsokt-koronasvindlet.

[16] J. Cohen, 'Phishing Attacks Increase 350 Percent Amid COVID-19 Quarantine', *PCMag*, Mar. 2020, Accessed: 26 April 2020. [Online]. Available: https://uk.pcmag.com/antispam/125444/phishing-attacks-increase-350-percent-amid-covid-19-quarantine.

[17] J. Azara, 'The billing scam that cost 2 tech giants $122 million', *CFO Daily News*, Apr. 2019, Accessed: 26 April 2020. [Online]. Available: https://www.cfodailynews.com/news/billing-scam/.

[18] J. A. Miller, 'Why gifts cards are the new favorite target for fraud', *CIO*, May 2017, Accessed: 26 April 2020. [Online]. Available: https://www.cio.com/article/3193653/why-gifts-cards-are-the-new-favorite-target-for-fraud.html.

[19] scikit-learn, '3.5. Validation curves: plotting scores to evaluate models', *scikit-learn 0.23.1 documentation*, 2019, Accessed: 22 May 2020. [Online]. Available: https://scikit-learn.org/stable/modules/learning_curve.html#learning-curve.

[20] scikit-learn, 'API Reference', *scikit-learn 0.23.1 documentation*, 2019, Accessed: 22 May 2020. [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.

[21] scikit-learn, '3.1. Cross-validation: evaluating estimator performance', *scikit-learn 0.23.1 documentation*, 2019, Accessed: 20 May 2020. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation.

[22] Wikipedia, *Natural language processing*, Accessed: 26 April 2020, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Natural_language_processing.

[23] C. D. Manning, P. Raghavan and H. Schütze, 'Stemming and lemmatization', in *Introduction to Information Retrieval*, M. Broy and E. Denert, Eds. Cambridge University Press, 2008, pp. 32–36. DOI: 10.1017/CBO9780511809071. [Online]. Available: https://doi.org/10.1017/CBO9780511809071.

[24] John C. Chen and Dexter Whittinghill and Jennifer Kadlowec, 'Using Rapid Feedback to Enhance Student Learning and Satisfaction', Dec. 2006, pp. 13–18. DOI: 10.1109/FIE.2006.322306.

[25] Rachel J. Katz-Sidlow and Tamar G. Baer and Jeffrey C. Gershel, 'Providing rapid feedback to residents on their teaching skills: an educational strategy for contemporary trainees', IJME, Mar. 2016, pp. 83–86. DOI: 10.5116/ijme.56dc.908a. [Online]. Available: https://doi.org/10.5116/ijme.56dc.908a.

[26] K. Jansson and R. von Solms, 'Simulating malicious emails to educate end users on-demand', in *2011 3rd Symposium on Web Society*, Oct. 2011, pp. 74–80, ISBN: 978-1-4577-0211-2. DOI: 10.1109/SWS.2011.6101274. [Online]. Available: https://doi.org/10.1109/SWS.2011.6101274.

[27] L. A. Shepherd, J. Archibald and R. I. Ferguson, 'Assessing the Impact of Affective Feedback on End-User Security Awareness', in *Human Aspects of Information Security, Privacy and Trust*, T. Tryfonas, Ed., Cham: Springer International Publishing, 2017, pp. 143–159, ISBN: 978-3-319-58460-7.

[28] M. G. Ikhsan and K. Ramli, 'Measuring the Information Security Awareness Level of Government Employees Through Phishing Assessment', in *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, IEEE, Jun. 2019, pp. 1–4. DOI: 10.1109/ITC-CSCC.2019.8793292.

[29] Jemal Abawajy and Tai-hoon Kim, 'Performance Analysis of Cyber Security Awareness Delivery Methods', in *Security Technology, Disaster Recovery and Business Continuity*, Tai-hoon Kim and Wai-chi Fang and Muhammad Khurram Khan and Kirk P. Arnett and Heau-jo Kang and Dominik Ślęzak, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 142–148, ISBN: 978-3-642-17610-4. DOI: 10.1007/978-3-642-17610-4_16.

[30] G. Assenza, A. Chittaro, M. C. D. Maggio, M. Mastrapasqua and R. Setola, 'A Review of Methods for Evaluating Security Awareness Initiatives', *European Journal for Security Research*, Sep. 2019. DOI: 10.1007/s41125-019-00052-x.

[31] Ronald C. Dodge and Aaron J. Ferguson, 'Using Phishing for User Email Security Awareness', in *Security and Privacy in Dynamic Environments*, Simone Fischer-Hübner and Kai Rannenberg and Louise Yngström and Stefan Lindskog, Ed., Boston, MA: Springer US, 2006, 454–459, ISBN: 978-0-387-33406-6. DOI: 10.1007/0-387-33406-8_41.

[32] T. Asghar, S. Rasool, M. Iqbal, Z. u. Qayyum, A. N. Mian and G. Ubakanma, 'Feasibility of serverless cloud services for disaster management information systems', in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Jun. 2018, pp. 1054–1057. DOI: 10.1109/HPCC/SmartCity/DSS.2018.00175.

[33] L. Feng, P. Kudva, D. Da Silva and J. Hu, 'Exploring Serverless Computing for Neural Network Training', in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, Jul. 2018, pp. 334–341. DOI: 10.1109/CLOUD.2018.00049.

[34] H. Wang, D. Niu and B. Li, 'Distributed Machine Learning with a Serverless Architecture', in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Apr. 2019, pp. 1288–1296. DOI: 10.1109/INFOCOM.2019.8737391.

[35] A. Deese, 'Implementation of Unsupervised k-Means Clustering Algorithm Within Amazon Web Services Lambda', in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2018, pp. 626–632. DOI: 10.1109/CCGRID.2018.00093.

[36] M. N. Krasnyanskiy and A. D. Obukhov and E. M. Solomatina, 'The Algorithm of Document Classification of Research and Education Institution Using Machine Learning Methods', in *2019 International Science and Technology Conference "EastConf"*, 2019, pp. 1–6, ISBN: 978-1-7281-1931-1. DOI: 10.1109/EastConf.2019.8725319.

[37] Nur Syafiqah Mohd Nafis and Suryanti Awang, 'The Impact of Pre-processing and Feature Selection on Text Classification', in *Advances in Electronics Engineering*, Zahriladha Zakaria and Rabiah Ahmad, Ed., Singapore: Springer Singapore, 2020, pp. 269–280, ISBN: 978-981-15-1289-6. DOI: 10.1007/978-981-15-1289-6_25.

[38] Alper Kursat Uysal and Serkan Gunal, 'The impact of preprocessing on text classification', *Information Processing & Management*, vol. 50, no. 1, pp. 104–112, 2014, ISSN: 0306-4573. DOI: 10.1016/j.ipm.2013.08.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457313000964.

[39] P. A. Bonatti and S. Kirrane, 'Big Data and Analytics in the Age of the GDPR', in *2019 IEEE International Congress on Big Data (BigDataCongress)*, Jul. 2019, pp. 7–16. DOI: 10.1109/BigDataCongress.2019.00015.

[40] N. Gruschka, V. Mavroeidis, K. Vishi and M. Jensen, 'Privacy Issues and Data Protection in Big Data: A Case Study Analysis under GDPR', in *2018 IEEE International Conference on Big Data (Big Data)*, Dec. 2018, pp. 5027–5033. DOI: 10.1109/BigData.2018.8622621.

[41] C. Rustici, 'GDPR Profiling and Business Practice', *Computer Law Review International*, vol. 19, no. 2, pp. 34–43, 2018. [Online]. Available: https://search.proquest.com/docview/2061375839?accountid=12870.

[42] E. G. González and P. de Hert, 'Understanding the legal provisions that allow processing and profiling of personal data—an analysis of GDPR provisions and principles', *ERA Forum*, vol. 19, no. 4, pp. 597–621, Apr. 2019, ISSN: 1863-9038. DOI: 10.1007/s12027-018-0546-z.

[43] Frederike Kaltheuner and Elettra Bietti, 'Data is power: Towards additional guidance on profiling and automated decision-making in the GDPR', *Journal of Information Rights, Policy and Practice*, vol. 2, no. 2, 2018, ISSN: 2398-5437. DOI: 10.21039/irpandp.v2i2.45.

[44] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. O'Reilly Media, Inc., ISBN: 9781491962282.

[45] adeshpande3, *Machine-Learning-Links-And-Lessons-Learned*, https://github.com/adeshpande3/Machine-Learning-Links-And-Lessons-Learned, 2019.

[46] Y. Liu, *Python Machine Learning By Example*. Packt Publishing, May 2017, ISBN: 9781783553112.

[47] Microsoft, *nlp-recipes*, https://github.com/microsoft/nlp-recipes, 2019.

[48] R. Arumugam and R. Shanmugamani, *Hands-On Natural Language Processing with Python*. Packt Publishing, Jul. 2018, ISBN: 9781789139495.

[49] G. Bonaccorso, *Machine Learning Algorithms*. Packt Publishing, Jul. 2018, pp. 242–260, ISBN: 9781785889622.

[50] B. Gardner and V. Thomas, *Building an Information Security Awareness Program: Defending Against Social Engineering and Technical Threats 1st Edition*. Syngress, Aug. 2014, ISBN: ISBN-10: 0124199674 ISBN-13: 978-0124199675.

[51] D. L. Parnas, 'On the Criteria to Be Used in Decomposing Systems into Modules', in *Pioneers and Their Contributions to Software Engineering: sd&m Conference on Software Pioneers, Bonn, June 28/29, 2001, Original Historic Contributions*, M. Broy and E. Denert, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 479–498, ISBN: 978-3-642-48354-7. DOI: 10.1007/978-3-642-48354-7_20.

[52] S. Bird, E. Klein and E. Loper, *Natural Language Processing with Python*. O'Reilly Media, Inc., Jun. 2009, ISBN: 9780596516499.

[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, 'Scikit-learn: Machine Learning in Python', *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[54] A. Al-Masri, 'What Are Overfitting and Underfitting in Machine Learning?', *Towards Data Science*, 2019, Accessed: 20 April 2020. [Online]. Available: https://towardsdatascience.com/what-are-overfitting-and-underfitting-in-machine-learning-a96b30864690.

[55] G. Ciaburro and P. Joshi, *Python Machine Learning Cookbook - Second Edition*. Packt Publishing, Mar. 2019, ISBN: 9781789808452.

[56] scikit-learn, '1.9.2. Multinomial Naive Bayes', *1.9. Naive Bayes — scikit-learn 0.23.0 documentation*, 2019, Accessed: 1 May 2020. [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes.

[57] scikit-learn, '1.9.3. Complement Naive Bayes', *1.9. Naive Bayes — scikit-learn 0.23.0 documentation*, 2019, Accessed: 1 May 2020. [Online]. Available: https://scikit-learn.org/stable/modules/naive_bayes.html#complement-naive-bayes.

[58] W. van Lit, 'Email spam – ham and spam emails from spamassasin', *Kaggle*, 2019, Accessed: 20 March 2020. [Online]. Available: https://www.kaggle.com/veleon/ham-and-spam-dataset.

[59] V. Listik, 'Phishing corpus', Accessed: 20 March 2020. [Online]. Available: https://academictorrents.com/details/a77cda9a9d89a60dbdfbe581adf6e2df9197995a.

[60] V. Metsis, I. Androutsopoulos and G. Paliouras, 'Enron-Spam (GP) – Spam emails from the Enron data set', 2006, Accessed: 20 March 2020. [Online]. Available: http://nlp.cs.aueb.gr/software_and_datasets/Enron-Spam/raw/spam/GP.tar.gz.

[61] R. Tatman, 'Fraudulent e-mail corpus', 2017, Accessed: 26 March 2020. [Online]. Available: https://www.kaggle.com/rtatman/fraudulent-email-corpus.

[62] D. Radev, 'Clair collection of fraud email', *ACL Data and Code Repository*, 2008, ADCR2008T001. [Online]. Available: http://aclweb.org/aclwiki.

[63] Google, 'Imbalanced Data', *Data Preparation and Feature Engineering for Machine Learning*, 2020, Accessed: 03 March 2020. [Online]. Available: https://developers.google.com/machine-learning/data-prep/construct/sampling-splitting/imbalanced-data.

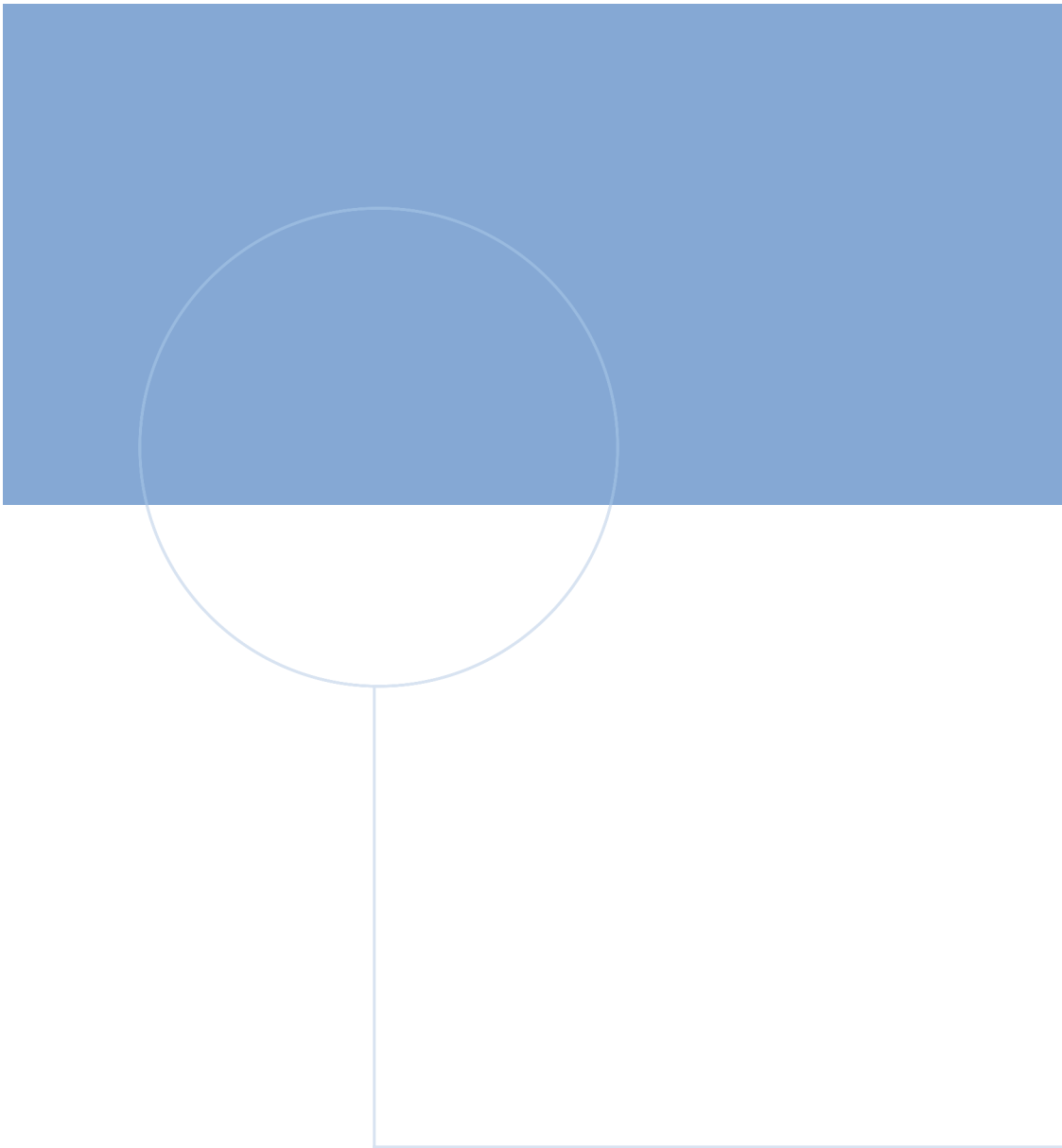
[64] Xiaopeng Zhang, 'Microsoft Excel Files Increasingly Used To Spread Malware', *Fortinet Threat Research*, Mar. 2017, Accessed: 25 May 2020. [Online]. Available: https://www.fortinet.com/blog/threat-research/microsoft-excel-files-increasingly-used-to-spread-malware.html.

[65] Trend Micro Incorporated, 'Common file types used by malware as email attachment', Mar. 2019, Accessed: 25 May 2020. [Online]. Available: https://success.trendmicro.com/solution/1101849-common-file-types-used-by-malware-as-email-attachment.

Appendix A

Acronyms and Abbreviations

- **APT:** Advanced Persistent Threat
- **AI:** Artificial Intelligence
- **DKIM:** DomainKeys Identified Mail
- **DMARC:** Domain-based Message Authentication, Reporting and Conformance
- **GDPR:** General Data Protection Regulation
- **HTML:** Hypertext Markup Language
- **HTTP:** Hypertext Transfer Protocol
- **HTTPS:** Hypertext Transfer Protocol Secure
- **IEEE:** Institute of Electrical and Electronics Engineers
- **InfoSec:** Information Security
- **IP:** Internet Protocol
- **KPI:** Key Performance Indicator
- **ML:** Machine Learning
- **NB:** Naïve Bayes
- **NLP:** Natural Language Processing
- **PII:** Personally Identifiable Information
- **PoC:** Proof of Concept
- **TF-IDF:** Term Frequency-Inverse Document Frequency
- **SPF:** Sender Policy Framework
- **URL:** Uniform Resource Locator
- **WVC:** Word Vector Counts



NTNU
Norwegian University of
Science and Technology

sportradar