Cyber Risk Detection Using Machine Learning Algorithms

Tintswalo Kissey Mhelembe (tintswalo@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Professor Phillip Mashele
North West University & African Institute for Mathematical Sciences

22 September 2020

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

In the last decade, the usage of technology has grown rapidly. As technology becomes part of our daily use, cybercrime incidents such as identity theft, hacking and loss of personal information due to visiting websites has also taken place. Different organizations around the world fall prey to sophisticated cyber attacks. This project uses different Machine Learning algorithms such as the Naive Bayes, K-means and the Support Vector Machine (SVM) to detect different cyber risks such as theft, hacking/IT incidents and loss. We compare these algorithms to see which one detects the cyber risk better and also point out which cyber risk is encountered the most by companies. In doing this, we found that the dataset was imbalanced, so we resampled the dataset by using the SMOTE, Random Oversampling and SMOTE+TOMEK Links techniques. The evaluation metrics used to evaluate the efficiency of the algorithms is Precision, Recall and $F1$-score. Experimental results show that theft is the cyber breach that most organizations are currently facing and the K-means algorithm is better at detecting cyber risk.

Keywords: Cyber risk, Naive Bayes, K-means, Support Vector Machine, SMOTE, Random Oversampling, TOMEK Links, Precision, Recall, $F1$-score

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

Tintswalo Kissey Mhelembe, 22 September 2020

Contents

1. Introduction

Technology continues to grow rapidly daily and it is taking over the world. Today, technology comes in the form of education, communications and to balance our daily lives. Decision making, business and, relationships are fast depending on the emerging technologies around us. Moreover, with the present COVID-19 pandemic, the use of technology cannot be overemphasized, from conference meetings to online classes, online shopping, and so on. Despite the importance of online activities that technology is bringing, we cannot forget that we are working on an insecure network that attracts many cybercrimes. Evaluating and managing the risk involved with online transactions has become a matter of urgency, which is a challenge that varies from one company to another. In this project, we use different machine learning algorithms to predict the different cyber breaches that companies are most likely to encounter. We compare different machine learning algorithms to see which one can make better predictions.

1.1 Background

The history of cybersecurity started with a research project at 1971. Ray Tomlison tried to improve a program and made it self-replicating. This program was the first worm. He then created another program that was the first antivirus software. This antivirus was to erase the creeper (Sentinel One,2020).

In 1989, Robert Morris created a computer worm that slowed down the internet. This worm was the first Denial of Service (DoS) attack in history. When Morris created this worm, his intentions were not to cause harm, but he wanted to highlight security flaws like weak passwords in Unix send mail. Instead, the worm replicated and caused a lot of damages (Orman, 2003). In the same year, a biologist named Joseph Popp created the first ransomeware attack called the AIDS Trojan. Joseph distributed this ransomeware through a floppy disk to the attendees of the Word Health Organization's AIDS conference. Joseph wanted to extort money from people (O'Kane, Sezer, and Carlin, 2018).

In 1990, the first legislation, called the Computer Misuse Act came into place. This act passed in the United Kingdom and made any unauthorized attempts to access computer systems illegal. In 1999, Microsoft Windows 98 came out and ushered in an exceedingly whole new accessibility. People started using computers more often and this made software security systems common. Furthermore, Windows released many patches and commercial products. Also, several security vendors discharged anti-hacking computer code for computing device usage (CyberExperst,2020).

In 2003, an international hacktivist group called Anonymous started. They were known for a spread of cyber attacks against the government and many organizations. From 2000, going onwards is known as the era for significant breaches. In 2013, a former CIA employee Edward Snowman copied and leaked confidential information from the National Security Agency (NSA). Edward wanted to show people in the US that the government was spying on them (CyberExperst,2020).

From 2013-2014, hackers broke into Yahoo and threatened the personal information of their 3 billion users. Yahoo was fined 35 million dollars for failing to disclose news of the breach on time. Yahoo's sale price decreased by 350 million dollars. In 2017, a ransomware attack known as the WannaCry took place. Computers that used Microsoft Windows were primary targets. These hackers demanded ransom payments in Bitcoin. In a day, this worm had affected over 230,000 in computers across 150 countries (CyberExperst,2020). Today, there are over a hundred reports of cybercrime incidents daily. These reports show that every 39 second, people are hacked across the globe. By estimation, in 2021, cybercrime will be triple than it is now.

1.2 Existing Work

(Ch, Gadekallu, Abidi, and Al-Ahmari, 2020) proposed a method called K-means to analyse the rate of cyber crime in India. This helped them to classify and analyse crime offences based on integrated data. Their focus was on finding the cyber attacks that take advantage of security vulnerabilities. They found out that Madhya Pradesh had the highest crime rate and that identity theft occurred the most.

Moreover, (Bouveret, 2018) classified the different types of cyber risk incidents in financial institutions globally. (Bouveret, 2018) identified patterns using a variety of datasets which he later provided to the institutions as a manageable framework. This framework is used to assess cyber risk. The results of his study show that cyber risk is a threat to all institutions.

(Rish et al., 2001) conducted an emperical study on the Naive Bayes classifier using the Monte Carlo simulations. The author analysed the impact of the distribution entropy on the classification error.(Rish et al., 2001) wanted to show that feature distributions with low entropy performed well for Naive Bayes. (Rish et al., 2001) also showed that Naive Bayes performed best while being completely independent on the features and also while being functionally depended on the features. On his study, he found out that the accuracy of Naive Bayes is not correlated with the degree of the feature dependencies but a better accuracy prediction is the loss of information that the features contain about a particular class.

1.3 Aims and Objectives

The main objective of this project is to use different machine learning algorithms and then choose the one that provides better detection of cyber risk. This aim will be accomplished by the following objectives:

1. Using the Term Frequency Inverse Document Frequency (TFIDF) to extract features.
2. Re-sampling the training dataset using SMOTE, Random Oversampling and SMOTE+TOMEK Links.
3. Performing the predictive analysis using three different machine learning algorithms such as the Naive Bayes, K-means and the Support Vector Machine.
4. Evaluating the efficiency of the algorithms using the different metrics such as precision, recall and $F1$-score.

1.4 Essay Layout

This essay is structured as follows: In Chapter 2 we analyse Cyber Risk, we define Cyber risk and give the different risks that organizations face and also ways in how to reduce the risk. We then give machine learning concepts followed by different Machine Learning algorithms used for this project. In Chapter 3 we have the methodology, this is where we explain the steps we took in achieving our results. In Chapter 4 we have the results for the different algorithms used. In Chapter 5 we have the conclusion and future work.

2. Preliminaries

In this chapter, we look at some concepts of Cyber Risk in Section 2.1, Machine Learning concepts in Section 2.2 that will be relevant to our work. Machine Learning algorithms such as the Naive Bayes, K-means and the Support Vector Machines (SVM) that are used in this project in Section 2.3.

2.1 Cyber Risks Concepts

Cyber Risk is defined as the exposure to harm or loss as a result of breaches or attacks on information systems (Cyber Risk Enterprise,2020). It can be simplified as a potential loss or harm caused by technical infrastructure or the use of technology in an organization. Risk is defined as the possibility of something bad happening and is given by the Equation 2.1.1 below.

$$\text{Risk} = \text{Probability of a loss} \times \text{Magnitude of a loss} \tag{2.1.1}$$

Cyber Risk shares characteristics with both property and liability risk when compared to the risk categories covered by insurance (Bouveret, 2018). Cyber attacks can impact organizations in the following three aspects of information security: Confidentiality, Integrity, and Availability.

Confidentiality refers to protecting information from being accessed by unauthorized parties (Scarfone, Jansen, and Tracy, 2008). Confidentiality issues arise when the information within the organization is shared/disclosed outside the organization, say to third parties which results in data breaches.

Integrity refers to ensuring the authenticity of information—that information is not altered, and that the source of the information is genuine (Scarfone, Jansen, and Tracy, 2008). Integrity issues occur when there is a misuse of systems, which may result in fraud (Bouveret, 2018).

Availability means that information is accessible by authorized users (Scarfone, Jansen, and Tracy, 2008). Availability issues are linked to business disruptions.

2.1.1 Types of Cyber Risks faced by Organizations.

There are different types of Cyber Risks that different organizations face these days.

1. **Ransomware** is when cybercriminals prevent access for valuable information that employees in an organization are supposed to work with. Cybercriminals do this in order to request payments in cryptocurrency from the organization. Even if access is gained, the organization is still at risk for losing the information again since their systems have proven to be weak.
2. **Hacking** is when cybercriminals try to access the organizations database or employees' computers. Hacking can be done manually in an organization or by breaching the organizations system.
3. **Malware** is a short term for Malicious software, cybercriminals install this software on the organizations systems and can be used to access personal information.
4. **Malicious code** is used by cybercriminals in a form of a code or link that is sent either by email or text messages to employees. This code comes in a form of downloadable files or attachment files that employees will follow or download then the organizations personal information will be in the hands of the cybercriminals.
5. **Phishing** is when cybercriminals pretend to belong to a certain company in order to gain access to the organizations system. These criminals use official letterheads in emails or even make phone calls to employees. In most cases, these type of attackers come from employees in the organization.

6. **Denial of service attacks** is when cybercriminals cause an overload on the organizations systems so that employees cannot gain access to it. This causes a slow down on the organizations system and may lead to financial losses.

7. **Outsourced company access**. This is when cybercriminals use the organizations network to get the organizations personal information. If criminals find a loop from the network then they can take advantage of it.

2.1.2 Ways to Reduce Cyber Risk.

1. **Train your team/staff about Cyber Risk:** Cybercriminals can get access to the organization through the employees, so it is advisable to train employees. Warn them about opening links sent to their emails and finding an email style that the organization can use to make them know whether the email they have received is legitimate.

2. **Information and Data Security:** Personal and valuable information should be stored appropriately with access control. It is more advisable for organization to be able to identify valuable information and access to that information should be controlled.

3. **Always keep your software and systems up to date:** Most of the time cyber attacks happen because the organizations systems are not up to date which may create weak points for these attackers to exploit.

4. **Passwords:** Using the same password for everything can be dangerous, once cybercriminals know the organizations password it will be easier for them to gain access to everything on the organizations system. It is safer to have different passwords for every application, and changing these passwords more often maintains high levels of protections.

5. **Control access to your systems:** It is essential for organizations to control who has access to their systems. Cybercriminals may come with USBs containing infected files and plug them into the organizations computer to gain personal information. It is more advisable for organizations to have systems that only read in information and cannot read out.

6. **Employee personal accounts:** Organizations should create login accounts for every application programme that they will be using. Using the same username for all employees can cause danger for the organization and this danger cannot be traced back to anyone since every employee was login to the same system.

7. **Web 2.0 Security:** Web 2.0 is a tool that organizations use these days to get new business connections, this tool helps organizations to communicate with a range of strategic allies, clients and prospects. Web 2.0 entities are FaceBook, Twitter, YouTube and Wikipedia.

2.2 Machine Learning Concepts

Machine Learning uses Artificial Intelligence(AI) that makes the system to be able to learn and improve automatically without being explicitly programmed. Artificial Intelligence refers to the simulation of human intelligence in machines that is programmed to think like humans and mimic their actions (Investopedia,2020).

In the learning process, the machine starts by observing the data first. This data can be in the form of examples, previous experience, or given instructions. The machine uses these examples to search for

patterns in the given data and then make predictions for the future. Machine Learning consists of three types of classes, namely Supervised Learning, Unsupervised Learning and Reinforcement Learning.

2.2.1 Supervised Machine Learning.

Supervised Machine Learning is a Machine Learning algorithm that learns from existing examples (Kotsiantis, Zaharakis, and Pintelas, 2007). Supervised Machine Learning aims to have a model that predicts a label when given features (Mohri, Rostamizadeh, and Talwalkar, 2018).

Some supervised machine learning algorithms are Classification(Logistic Regression, Naive Bayes, K-Nearest Neighbor, and Support Vector Machine) and Regression(Linear Regression, Ridge Regression, Ordinary Least Squares Regression, and Stepwise Regression).

2.2.2 Unsupervised Learning.

Unsupervised Machine Learning uses patterns from the data without knowing what the outcome will be (Data Robot,2020). The algorithm learns by itself the patterns and then come up with a conclusion based on what it learned.

Some types of Unsupervised Machine Learning algorithms are Clustering (K-means, K-median, Hierarchical Clustering, and Expectation Maximization) and Association Analysis (APRIORI, Eclat, and FP-Growth).

2.2.3 Reinforcement Learning.

Reinforcement Learning trains machine learning models to make a sequence of decisions, where the agent learns to achieve a goal in an uncertain environment (Sutton and Barto, 2018).

The reinforcement learning algorithms are Model-Free (Q-Learning, Hybrid, and Policy optimization) and Model-Based (Learn the model and Given the model)

2.3 Different Machine Learning Algorithms

For the classification, Naive Bayes and Support Vector Machine (SVM) are used and for the clustering, K-means is used. Below we give a full explanation of the algorithms.

2.3.1 Naive Bayes Classifier.

Naive Bayes is a supervised classification algorithm and it is based on the statistical Bayes' Theorem. It deals with probability distributions of the variables in the dataset and predicting the probability of the response variable belonging to a particular class, given the attributes of a new instance (Rish et al., 2001).

A good example of Naive Bayes would be filtering spam messages in an email. When a person receives an email, Naive Bayes can be used to classify it as to whether it goes into the spam folder or the inbox. Naive Bayes does this by checking the message subject, header, and the body, and if it consists of words that are classified as bad words on the training set it automatically classifies the email as a spam message.

Mathematically, we interpret Naive Bayes using the Bayes Theorem in the following way, let X be a new observation which is a feature vector $X = \{x_1, x_2, ..., x_n\}$ representing n features which are independent variables and c be the class of variables, then Bayes Theorem is given by Equation 2.3.1

$$P(c|X) = \frac{P(X|c)P(c)}{P(X)} \tag{2.3.1}$$

where

• $P(c|X)$ is the conditional probability that event c occurs given that X has already occurred, this is known as the posterior probability of a class c (which is the target) given the predictor X (which is the attribute)

• $P(c)$ is the prior probability of a class

• $P(X|c)$ is the conditional probability that event X occurs given that c has already occurred, this is the likelihood which is the probability of the predictor given class

• $P(X)$ is the prior probability of the predictor.

We can re-write the conditional probability in Equation 2.3.1 as a joint probability and it is given by Equation 2.3.2 below.

$$P(X|c) = P(x_1, x_2, \ldots, c) \tag{2.3.2}$$

We can decompose Equation 2.3.2 using the chain rule of probability, the reason why we are decomposing is because in Naive Bayes there is no relationship between the features and the class c and the values of the features are given so that the conditional probability is effectively constant.

$$\begin{gathered} P(x_1, x_2, \ldots, c) = P(x_1|x_2, \ldots, x_n, c)P(x_2|x_3, \ldots, x_n, c)\ldots P(x_{n-1}|x_n, c)P(x_n|c)P(x_n|c)P(c) \\ \therefore P(x_1, x_2, \ldots, c) = P(x_1|c)P(x_2|c)P(x_3|c)\ldots P(x_{n-1}|c)p(x_n|c)P(c) \end{gathered} \tag{2.3.3}$$

We can write the joint probability from Equation 2.3.3 as the product of the conditional probability, this is given by Equation 2.3.4 below.

$$P(x_1, x_2, \ldots, c) = \prod_{i=1}^{n} P(x_i|c)P(c) \tag{2.3.4}$$

Since Naive Bayes takes the highest probability when classifiying new data points, using the maximum posterior (MAP) we get Equation 2.3.5 below

$$C_{\text{MAP}} = \textbf{ arg max } p(c) \prod_{i=1}^{n} p(x_i|c) \tag{2.3.5}$$

where **arg max** returns the class with the largest predicted probability for a new data point.

Now we need to remove the features since they are independent to each other.

There are three types of Naive Bayes Models namely:

1. **Gaussian Naive Bayes:** Assumes that the features follow a normal distribution.
2. **Multinomial Naive Bayes:** Is used for discrete counts in document classification. The Multinomial Naive Bayes is the one used for this project since it is good for text classification.
3. **Bernoulli Naive Bayes:** The binomial model is useful for feature vectors that are in binary, which is in zeros and ones. Bernoulli Naive Bayes can be used to check if a word occurs in the text or not.

2.3.2 K-means Classifier.

K-means is an unsupervised clustering algorithm, it assigns data points to categories or clusters by finding the mean distance between data points (Medium,2020, b). K-means was chosen for this project because we want to cluster the cyber breaches into four clusters and then be able to do the prediction for the classes. With K-means it is easier to see which cluster has the most data points. K-means can be done in the following steps:

1. Select the number of clusters that you want to group the data into. In this project we selected four clusters.
2. Initialize cluster centres and place them at random positions.
3. Compute the distances of every data point to the nearest cluster centre and cluster them accordingly. In this step, every data point is being checked to find the closest cluster centre. The distance is calculated for every cluster centre and data points are clustered according to the shortest distance found. This is given by Equation 2.3.6 below

$$z_i \longleftarrow \textbf{arg min}||\mu_j - x_i||_2^2 \tag{2.3.6}$$

   where

   - z_i is the cluster label
   - **arg min** returns the index j of the cluster whose centre is closest to the observation x_i
   - μ_j is the j^{th} cluster centre
   - x_i is the i^{th} observation.

4. Update the position of the clusters passed on the values assigned to them (reverse cluster centres as the mean of the assigned observations). Here we calculate the mean of each cluster and re-assign the data points to the clusters and it is given by Equation 2.3.7 below

$$\mu_j = \frac{1}{n_j} \sum_{i:z_i=j} x_i \tag{2.3.7}$$

   where

   - n_j is the number of observations in cluster j
   - $\sum_{i:z_i=j}$ sums up all the observations in i such that $z_i = j$(observation i is in cluster j).

5. Repeat 3 and 4 until coverage.

A proper illustration of how K-means works is given by Figure 2.1 below.

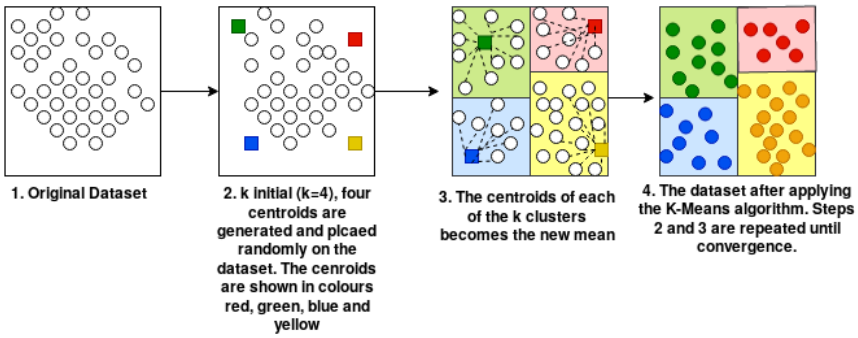

Figure 2.1: K-means

2.3.3 Support Vector Machine(SVM) Classifier.

Support Vector Machine (SVM) is a classification supervised machine learning algorithm. SVM can be used to analyze data for regression problems and classification problems (Tveit and Hetland, 2003). In this project, SVM is used for a classification problem. SVM was used in this project because it performs well on small datasets. The dataset used consists of 1055 observations which is a problem when different machine learning algorithms were used.

Support Vector Machine wants to creates a hyperplane such that when the classes are provided, all the data points must satisfy the hyperplane (Medium,2020, c). Given a dataset in Equation 2.3.8 with n vectors, we want the SVM to be able to predict if a new data point is a theft, loss, hacking/IT incidents or other incidents.

$$D = \{(x_n, y_n) | x_n \in \mathbb{R}^N, y_n \in \{-1, 1\}\}_i^N \tag{2.3.8}$$

Where $x_n \in \mathbb{R}^N$ is the n^{th} observation. Each x_n will be associated with a value y_n that indicates if the new data point belong to class $+1$ or class -1, $y_n \in \{-1, 1\}$. Since we are dealing with text classification, x_n is N dimensional.

SVM takes these existing data points and outputs a hyperplane which best separates these tags (Maimon and Rokach, 2010), the hyperplane is the decision boundary and its given by Equation 2.3.9 below

$$wx + b = 0 \tag{2.3.9}$$

where

- w is a weight vector $w = \{w_1, w_2, \ldots, w_N\}$
- b is the bias.

Assuming that the data points are linearly separable, SVM wants to maximize the distance between the hyperplane and the closest data point. The closest points are known as the support vectors. To do this, SVM wants to find a hyperplane that maximizes the margin. A bigger margin is better because it is easier to place a new data point on the correct side of the line (Maimon and Rokach, 2010). The margin is a distance from the hyperplane to a data point and it is given by Equation 2.3.10 below.

$$\text{margin} = \frac{1}{||w||} \tag{2.3.10}$$

Since the margin is given by Equation 2.3.10 and we want to maximize the margin, this now becomes an optimization problem (Kroon and Omlin, 2004). With the data points that are correctly classified

as our constraints.

$$\max \frac{1}{||w||} \tag{2.3.11}$$

$$\textbf{Subject to:} \quad \min_{n=1,\dots,N} |wx_n + b| = 1,$$

The constraints from Equation 2.3.11 becomes a problem because they are being minimized. Since SVM assumes that the data points are linearly separable, the hyperplane satisfies the Equation 2.3.12.

$$|wx_n + b| = y_n(wx_n + b) \text{ where } n \in \{1, \dots, N\} \tag{2.3.12}$$

Now this becomes a minimization optimization problem because we have an objective function and a constraint function, and thus we have the Equation 2.3.13 below.

$$\min \frac{1}{2}||w||^2 \tag{2.3.13}$$

$$\begin{aligned} \textbf{Subject to:} \quad & y_n(wx_n + b) \geq 1, \\ & \text{for } n = 1, 2, \dots, N, \\ & w \in \mathbb{R}^N, b \in \mathbb{R} \end{aligned}$$

Since the constraints have now become an inequality constraints, we will introduce the lagrangian multipliers $\alpha_n, n \in \{1, ..., N\}$. To solve this we take the inequality constraints and put them in a zero form, and multiply it with the lagrangian multiplier (Kroon and Omlin, 2004). Lagrange multipliers are auxilary variables that are introduced in a problem of constrained minimization in order to write first order optimality conditions formally as a system of equations (Rockafellar, 1993). Given by Equation 2.3.14 below.

$$\alpha_n(y_n(wx_n + b) - 1) \tag{2.3.14}$$

Now we minimize the primal form of the equation, with the constraints that are multiplied by the lagrangian multipliers in Equation 2.3.14 above. This is given by Equation 2.3.15 below.

$$\min \quad \mathcal{L}(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{n=1}^{N} \Big(\alpha_n(wx_n + b) - 1 \Big) \tag{2.3.15}$$

$$\begin{aligned} &\textbf{With respect to:} \ w \text{ and } b \\ &\text{and maximizing w.r.t } \alpha_n \geq 0 \end{aligned}$$

We maximize $\alpha_n \geq 0$ by taking its partial derivatives of Equation 2.3.15 with respect to the variables w and b and equate it to zero (Kroon and Omlin, 2004). Given by the Equations 2.3.16 and 2.3.17 below.

$$\frac{\partial \mathcal{L}(w, b, \alpha)}{\partial w} = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0 \implies w = \sum_{n=1}^{N} \alpha_n y_n x_n \tag{2.3.16}$$

$$\frac{\partial \mathcal{L}(w, b, \alpha)}{\partial b} = -\sum_{n=1}^{N} \alpha_n y_n = 0 \implies 0 = \sum_{n=1}^{N} \alpha_n y_n \tag{2.3.17}$$

Now we will check the relationship between the lagrangian multipliers and the constraints, since the constraints are inequality constraints we will be using the Karush-Kuhn-Tucker (KKT) condition (Kowalczyk, 2017). Substituting Equation 2.3.16 and Equation 2.3.17 into Equation 2.3.15, we get the dual formulation of the problem and it is given by Equation 2.3.18 below.

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2}\Big[\Big(\sum_{n=1}^{N}\alpha_n y_n x_n\Big)\Big(\sum_{n=1}^{N}\alpha_n y_n x_n\Big) - \Big(\sum_{n=1}^{N}\alpha_n y_n x_n\Big)\Big(\sum_{n=1}^{N}\alpha_n y_n x_n\Big)\Big] + \sum_{n=1}^{N}\alpha_n - \sum_{n=1}^{N}\alpha_n(y_n(b)) \tag{2.3.18}$$

$$\therefore \mathcal{L}(\alpha) = \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{M} y_n y_m \alpha_n \alpha_m x_n x_m$$

To solve the dual optimization problem in Equation 2.3.18, we use the quadratic programming package (Rhea,2020). We let α^* be the maximization solution while w^* and b^* are the parameters of the hyperplane. If we have a new data point x and we want to classify it, the decision function will be 2.3.19 below

$$sign(w^* x + b^*) = sign\Big(\sum_{n=1}^{N}\alpha^* y_n x_n . x + b^*\Big) \tag{2.3.19}$$

where

- $sign$ is the sign of the class, either -1 or $+1$
- $w^* = \sum_{N}^{n=1}\alpha_n^* y_n x_n$
- $b^* = y_n - w * x_n$.

For instances where the data points are not linearly separable we use the kernel trick. The kernel trick transforms the data points to a higher dimensional feature space where we can make the data points be linearly separable (Rhea,2020). The kernel trick maps the low dimensional input space X into a higher dimensional space Z, the mapping is shown by 2.3.20 below

$$K : X \to Z \tag{2.3.20}$$

where K is the kernel.

Sometimes the kernel trick becomes a problem when separating the data points, in this case we use the soft margin. Soft margin lets SVM makes mistakes so that the margin can be kept as wide as possible. In this way, data points can still be classified correctly (Medium,2020, a). If a data point is classified wrong, then the points that are far from the margin are given penalty. For every data point, we introduce a slack, the slack helps avoid overfitting and noisy data.

3. Methodology

This chapter consists of the steps that are involved in the predictive analysis. These steps includes Information Gathering (Data Description) given in Section 3.1, Preprocessing of the Dataset given in Section 3.2, Different Machine Learning Algorithms are used in Section 3.3, the Evaluation Metrics Section 3.4 and the Prediction Analysis is given in Section 3.5.

3.1 Information Gathering (Data Description)

The dataset used in this essay is collected from Kaggle. Figure. 3.1 shows the sample of the dataset. The dataset contains 1055 observations. From the 1055 records, 517 companies were affected by theft, 379 were affected by other incidents which were not specified, 85 were affected by loss and 75 were affected by hacking/IT incidents. The data consist of 10 features namely, name of the covered entity, state, individuals affected, date of breach, type of breach, location of breached information, date posted/updated, summary, breach start and the year.

	Name_of_Covered_Entity	State	Individuals_Affected	Date_of_Breach	Type_of_Breach	Location_of_Breached_Information	Date_Posted_or_Upd
0	Brooke Army Medical Center	TX	1000	10/16/2009	Theft	Paper	2014-
1	Mid America Kidney Stone Association, LLC	MO	1000	9/22/2009	Theft	Network Server	2014-
2	Alaska Department of Health and Social Services	AK	501	10/12/2009	Theft	Other Locations	2014-
3	Health Services for Children with Special Need...	DC	3800	10/9/2009	Loss	Laptop	2014-
4	L. Douglas Carlson, M.D.	CA	5257	9/27/2009	Theft	Desktop Computer	2014-

Figure 3.1: Sample of the processed dataset

3.2 Preprocessing the Dataset

The preprocessing step is involves cleaning the dataset and transforms it into a useful and usable when applying the different machine learning algorithms. This step involves the removal of columns in our dataset, and also fixing rows that have missing values and also techniques for handling imbalanced classes. Figure 3.2 gives a sample of the dataset before it was preprocessed. In this step, columns that are not useful are deleted. In the type of breach column and the location of breached information column, we have a repetition of information, this was resolved by grouping the repetition of information while not deleting useful information.

	Unnamed: 0	Number	Name_of_Covered_Entity	State	Business_Associate_Involved	Individuals_Affected	Date_of_Breach	Type_of_Breach	Locat
0	1	0	Brooke Army Medical Center	TX	NaN	1000	10/16/2009	Theft	
1	2	1	Mid America Kidney Stone Association, LLC	MO	NaN	1000	9/22/2009	Theft	
2	3	2	Alaska Department of Health and Social Services	AK	NaN	501	10/12/2009	Theft	Oth
3	4	3	Health Services for Children with Special Need...	DC	NaN	3800	10/9/2009	Loss	
4	5	4	L. Douglas Carlson, M.D.	CA	NaN	5257	9/27/2009	Theft	

Figure 3.2: Sample of the unprocessed dataset.

3.2.1 Techniques on handling missing data.

Missing data is when the data value is not stored on the dataset. The problem of missing data is that they can have an effect on the results that the algorithms produces. Below are the different ways in which missing data can be handled in a dataset.

1. Delete the rows or the columns.
2. Replace the missing data with the most frequent values. This can be problematic, it can lead to an imbalanced dataset within the categories.
3. Apply a classifier algorithm to predict the particular category of the missing row or column. You can do this by using the other columns as the features and the one consisting of missing data as the one you're predicting.
4. Apply an unsupervised machine learning technique.

Figure 3.2 is a sample of the dataset before the preprocessing was done. The figure shows that there are columns with missing data. Certain columns from the dataset were deleted because there was not even a single data point on the columns, this made the dataset unclean and not presentable.

The columns that had some missing values but still had useful information were not deleted but the empty rows were replaced with the most frequent values, but since it is text data they were replaced with the most frequent words in the column.

3.2.2 Feature Extraction.

For feature extraction, we will use the Term Frequency-Inverse Document Frequency (TF-IDF). The TF-IDF is part of Natural Language Processing (NLP). NLP is an automated way to understand and analyze natural human languages and extract information by applying machine learning algorithms (Collobert, Weston, Bottou, Karlen, Kavukcuoglu, and Kuksa, 2011). The TF-IDF uses word frequency information as a measure of the importance of feature items in the dataset (Wu and Yuan, 2018). Since Machine Learning algorithms work better with numbers, TF-IDF helps them by converting text data into numerical values or vectors.

3.2.3 Definition. (Ch, Gadekallu, Abidi, and Al-Ahmari, 2020) **Term Frequency** recapitulates how often a word has occurred in the given report and finds out the importance of each word in a document to the whole corpus. And it is given by Equation (3.2.1) below

$$tf(t,d) = \frac{n_t}{\sum_k n_k}, \tag{3.2.1}$$

where n_t is the number of words in the current document and n_k is the sum of all terms in the current document.

3.2.4 Definition. (Ch, Gadekallu, Abidi, and Al-Ahmari, 2020) **Inverse Frequency Document Frequency** downscales the given words that are present in the report that appeared many times. It is the log of the ratio of the number of all documents in the corpus to the number of documents with the specific term as seen in Equation (3.2.2) below.

$$idf(t,D) = \log \frac{|D|}{|di \in D | t \in di|} \tag{3.2.2}$$

The TF-IDF is the product of $tf(t,d)$ and $idf(t,D)$ given by Equation (3.2.3) below.

$$tf - idf(t,d,D) = tf(t,d) \times idf(t,D) \tag{3.2.3}$$

To find the TF-IDF we have to find the Term Frequency (TF) first and then store it in a matrix. We then evaluate the Inverse Document Frequency (IDF) for each term in the record and store it in another matrix. Then calculate the TF-IDF while considering the individual matrices of both the frequencies.

3.2.5 Data Splitting.

We split the dataset into 70% training set and 30% testing set for the models. Since the dataset was very small and highly imbalanced, the split of 70% and 30% was the best. Both the training and testing consist of different features like the Location of Breach, The State, The Summary and the Name of the Affected Company/Entity.

3.2.6 Techniques on solving an imbalanced classes.

Figure 3.3 below shows the four classes before the techniques for solving imbalanced classes are applied. We can see that Theft is the highest, followed by Other incidents, then Loss and Hacking/IT Incident is the lowest. This becomes a problem when applying different Machine Learning classifiers, they tend to learn the data based on the majority class and therefore predicts that class only.

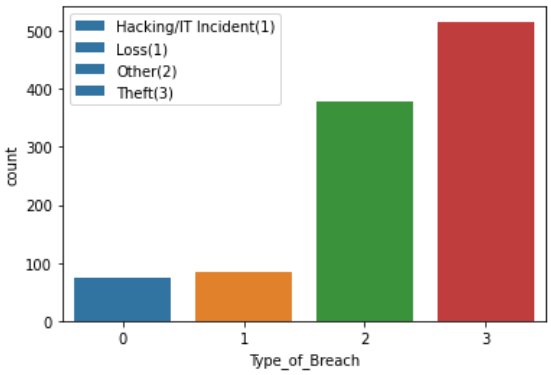

Figure 3.3: Imbalanced classifiers

The problem with imbalanced classes is that the algorithms only learn the majority classes and leave out the minority classes. This can be solved by either oversampling the minority class or undersampling the majority class depending on the size and type of the dataset. Since the dataset used for this project is small, the best technique to use would be to oversample the minority class, but we also experimented with both oversampling and undersampling. For oversampling we used the Synthetic Minority Oversampling Technique (SMOTE) and Random Oversampling techniques. We also used a combination of SMOTE+TOMEK Links technique.

1. SMOTE:

SMOTE is an oversampling technique, it creates artificial samples of the minority classes. In simple terms, SMOTE oversamples the minority classes. Below are the steps of how the SMOTE technique works.

1. Take the differences between the feature vector of an instance and the nearest neighbour.
2. Multiply this difference by a value that is between 0 and 1.
3. Form a new instance by adding the result to the original instance value.

Figure 3.4 below gives a clear illustration of how the SMOTE algorithm works.

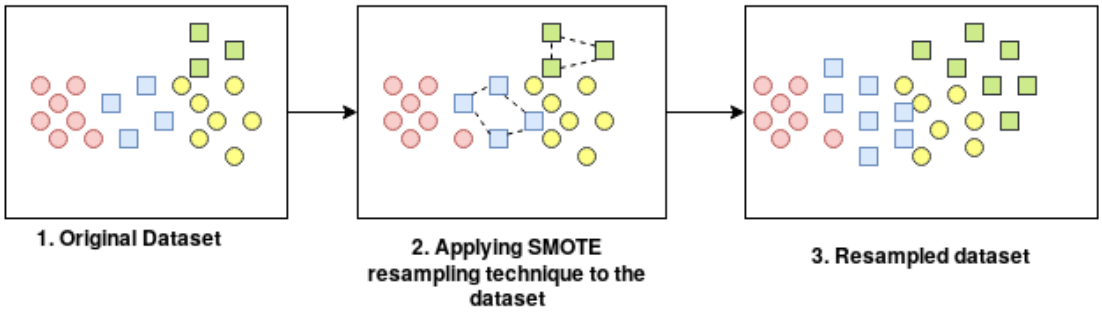

Figure 3.4: SMOTE Algorithm

2. Random Oversampling:

Random oversampling randomly creates a replica of the minority class examples. In simple words, it increases the size of the training data set through repetition of the original examples. It randomly selects a line of the minority class and adds it to the dataset until the minority classes are almost the same size as the majority classes. Figure 3.5 below gives a clear illustration of how the algorithm works.

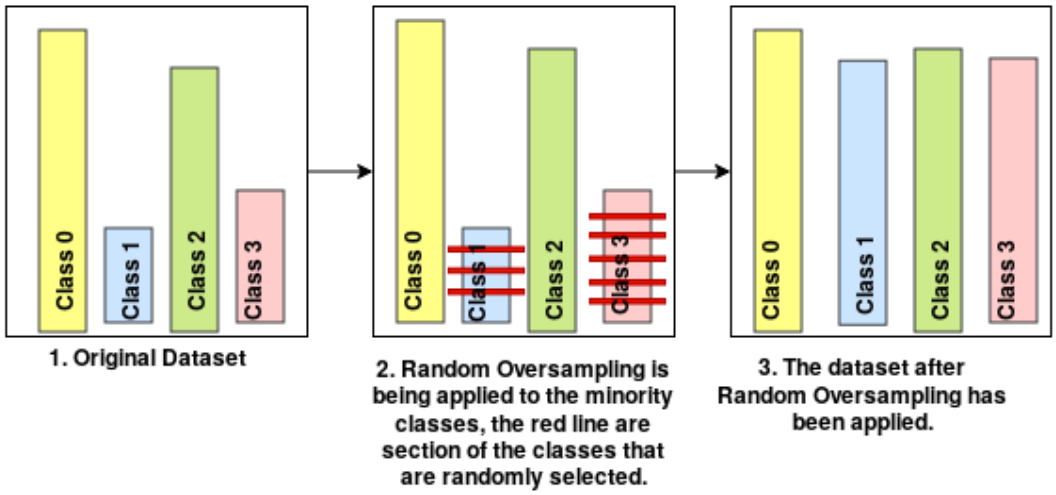

Figure 3.5: Random Oversampling

3. TOMEK Links:

Tomek Links is an undersampling method, for a dataset with some target class, a Tomek link is a pair of examples that are the nearest neighbors of one another and have different target class label (Boardman, Biron, and Rimbey, 2018). The Tomek Link removes only the majority classes.

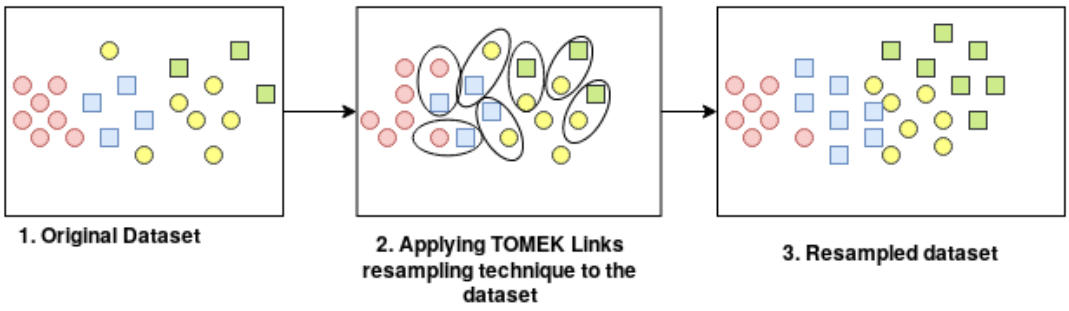

Figure 3.6: Tomek Links

4. SMOTE + TOMEK Links:

SMOTE + Tomek link is a combination of the preprocessing method whereby the two aforementioned re-sampling techniques are applied back-to-back (Boardman, Biron, and Rimbey, 2018). SMOTE is applied first to generate synthetic minority class examples, and, subsequently, Tomek link undersampling is applied to the dataset composed of both the original and new, synthetic observations.

3.2.7 Hyper-parameter Tuning.

The hyper-parameters are tuned manually and using the GridSearchCV. The GridSearchCV takes in a list of values of the hyper-parameters and the model goes through all the values in the list and chooses the one that is the best. Tuning the hyper-parameters manually means that we had to pick values and range them until the model gives the best output.

Hyper-parameter tuning in K-means

In K-means the hyper-parameters are tuned manually, the number of clusters were tunned ($n\_clusters$) which was given the value 4 since we want 4 clusters. The initialization method was also tuned, and for this, K-means++ is chosen for this process, K-means++ selects initial cluster centers for K-means clustering in a smart way to speed up convergence. The number of times the K-means algorithm had to run in different centroids was tuned, which is the n_init. Lastly, we tune maximum number of iterations the K-means algorithm uses, which is the max_iter.

Hyper-parameter tuning for SVM.

In SVM, the hyper-parameters are tuned using the GridSearchCV. The kernel, C and γ(gamma) are tuned. The the γ parameter defines how far the influence of a single training example reaches, C is the cost of penalty of the misclassification. Small values between 0.05 - 1 were given to γ and C because we wanted to maximize the margin. The GridSearchCV chooses the optimal values that works best for the SVM. Lastly we had to tune the kernel using Linear and Radial Basis Function (RBF), then GridSearchCV chooses the one that works best.

3.3 Different Machine Learning Algorithms Used

Since we want to classify, we used two supervised classification learning algorithms, which is the Naive Bayes and Support Vector Machine. These two supervised algorithms were used for this project because they work well with small datasets and text data. We used unsupervised clustering algorithm known as the $K-$means in order to check if the model works without given examples.

3.4 Evaluation Metrics

Our dataset is small and highly imbalanced, accuracy will not be used because it can either be very low or can favour the majority classes. Below we give the different metrics that are used to evaluate the efficiency of the algorithms.

3.4.1 Confusion Metrics.

A confusion matrix is an $N \times N$ matrix used for evaluating the performance of a classification model, where N is the number of target classes (Analytics Vidhya,2020). The confusion matrix compares the values predicted by the machine learning model and the actual target values. This helps us see how our model is performing and the errors that it is making. The confusion matrix is a very important metric for it gives the other metrics. Figure 3.7 gives a graphical illustration of the confusion matrix where

• **True Positive (TP):** Is the predicted value that matches the actual value, the observations that were predicted to be positive and were actually positive. In this case, predicting Theft as Theft and it turns out to be true.

• **True Negative (TN):** When the actual value was predicted to be negative and were actually negative.

• **False Positive (FP):** This is also known as the $Type-1$ Error, the actual value was negative but the model predicted it as a positive value. Example, Theft predicted as negative while it was supposed to be a positive value.

• **False Negative (FN):** This is also known as the $Type-2$ Error, the actual value was positive but the model predicted it as a negative value.

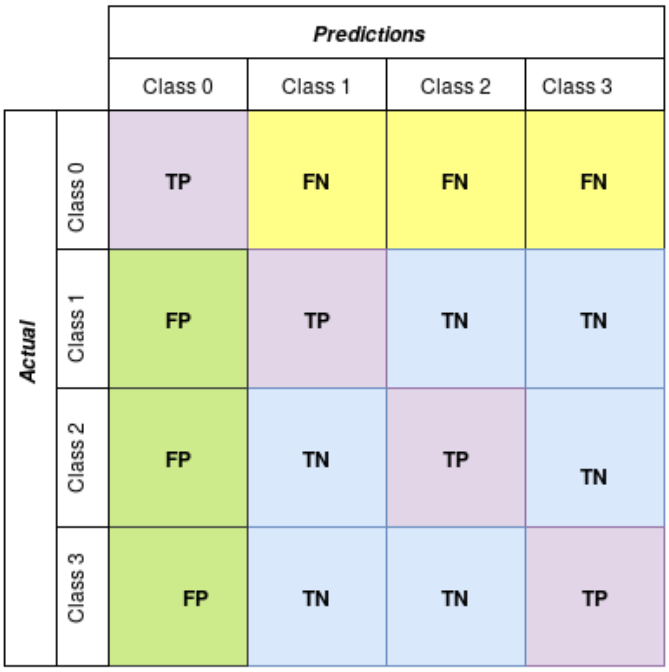

Figure 3.7: Confusion Matrix

3.4.2 Precision.

Precision gives the percentage of the results that were correctly identified (Analytics Vidhya,2020). The equation of Precision is given by Equation 3.4.1 below.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.4.1}$$

3.4.3 Recall.

Recall tells us how many of the actual positive cases we were able to predict correctly with our model (Analytics Vidhya,2020). The equation of Recall is given by Equation 3.4.2 below.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.4.2}$$

3.4.4 F1-Score.

$F1$-Score shows the balance between Precision and Recall and it is given by Equation 3.4.3 below.

$$F1\text{-Score} = 2 * \left(\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \right) \tag{3.4.3}$$

3.5 Prediction Analysis

In the prediction analysis step, the cybercrime data were analysed and used to predict which cyber breach is most likely to happen. We were predicting the type of breach that most companies are most likely to encounter and using different machine learning algorithms. Also,comparing which algorithm best predicts the cyber breach.

4. Results

In this Chapter, we will be discussing the results obtained from the algorithms used. To get the results, we used Recall, Precision and $F1$-Score to evaluate the algorithms' efficiency. Accuracy is not appropriate when the dataset is imbalanced (Chawla, Bowyer, Hall, and Kegelmeyer, 2002). The dataset was not balanced and consisted of 1055 observations, which was very small to train the algorithms, this resulted in the accuracy being very low. The algorithms produced poor results due to the problems stated above. The results given below are for the predicted classes where 0 represents hacking, 1 represents loss, 2 represents other incidents and 3 represents theft. All the code used can be found on the this link.

4.1 Naive Bayes

4.1.1 Unbalanced.

Table 4.1 below gives the evaluation matrix for Naive Bayes without any re-sampling technique applied for the predicted four classes. Figure 4.1 shows the confusion matrix for Naive Bayes. From Table 4.1, not all classes were predicted, Naive Bayes without balancing was only able to predict class 2 and class 3, and these classes are the majority classes on our dataset. For class 2, it could predict 34% and out of that, only 44% were predicted correctly with 39% of balance between Precision and Recall. For class 3, it was able to predict 83% and out of that, only 55% were predicted correctly with 66% of balance between Precision and Recall. The diagonal of the confusion matrix in Figure 4.1 shows the total number of correct predictions for the classes. It was able to correctly classify 40 correct predictions for class 2 and 124 correct predictions for class 3.

Classes	Precision	Recall	F1-Score
0	0.00	0.00	0.00
1	0.00	0.00	0.00
2	0.44	0.34	0.39
3	0.55	0.83	0.66

Table 4.1: Evaluation for unbalanced Naive Bayes

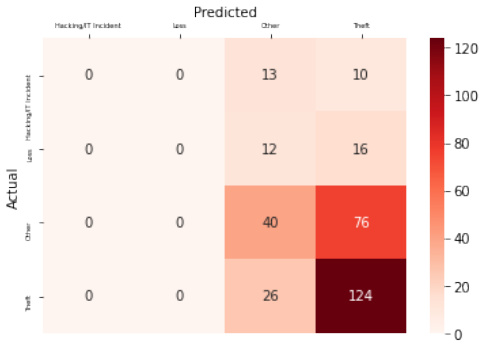

Figure 4.1: Confusion Matrix for Naive Bayes

4.1.2 SMOTE.

Table 4.2 shows the evaluation for Naive Bayes with the SMOTE re-sampling technique. Figure 4.2 gives the confusion matrix. After Naive Bayes has been re-sampled using SMOTE, it predicted all the classes. This algorithm has been improved from unbalanced to balanced with SMOTE. The improvement of the algorithm is evaluated using Table 4.2 and Figure 4.2, all the classes are predicted correctly. From Table 4.2, Naive Bayes with SMOTE was able to predict 61% of class 0 and out of that, only 28% of them were predicted correctly with a 38% of balance between Precision and Recall. This algorithm then predicted 39% of class 1, and only 13% were correct predictions with 20% of balance between Precision and Recall. In class 2, it predicted 44% and only 55% were predicted correctly with 49% of balance between Precision and Recall. In class 3, it predicted 48% and only 80% were predicted correctly with 60% of balance between Precision and Recall. The diagonal of the confusion matrix in

Figure 4.2 shows the total number of correct predictions for the classes. It was able to correctly classify 14 correct predictions for class 0, 11 correct predictions for class 1, 51 correct predictions for class 2 and 72 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.28	0.61	0.38
1	0.13	0.39	0.20
2	0.55	0.44	0.49
3	0.80	0.48	0.60

Table 4.2: Evaluation for Naive Bayes with SMOTE

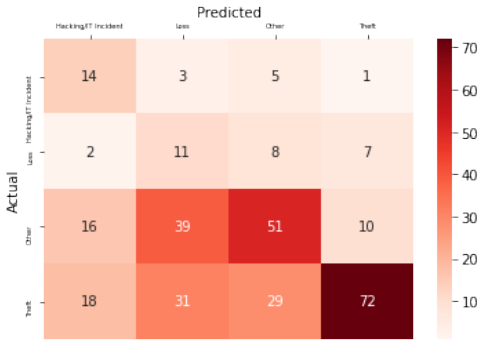

Figure 4.2: Confusion Matrix for Naive Bayes with SMOTE

4.1.3 Random Over Sampling.

Table 4.3 shows the evaluation for Naive Bayes with the Random Oversampling re-sampling technique. Figure 4.3 shows the confusion matrix. Naive Bayes with Random Oversampling was not able to predict all the classes. This shows that Random Oversampling was not a good re-sampling technique for Naive Bayes. This technique only predicted class 1, class 2 and class 3. For class 0, it did not predict anything. In class 1, it predicted 39% and only 12% were correct predictions with 18% of balance between the precision and recall. In class 2, it predicted 48% and only 47% were correct predictions of this class with 48% of balance between the precision and recall. In class 3, it predicted 53% and only 75% were correct predictions of this class with 62% of balance between the recall and precision. The diagonal line from the confusion matrix in Figure 4.3 shows the total number of correct predictions for the classes. Since it did not predict anything for class 0, we have a 0 for this class in the confusion matrix. It was able to correctly classify 11 correct predictions for class 1, 56 correct predictions for class 2 and 80 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.00	0.00	0.00
1	0.12	0.39	0.18
2	0.47	0.48	0.48
3	0.75	0.53	0.62

Table 4.3: Evaluation for Naive Bayes with Random Oversampling

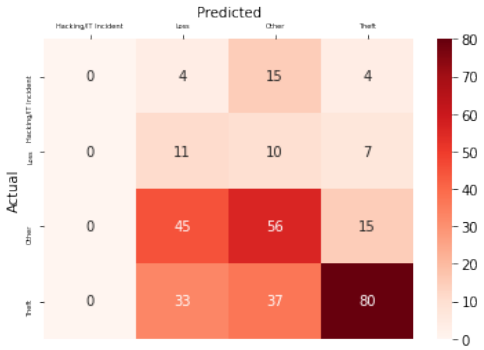

Figure 4.3: Confusion Matrix for Naive Bayes with Random Oversampling

4.1.4 SMOTE and TOMEK Links.

Table 4.4 below shows the evaluation matrix for Naive Bayes with the SMOTE+TOMEK Links re-sampling technique. Figure 4.4 shows the confusion matrix. This technique performed well. It predicted all the classes. In class 0, it predicted 70% and only 26% were correct predictions for this class with 38% of balance between Precision and Recall. In class 1, it predicted 43% and only 14% were correct predictions for this class with 21% of balance between Precision and Recall. In class 2, it predicted 41% and only 53% were correct predictions with 46% of balance between Precision and Recall. In class 3, it predicted 45% and only 81% were correct predictions with 58% of balance between Precision and Recall. The diagonal line from the confusion matrix in Figure 4.4 shows the total number of correct predictions for the classes. It was able to correctly classify 16 correct predictions for class 0, 12 correct predictions for class 1, 47 correct predictions for class 2 and 67 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.26	0.70	0.38
1	0.14	0.43	0.21
2	0.53	0.41	0.46
3	0.81	0.45	0.58

Table 4.4: Evaluation for Naive Bayes with SMOTE+TOMEK Links

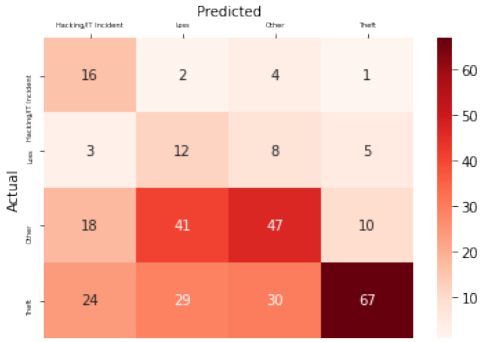

Figure 4.4: Confusion Matrix for Naive Bayes with SMOTE and TOMEK Links

4.2 K-Means

4.2.1 Unbalanced.

Table 4.5 below gives the evaluation for K-means without any re-sampling technique. Figure 2.1 gives the confusion matrix. K-means without balancing performed well since it predicted all four classes. Table 4.5 shows that for class 0, it predicted 52%, and only 8% were predicted correctly with 15% of balance between Precision and Recall. In class 1, it predicted 4% and only 3% were correct predictions for this class with 3% of balance between Precision and Recall. In class 2, it predicted 3% and only 11% were correct predictions for this class with 5% of balance between Precision and Recall. In class 3, it predicted 19% and only 28% were predicted correctly with 23% of balance between Precision and Recall. The diagonal line from the confusion matrix in Figure 2.1 shows the total number of correct predictions for the classes. It was able to correctly classify 12 correct predictions for class 0, 1 correct predictions for class 1, 4 correct predictions for class 2 and 29 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.08	0.52	0.15
1	0.03	0.04	0.03
2	0.11	0.03	0.05
3	0.28	0.19	0.23

Table 4.5: Evaluation for unbalanced K-means

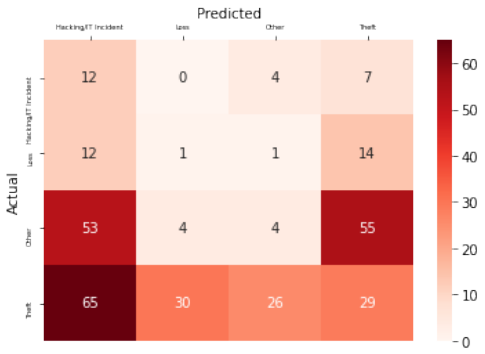

Figure 4.5: Confusion Matrix for unbalanced K-means

4.2.2 SMOTE.

Table 4.6 shows the efficiency of K-means with the SMOTE re-sampling technique. Figure 4.6 shows the confusion matrix. K-means with SMOTE was able to predict all four classes. In class 0, it predicted 30% and only 7% were correct predictions for the class with 11% of balance between Precision and Recall. In class 1, it predicted 4% and only 3% were correct predictions with 3% of balance between Precision and Recall. In class 2, it predicted 6% and only 29% were correct predictions for the class with 10% of balance between Precision and Recall. In class 3, it predicted 57% and only 56% were correct predictions for the class with 57% of balance between Precision and Recall. The diagonal line from the confusion matrix in Figure 2.1 shows the total number of correct predictions for the classes. It was able to correctly classify 7 correct predictions for class 0, 1 correct predictions for class 1, 7 correct predictions for class 2 and 86 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.07	0.30	0.11
1	0.03	0.04	0.03
2	0.29	0.06	0.10
3	0.56	0.57	0.57

Table 4.6: Evaluation for K-means with SMOTE

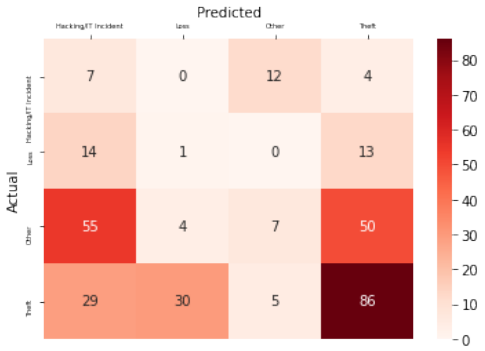

Figure 4.6: Confusion Matrix for K-means with SMOTE

4.2.3 Random Over Sampling.

Table 4.7 shows the efficiency of K-means with the Random Oversampling technique. Figure 4.7 shows the confusion matrix. This technique performed well since it predicted all the classes. In class 0, it predicted 65% and only 9% were correct predictions for this class with 15% of balance between Precision and Recall. In class 1, it predicted 4% and only 3% were correct predictions for this class with 3% of balance between Precision and Recall. In class 2, it predicted 2% and only 29% were correct predictions for this class with 3% of balance between Precision and Recall. In class 3, it predicted 19% and only 27% were correct predictions for this class with 22% of balance between Precision and Recall. The diagonal of the confusion matrix in Figure 4.7 shows the total number of correct predictions for

each class. It was able to correctly classify 15 correct predictions for class 0, 1 correct predictions for class 1, 2 correct predictions for class 2 and 28 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.09	0.65	0.15
1	0.03	0.04	0.03
2	0.29	0.02	0.03
3	0.27	0.19	0.22

Table 4.7: Evaluation for K-means with Random Oversampling

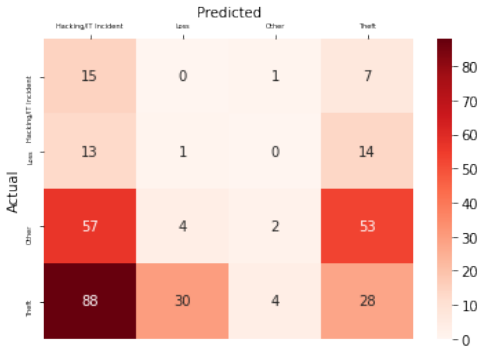

Figure 4.7: Confusion Matrix for K-means with Random Oversampling

4.2.4 SMOTE and TOMEK Links.

Table 4.8 show the efficiency for K-means with the SMOTE+TOMEK Links re-sampling technique. Figure 4.8 shows the confusion matrix. K-means with SMOTE+TOMEK Links performed well, it predicted all four classes. From Table 4.8 below shows that, for class 0, it was able to predict 17% of it and correctly predicted 3% of them with 5% of balance between Precision and Recall. In class 1, it predicted 50% and only 13% were correct predictions for this class with 21% of balance between Precision and Recall. In class 2, it predicted 6% and only 29% were correct predictions for this class with 10% of balance between Precision and Recall. In class 3, it predicted 20% and only 86% were correct predictions for this class with 32% of balance between Precision and Recall. The diagonal of the confusion matrix in Figure 4.8 shows the total number of correct predictions for each class. It was able to correctly classify 4 correct predictions for class 0, 14 correct predictions for class 1, 7 correct predictions for class 2 and 30 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.03	0.17	0.05
1	0.13	0.50	0.21
2	0.29	0.06	0.10
3	0.86	0.20	0.32

Table 4.8: Evaluation for K-means with SMOTE+TOMEK Links

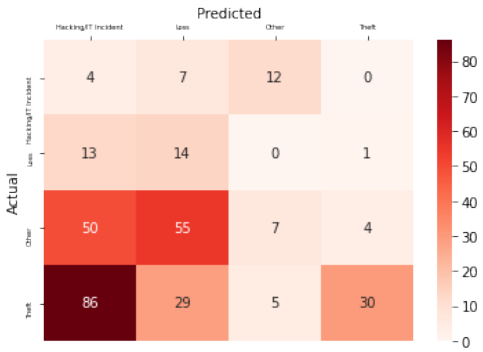

Figure 4.8: Confusion Matrix for K-means with SMOTE and TOMEK Links

4.3 Support Vector Machine

4.3.1 Unbalanced.

Table 4.9 shows the efficiency of SVM without applying any re-sampling techniques. Figure 4.9 shows the confusion matrix. SVM without the re-sampling techniques did not perform well. It did not predict all four classes. It was able to predict only class 2 and class 3 and these two are the majority classes on the dataset. In class 2 it predicted 87% and only 54% were correct predictions for this class with 67% of balance between Precision and Recall. In class 3, it predicted 72% and only 82% were correct predictions for this class with 77% of balance between Precision and Recall. The diagonal of the confusion matrix in Figure 4.9 shows the correct predictions of each class. In class 0 and class 1, there were no predictions. In class 2, only 101 were predicted correctly. For class 3, only 108 were correctly predicted.

Classes	Precision	Recall	$F1$-Score
0	0.00	0.00	0.00
1	0.00	0.00	0.00
2	0.54	0.87	0.67
3	0.82	0.72	0.77

Table 4.9: Evaluation for unbalanced SVM

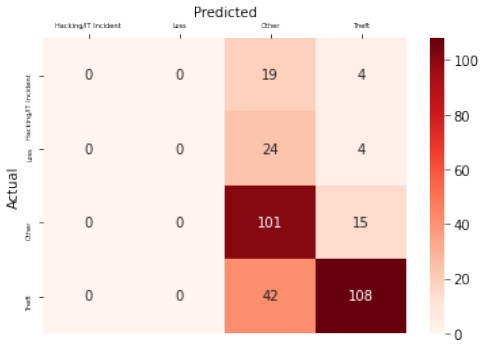

Figure 4.9: Confusion matrix for an unbalanced SVM

4.3.2 SMOTE.

Table 4.10 show the efficiency for SVM with the SMOTE re-sampling technique. Figure 4.10 shows the confusion matrix. SVM with SMOTE performed well, it was able to predict all four classes. From Table 4.10 below, for class 0, it predicted 4% and only 33% were correct predictions for this class with 8% of balance between Precision and Recall. In class 1, it predicted 4% and all of them predicted correctly, hence the 100% on precision with 7% of balance between Precision and Recall. For class 2, it predicted 86% and only 55% were correct predictions for this class with 67% of balance between Precision and Recall. In class 3, it predicted 71% and only 82% were correct predictions for this class with 76% of balance between Precision and Recall. The diagonal of the matrix in Figure 4.10 shows the correct predictions of each class. It was able to correctly classify 1 correct predictions for class 0, 1 correct predictions for class 1, 100 correct predictions for class 2 and 107 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.33	0.04	0.08
1	1.00	0.04	0.07
2	0.55	0.86	0.67
3	0.82	0.71	0.76

Table 4.10: Evaluation for SVM with SMOTE

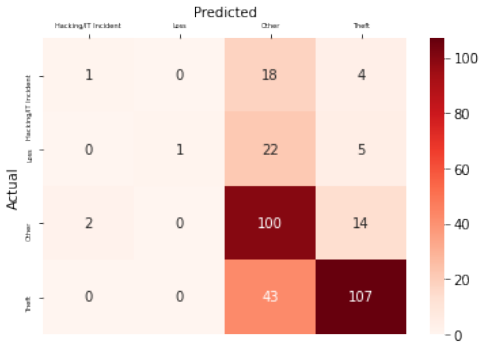

Figure 4.10: Confusion matrix for SVM with SMOTE

4.3.3 Random Over Sampling.

Table 4.11 show the efficiency for SVM with the Random Oversampling re-sampling technique. Figure 4.11 shows the confusion matrix. SVM with Random Oversampling did not perform well, since there were no predictions for class 0. From Table 4.11 below, in class 1, it predicted 4% and only 33% were correct predictions for this class with 6% of balance between Precision and Recall. In class 2, it predicted 88% and only 55% were correct predictions for this class with 67% of balance between Precision and Recall. In class 3, it predicted 70% and only 83% were correct predictions for this class with 76% of balance between Precision and Recall. The diagonal of the matrix in Figure 4.11 shows the correct predictions of each class. It was able to correctly classify 1 correct predictions for class 1, 102 correct predictions for class 2, 105 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.00	0.00	0.00
1	0.33	0.04	0.06
2	0.55	0.88	0.67
3	0.83	0.70	0.76

Table 4.11: Evaluation for SVM with Random Oversampling

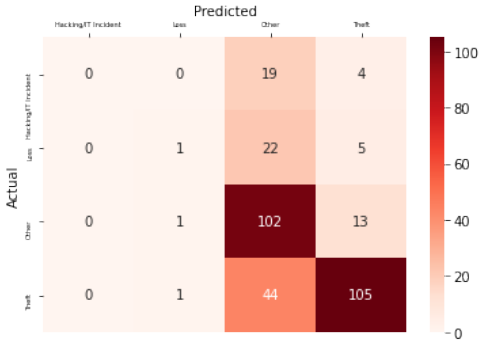

Figure 4.11: Confusion matrix for SVM with Random Oversampling

4.3.4 SMOTE and TOMEK Links.

Table 4.12 show the efficiency for SVM with the SMOTE+TOMEK Links re-sampling techniques. Figure 4.12 shows the confusion matrix. SVM with SMOTE+TOMEK Links did not perform well; it did not predict class 0. From Table 4.12 below shows that in class 0 there were no predictions. In class 1, it predicted 4% and only 33% were correct predictions for this class with 6% of balance between Precision and Recall. In class 2, it predicted 88% and only 55% were correct predictions for this class with 67% of balance between Precision and Recall. In class 3, it predicted 70% and only 83% were correct predictions for this class with 76% of balance between Precision and Recall. The diagonal of the matrix in Figure 4.10 shows the correct predictions of each class. It was able to correctly classify 1 correct predictions for class 1, 102 correct predictions for class 2, 105 correct predictions for class 3.

Classes	Precision	Recall	$F1$-Score
0	0.00	0.00	0.00
1	0.33	0.04	0.06
2	0.55	0.88	0.67
3	0.83	0.70	0.76

Table 4.12: Evaluation for SVM with SMOTE+TOMEK Links

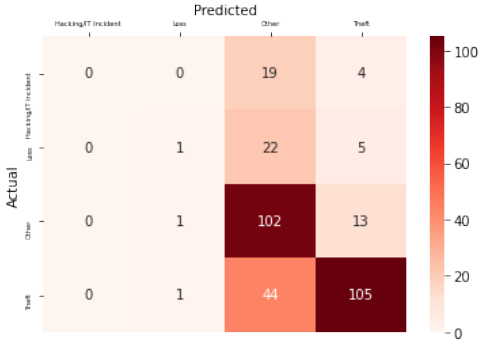

Figure 4.12: Confusion matrix for SVM with SMOTE+TOMEK Links

4.4 Comparison of the Algorithms

Table 4.13 gives a summary of all the algorithms together with the resampling techniques applied to them. Below we give a brief discussion of the algorithms based on Table 4.13.

Naive Bayes does not perform well when there are no resampling techniques applied. The Random Oversampling technique is not suitable for Naive Bayes. This technique could only predict class 1, class 2 and class 3 but not class 0. Class 0 was one of the minority classes, so Random Oversampling did not do a good job when oversampling the minority classes. Looking at SMOTE and SMOTE+TOMEK Links, these two resampling techniques performed well with Naive Bayes, for both techniques, all the classes were predicted. SMOTE outperforms SMOTE+TOMEK Links. Looking at the percentages of precision, SMOTE has the highest percentages when compared to SMOTE+TOMEK Links. This means that out of all the predictions made in recall, SMOTE was able to classify the four classes correctly.

K-means performed well with the re-sampling techniques applied and without the techniques. K-means on its own was able to predict all four classes and correctly classify them. When looking at the SMOTE, Random Oversampling and SMOTE+TOMEK Links resampling techniques, we can see that SMOTE+TOMEK Links outperforms Random Oversampling and SMOTE. SMOTE+TOMEK Links was able to predict all four classes and correctly classify them. This technique gave the highest percentages of precision when compared to the other two methods. Then SMOTE performed better than Random Oversampling.

SVM does not perform well on its own, to make this algorithm better we had to re-sample the datset using SMOTE, Random Oversampling and SMOTE+TOMEK Liks. SMOTE was the only re-sampling technique that SVM could perform well on. SMOTE was able to predict all four classes without fail and gave the highest percentage of correct predictions in Precision. Random Oversampling and SMOTE+TOMEK Links could only predict and classify class 1, class 2 and class 3, but not class 0. The class that was not predicted fall under the minority class, this means that SVM with Random Oversampling and SMOTE+TOMEK Links could only predict the majority classes.

When comparing the three algorithms used, K-means outperforms the Naive Bayes and SVM. From the three algorithms, K-means was the only algorithm that could detect cyber risk by predicting all four classes with the re-sampling techniques applied and without the techniques. Even though K-means gave poor results, the percentages of precision and recall are very low, it is still a batter algorithm to detect cyber risk. Naive Bayes is a better cyber risk detection when compared to SVM, out of the three re-sampling techniques, only one did not perform well with Naive Bayes. While in SVM, only one technique performed well. All the code used can be found on the this link.

When looking at the different crimes, theft, which is class 3 was found to be the highest. The followed by class 2, which is other incidents which were not specified and then followed by class 1, which is loss and the least cyber crime is class 0 which is hacking.

Algorithm	Re-sampling Technique	Classes	Precision	Recall	$F1$-Score
Naive Bayes	Unbalanced	0	0.00	0.00	0.00
		1	0.00	0.00	0.00
		2	0.44	0.34	0.39
		3	0.55	0.83	0.66
	SMOTE	0	0.28	0.61	0.38
		1	0.13	0.39	0.20
		2	0.55	0.44	0.49
		3	0.80	0.48	0.60
	Random Oversampling	0	0.00	0.00	0.00
		1	0.12	0.39	0.18
		2	0.47	0.48	0.48
		3	0.75	0.53	0.62
	SMOTE and Tomek Links	0	0.26	0.70	0.38
		1	0.14	0.43	0.21
		2	0.53	0.41	0.46
		3	0.81	0.45	0.58
$K-$**means**	Unbalanced	0	0.08	0.52	0.15
		1	0.03	0.04	0.03
		2	0.11	0.03	0.05
		3	0.28	0.19	0.23
	SMOTE	0	0.07	0.30	0.11
		1	0.03	0.04	0.03
		2	0.29	0.06	0.10
		3	0.56	0.57	0.57
	Random Oversampling	0	0.09	0.65	0.15
		1	0.03	0.04	0.03
		2	0.29	0.02	0.03
		3	0.27	0.19	0.22
	SMOTE and Tomek Links	0	0.03	0.17	0.05
		1	0.13	0.50	0.21
		2	0.29	0.06	0.10
		3	0.86	0.20	0.32
SVM	Unbalanced	0	0.00	0.00	0.00
		1	0.00	0.00	0.00
		2	0.54	0.87	0.67
		3	0.82	0.72	0.77
	SMOTE	0	0.33	0.04	0.08
		1	1.00	0.04	0.07
		2	0.55	0.86	0.67
		3	0.82	0.71	0.76
	Random Oversampling	0	0.00	0.00	0.00
		1	0.33	0.04	0.06
		2	0.55	0.88	0.67
		3	0.83	0.70	0.76
	SMOTE and Tomek Links	0	0.00	0.00	0.00
		1	0.33	0.04	0.06
		2	0.55	0.88	0.67
		3	0.83	0.70	0.76

Table 4.13: Summary on the evaluation of the machine learning algorithms used

5. Conclusion

This project was carried out to detect cyber risk using different machine learning algorithms such as the Naive Bayes, K-means and Support Vector Machine (SVM). The dataset used for this project is found on Kaggle and consisted of 1055 records with 10 columns. We found that the dataset was imbalanced. To solve the class imbalance we resampled the dataset using SMOTE and Random Oversampling which are oversampling techniques. Then combined SMOTE and TOMEK Links to see what happens when we oversample and undersample the dataset simultaneously. The efficiency of the algorithms used was evaluated using the Precision, Recall and $F1$-score.

We pointed out the top 7 cyber risk that most organizations are currently facing and suggested to protect themselves from these cyber risks in the future.

Naive Bayes performed well with the SMOTE re-sampling technique. K-means performed well with SMOTE+TOMEK Links and SVM performed well with SMOTE. K-means was found to be the algorithm that detects cyber risk effectively without fail when compared to the Naive Bayes and SVM. Even though it did not give good results, it could predict all four classes and correctly classify them. The highest cyber crime that was the highest on our dataset is theft. The dataset was not really specific as to what type of theft it was. Then followed by Loss and then Hacking.

In conclusion, K-means is a better prediction for cyber risk because it outperformed the Naive Bayes and SVM, with theft being the highest cybercrime faced by different organizations.

5.1 Future Work

Due to the limitation of k-means when the dataset contains outliers, the future work would be to improve this algorithm by sourcing for dataset that have more records so that the removal of outliers will not have an effect on the results. Also, to perform hyper-parameter tuning using advanced methods that will improve the performance of the algorithms.

Acknowledgements

I want to thank God for for giving me the strength, wisdom, knowledge and understanding to do this project. I also want to acknowledge AIMS and it's funders for supporting this work, as well as my supervisor, Prof Phillip Mashele from North West University. My sincere gratitude to my tutors Rahinatou Yuh NJAH and Rock Stephane KOFFI for guiding me throughout this project. Lastly, I would like to thank my parents Mr Tholi David Mhelembe and Mrs Tirhani Evelyn Mhelembe, and my siblings and friends for your support and continuous encouragement throughout the year 2020.

Ndza khensa swinene, Hosi ayi mi katekisi.

References

Analytics Vidhya,2020. Everything you should know about confusion matrix for machine learning. Analytics Vidhya, https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/?utm_campaign=feed&utm_medium=feed-articles&utm_source=feed, Accessed 02 September 2020.

Boardman, J., Biron, K., and Rimbey, R. Mitigating the effects of class imbalance using smote and tomek link undersampling in sas®. Retrieved from SAS: https://www. sas. com/content/dam/SAS/support/en/sas . . . , 2018.

Bouveret, A. *Cyber risk for the financial sector: A framework for quantitative assessment*. International Monetary Fund, 2018.

Ch, R., Gadekallu, T. R., Abidi, M. H., and Al-Ahmari, A. Computational system to classify cyber crime offenses using machine learning. *Sustainability*, 12(10):4087, 2020.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

Cyber Risk Enterprise,2020. Cyber risk and the enterprise. Cyber Risk Enterprise, https://smith.queensu.ca/_templates/documents/it-forum/cyber-risk-management.pdf, Accessed 6 August 2020.

CyberExperst,2020. The quick and dirty history of cybersecurity. CyberExperst, https://cyberexperts.com/history-of-cybersecurity/, Accessed 15 September 2020.

Data Robot,2020. Unsupervised machine learning. DataRobot, www.datarobot.com/wiki/unsupervised-machine-learning/, Accessed 7 August 2020.

Investopedia,2020. Artificial intelligence (ai). Investopedia, www.investopedia.com/terms/a/artificial-intelligence-ai.asp, Accessed 7 August 2020.

Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

Kowalczyk, A. *Support Vecto Machines Succinctly*. 10 2017. ISBN N/A.

Kroon, S. and Omlin, C. Getting to grips with support vector machines: Theory. *South African Statistical Journal*, 38, 01 2004.

Maimon, O. and Rokach, L. *Data Mining and Knowledge Discovery Handbook, 2nd ed*. 01 2010. ISBN 9780387098227.

Medium,2020. Support vector machines — soft margin formulation and kernel trick. Medium, https://towardsdatascience.com/support-vector-machines-soft-margin-formulation-and-kernel-trick-4c9729dc8efe, a.

Medium,2020. K-means clustering algorithm for machine learning. Medium, https://medium.com/capital-one-tech/k-means-clustering-algorithm-for-machine-learning-d1d7dc5de882, Accessed 08 August 2020b.

Medium,2020. Math behind svm (support vector machine). Medium1, https://medium.com/@ankitnitjsr13/math-behind-support-vector-machine-svm-5e7376d0ee4d, Accessed 21 September 2020c.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2018.

O'Kane, P., Sezer, S., and Carlin, D. Evolution of ransomware. *IET Networks*, 7(5):321–327, 2018.

Orman, H. The morris worm: a fifteen-year perspective. *IEEE Security Privacy*, 1(5):35–43, 2003.

Rhea,2020. Statistical pattern recognition and decision making processes(support vector machine). Rhea, https://www.projectrhea.org/rhea/index.php/Lecture_12_-_Support_Vector_Machine_and_Quadratic_Optimization_Problem_OldKiwi, Accessed 21 September 2020.

Rish, I. et al. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

Rockafellar, R. T. Lagrange multipliers and optimality. *SIAM review*, 35(2):183–238, 1993.

Sahoo, J. Modified tf-idf term weighting strategies for text categorization. 10 2018. doi: 10.1109/INDICON.2017.8487593.

Scarfone, K., Jansen, W., and Tracy, M. Guide to general server security. *NIST Special Publication*, 800(s 123), 2008.

Sentinel One,2020. The history of cyber security — everything you ever wanted to know. Sentinel One, https://www.sentinelone.com/blog/history-of-cyber-security/, Accessed 19 September 2020.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tveit, A. and Hetland, M. Multicategory incremental proximal support vector classifiers. volume 2773, pages 386–392, 09 2003. doi: 10.1007/978-3-540-45224-9_54.

Wu, H. and Yuan, N. An improved tf-idf algorithm based on word frequency distribution information and category distribution information. In *Proceedings of the 3rd International Conference on Intelligent Information Processing*, pages 211–215, 2018.