

Louisiana Tech University

Louisiana Tech Digital Commons

Master's Theses

Graduate School

Spring 5-2022

Analysis Of Selection Bias In Online Adversarial Aware Machine Learning Systems

Victor Barboza Morais

Follow this and additional works at: <https://digitalcommons.latech.edu/theses>

**ANALYSIS OF SELECTION BIAS IN ONLINE ADVERSARIAL AWARE
MACHINE LEARNING SYSTEMS**

by

Victor Barboza Morais, B.S.

A Thesis Presented in Partial Fulfillment
of the Requirements of the Degree
Master of Science

COLLEGE OF ENGINEERING AND SCIENCE
LOUISIANA TECH UNIVERSITY

May 2022

LOUISIANA TECH UNIVERSITY

GRADUATE SCHOOL

March 31, 2022

Date of thesis defense

We hereby recommend that the thesis prepared by

Victor Barboza Morais, B.S.

entitled **ANALYSIS OF SELECTION BIAS IN ONLINE ADVERSARIAL**

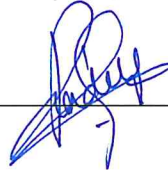
AWARE MACHINE LEARNING SYSTEMS

be accepted in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science



Pradeep Chowriappa
Supervisor of Thesis Research



Pradeep Chowriappa
Head of Computer Science

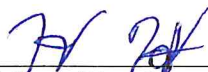
Thesis Committee Members:

Pradeep Chowriappa

Andrey Timofeyev

Manki Min

Approved:



Hisham Hegab
Dean of Engineering & Science

Approved:



Ramu Ramachandran
Dean of the Graduate School

ABSTRACT

As is evident in areas of privacy, security, and ethics, the hindrances to research is the lack of validated real-world data. Therefore, people resort to creating their own dataset and/or artificially increasing the size of existing datasets. However, in areas like countermeasures of phishing, this is not only insufficient but could introduce bias in the dataset in the process.

To raise the awareness of bias in Machine Learning (ML) / Artificial Intelligence (AI) and its consequences, this work tries to gauge one of its occurrences reliably, namely selection bias when generating more samples from existing samples in a dataset. However, there is currently no cross-disciplinary or cross-sector consensus in approaches to identifying or validating measurements, metrics, and key indicators of bias, or how data should be measured or understood in context.

The problem presented in this thesis relies on investigating the effects of selection bias on Adversary-Aware Online Support Vector Machines (AAOSVM) with the help of support vectors to represent selection bias.

APPROVAL FOR SCHOLARLY DISSEMINATION

The author grants to the Prescott Memorial Library of Louisiana Tech University the right to reproduce, by appropriate methods, upon request, any, or all portions of this Thesis. It is understood that “proper request” consists of the agreement, on the part of the requesting party, that said reproduction is for his personal use and that subsequent reproduction will not occur without written approval of the author of this Thesis. Further, any portions of the Thesis used in books, papers, and other works must be appropriately referenced to this Thesis.

Finally, the author of this Thesis reserves the right to publish freely, in the literature, at any time, any or all portions of this Thesis.

Author _____

Date _____

DEDICATION

This thesis is dedicated to my family for their unconditional support, to my advisor for not giving up on me, to my friends for keeping me sane through it all, and to everyone else that has helped me in one way or another.

“The question is the key/answer” - unknown

TABLE OF CONTENTS

ABSTRACT.....	iii
APPROVAL FOR SCHOLARLY DISSEMINATION	iv
DEDICATION	v
LIST OF FIGURES	ix
LIST OF TABLES	xi
ACKNOWLEDGMENTS	xii
CHAPTER 1 INTRODUCTION	1
1.1 Social Engineering.....	2
1.2 Adversarial Setting	4
1.2.1 Adversarial Machine Learning (AML).....	4
1.2.2 Adversarial Sampling Techniques (AST).....	5
1.3 Perspectives of Bias	5
1.3.1 Bias from Machine Learning Model.....	5
1.3.2 Bias Brought by Disparity in the Dataset	6
1.3.3 Bias Introduced During the Creation of the Dataset.....	7
1.3.4 Bias Introduced During the Preprocessing of the Dataset	8
1.4 Problem Statement.....	8
1.5 Key Contributions.....	9
CHAPTER 2 BACKGROUND	11
2.1 Challenges of Phishing Datasets.....	11
2.2 Creation of the Dataset	12

2.3	Extracting Information from the Dataset	12
2.3.1	Feature Engineering	13
2.3.2	Enhancing Machine Learning Models	13
2.4	Bias in the Dataset	14
2.5	Benchmarking Phishing Datasets	15
CHAPTER 3 METHODS		16
3.1	Creation of the AAOSVM	16
3.2	Experimental Settings	18
3.2.1	Proposed Framework	18
3.2.2	Bias Metrics	20
3.2.3	Performance Metrics	21
3.2.4	Instance Selection	22
3.3	Used Datasets.....	24
3.4	Attacker Model	26
3.5	Adversarial Sampling Technique (AST)	27
3.5.1	Conceptual Overview.....	28
3.5.2	Feature Selection.....	29
3.5.3	Sample Seed Selection.....	30
3.6	Adversarial Classification.....	32
3.6.1	Optimization of the AAOSVM.....	32
3.6.2	Bias in the Algorithm.....	35
3.6.3	Scoring Each Sample	36
CHAPTER 4 RESULTS AND DISCUSSION.....		38
4.1	Experiment 1 - Evaluation of Machine Learning Models	38
4.2	Experiment 2 - Performance of the Online SVM on a Moving Window	47

4.3	Experiment 3 - SVM Becoming Adversary-Aware.....	50
4.4	Experiment 4 - Changing the Scores	51
4.4.1	Significance of Experiment 3 and 4 – Bias Metrics	54
4.4.2	Significance of Experiment 3 and 4 - Testing for False Positives.....	57
CHAPTER 5 CONCLUSIONS AND FUTURE WORK.....		60
5.1	Conclusions.....	60
5.2	Future Work.....	60
5.2.1	Poisoning Attack.....	60
5.2.2	Proper Metrics.....	61
5.2.3	Other Machine Learning Models	61
5.2.4	Tuning Machine Learning Models Parameters.....	61
5.2.5	Time to Train	61
5.2.6	Improving the AAOSVM Scoring.....	61
BIBLIOGRAPHY.....		62

LIST OF FIGURES

Figure 1-1: Examples of bias in the dataset without discrimination.	6
Figure 1-2: Examples of bias in the dataset with discrimination. The blue histograms represent the phishing instances, and the orange histograms represent the legitimate instances.	7
Figure 3-1: Overview of the framework adopted in this thesis.	18
Figure 3-2: Desired instance selection.	23
Figure 3-3: Undesired instance selection.	23
Figure 3-4: Algorithm 1 from [12], illustrated. “x” is an instance with 4 features and a label, and the selected features are the second and the third one. “X” is any value and “Y” is the label of that instance. In this example, all the unique values in columns 1 and 2 with the same label as “x” of the dataset are 0 and 1.	28
Figure 3-5: Generation of samples. Number of samples generated for each seed sample, according to number of features manipulated.	30
Figure 3-6: Dataset Split.	31
Figure 3-7: Overview of the inner workings of the AAOSVM of Dynamic Scoring.	35
Figure 3-8: Adversary-Aware Online SVM of Dynamic Scoring.	37
Figure 4-1: Performance metrics on machine learning models on dataset DS1.	39
Figure 4-2: Performance metrics on machine learning models on dataset DS2.	39
Figure 4-3: Performance metrics on machine learning models on dataset DS3.	40
Figure 4-4: Performance metrics on machine learning models on dataset DS4.	40
Figure 4-5: Performance metrics on machine learning models on dataset DS5.	41
Figure 4-6: Performance metrics on machine learning models with normalized data of dataset DS1.	42

Figure 4-7: Performance metrics on machine learning models with normalized data of dataset DS2.	42
Figure 4-8: Performance metrics on machine learning models with normalized data of dataset DS3.	43
Figure 4-9: Performance metrics on machine learning models with normalized data of dataset DS4.	43
Figure 4-10: Performance metrics on machine learning models with normalized data of dataset DS5.	44
Figure 4-11: SVMs as compared on ACC.	44
Figure 4-12: SVMs as compared on ACC with normalized data.	45
Figure 4-13: SVMs as compared on TPR.	45
Figure 4-14: SVMs as compared on TPR with normalized data.	46
Figure 4-15: SVMs as compared on F1.	46
Figure 4-16: SVMs as compared on F1 with normalized data.	47
Figure 4-17: Performance metrics of OSVM.	48
Figure 4-18: Performance metrics of OSVM with normalized data.	48
Figure 4-19: Time to execute each run of the cross validation of the OSVM.	49
Figure 4-20: Time to execute each run of cross validation of the OSVM with normalized data, according to each dataset.	50
Figure 4-21: Performance metrics of AAOSVM, running 10x each dataset without 200 random samples.	51
Figure 4-22: Score values for DS4, on 200 samples.	53
Figure 4-23: Score values for DS5, on 200 samples.	53
Figure 4-24: Average DPPTL on one manipulated feature.	55
Figure 4-25: Average DPPTL on two manipulated features.	55
Figure 4-26: Average percentage of FP using different seeds on the AST for different number of manipulated features.	57
Figure 4-27: Comparative of Class Imbalance and Percentage of FP on each dataset for each type of sample seed.	58

LIST OF TABLES

Table 3-1: Parameters used in the classifiers.	19
Table 3-2: Summary of the objective features in each dataset.	25
Table 4-1: Performance metrics of AAOSVM changing scores, running 1x each dataset without 200 random samples on a holdout validation.	52
Table 4-2: Distance from the original sample. All the selected samples' type have the same values for each respective dataset.	56

ACKNOWLEDGMENTS

Firstly, I want to praise the Lord that through this work and through the journey that came with bringing this thesis to life, He has made Himself more present in my life. My eternal gratitude to Him for teaching me what devotion really means, both to Him and to my work.

Foremost, I would like to express my sincere gratitude to my advisor Dr. Pradeep Chowriappa for the continuous support of my master's study and research, for his patience, motivation, enthusiasm, immense knowledge, and keen eyes. Without him, this thesis would not have happened.

A special thanks to Matheus Leone e Danilo Carvalho for helping me review part of my code.

CHAPTER 1

INTRODUCTION

To take full advantage of the Internet, one must know how to use it. The freedom we enjoy comes in part from protection against harmful actions [1]. Socially Engineered Attacks (SEA) are one of the major forms of attacks that plague organizations as we accept the digitalization of various aspects of work [2], [3]. Not every cyber-threat has the same goal or impact [4]. It is increasingly difficult to keep up with the malicious actors who want to steal user's information and money. There is a paradigm shift in cybersecurity, whereby users are the first line of defense in online security of anything that is digital, e.g., networks, systems, users' passwords and users' identities. This paradigm shift in cybersecurity requires the involvement of as many areas of research [5].

SEAs differ in scale and scope making it difficult for cybersecurity experts to detect and prevent it [6]–[8]. The economic impact of an average data breach has risen from \$4.9 million in 2017 to \$7.5 million in 2018 [4]. A successful phishing attack can lead to much more losses as the leak of sensitive information can have long term future ramifications as well [9].

The work presented in this thesis is focused on the website phishing - one aspect of the SEAs. Phishing is a popular type of cyber-attack that uses messages (usually via emails) as its medium of attacks [6]–[8], [10], [11]. Phishing attempts to acquire sensitive information using malicious URLs are through electronic communication mediums or by

phone [6]. For example, consider the situation when one arrives at a potential fraudulent website either by clicking on a link or by searching for it; it is wise to have accurate means of detecting if that website is legitimate or not.

There is a gap between threat and defense, as ill-intentioned people deploy increasingly sophisticated attack technology and engage in cyber-crime as the world becomes more connected [5]. Phishing as a SEA cannot be solved just by informing the end users [12].

Considering the security risks involved and the need for appropriate cybersecurity responses – that can scale to every changing landscape – the motivation of this research is to explore the role of adversarial machine learning techniques employed in defending against the spread of phishing attacks. In this chapter, we introduce foundational concepts that we leveraged in the formulation of our problem statement, objectives, and specific aim of this work.

1.1 Social Engineering

Social Engineering (SE) in cybersecurity is the manipulation of trust of people to steal private information [13]. One popular mode of doing this is by phishing. Phishing can be categorized into two types based on how users are manipulated to carry out an attack. In the first type of a phishing attack, the user **actively** participates - where the user unintentionally inputs information or carrying out an action, such as downloading a virus. On the contrary, the second type of phishing attack is where the user **passively** participates: the user is monitored on malicious websites or by unsuspectingly downloading a virus when opening an email.

Socially engineered attacks (SEA) – phishing included – start by either actively or passively gathering information about the user. Once sufficient information is gathered the next set of phishing attacks are carried out days later [15], [16]. Here, attackers use the gathered information in any way they want to in order to exploit the user. By this point, it is extremely difficult for anyone, even a cybersecurity expert, to detect or mitigate the attack.

As of now, we have three methods of mitigating phishing attacks: (a) Manual mitigation, which should be avoided for many reasons—two of the main reasons being that it is inefficient and that it is error-prone, (b) Blacklisting malicious websites, which is good but must be constantly updated and does not account for new websites, and (c) The use of machine learning based heuristics which is considered the best and most reliable methods.

Recent advances in cyber security have shown that machine learning (ML) has played an important role in building monitoring systems to defend against socially engineered attacks. However, attackers target monitoring systems with the intent of deceiving models built to protect users. Adversarial machine learning (AML) is an area of research aimed at understanding the role of machine learning algorithms against carefully targeted attacks, with the intent of creating countermeasures to enable more secure learning algorithms.

In the context of adversarial learning - there are basically two types of adversarial attacks prominently studied in related works namely, poisoning and evasion.

Poisoning attacks try to mislead the machine learning algorithm during the **training** phase by manipulating only a small fraction of the training data to increase the

number of misclassified samples at **test** time significantly [14]. Evasion attacks consist of manipulating input data at **test** time to cause misclassifications. In this thesis, we will focus on evasion attacks [14].

1.2 Adversarial Setting

The challenge of detecting adversarial attacks using phishing is that the strategies used by the phishers adapt with time and the targeted user.

1.2.1 Adversarial Machine Learning (AML)

Most ML algorithms are designed to work on specific problems with datasets. From these datasets, both training and test data are derived to train and test a ML model. It is standard practice that both training and test data originate from the same statistical distribution. When those models are applied to the real world, adversaries aim to supply data that violates any statistical assumption made by a classifier. Attackers arrange the data to exploit vulnerabilities and compromise the performance of a classifier.

On the contrary, AML is a machine learning technique that attempts to fool models by supplying deceptive inputs. The intent is to cause a malfunction in a machine learning model. This malfunction will depend on the type of threat that it is facing. For example, the malfunction could be to misclassify a set of instances or not to “understand” what the true difference between a phishing and a legitimate website is.

Adversary-aware machine learning (AAML) models takes into consideration an attacker (simulated or not). The attacker’s objective is to make the classifier performance decrease, either in general or for some specific samples. This is one of the countermeasures taken against the attacker.

In the case of the AAOSVM, it is done by calculating probabilities of each label, having, and changing the scores for each sample, and only training on a sample if it reaches a certain threshold [16].

1.2.2 Adversarial Sampling Techniques (AST)

As is evident in areas of privacy, security, and ethics [5], the hindrances to the research is the lack of validated real-world data. We would benefit with data that was collected from a real-world phishing attack and solely verified that it is what indeed it was supposed to be. Due to the sparsity of the data, we employ data from ASTs that simulate a phishing attack.

AST as a technique randomly chooses a data sample from the input dataset and modifies the data sample with the intent of causing a machine learning to misclassify the said sample.

1.3 Perspectives of Bias

Bias is the simplifying assumptions made by a machine learning model that skews the overall model performance. Here, we emphasize the various types of bias in ML specifically with the SVM algorithm.

1.3.1 Bias from Machine Learning Model

Parameter bias in the context of the linear SVM and it makes use of **Eq. 1-1**, where the b is called bias or threshold, depending on the source [17], [18]. It only means that it is dislocating the hyperplane from the origin.

$$f(x) = w^T \cdot x + b \quad \text{Eq. 1-1}$$

This terminology derives from the point of view that the output of the transformation is biased toward being b in the absence of any input i.e., $f(x)$ will tend to b if $b \gg (w^T \cdot x)$, either by b being too big or $w^T \cdot x$ being too small.

1.3.2 Bias Brought by Disparity in the Dataset

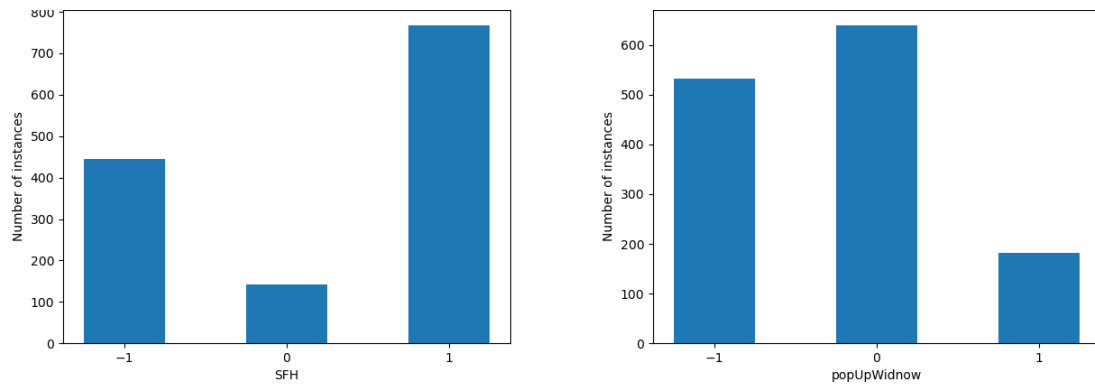


Figure 1-1: Examples of bias in the dataset without discrimination.

Each feature tends towards a certain value. **Figure 1-1** shows two features, one with a preference for the value of 1, and another avoiding the value of 1.

This is more evident when the classes are separated. **Figure 1-2** shows the difference in bias (i.e., preference) in the data for two features for each class.

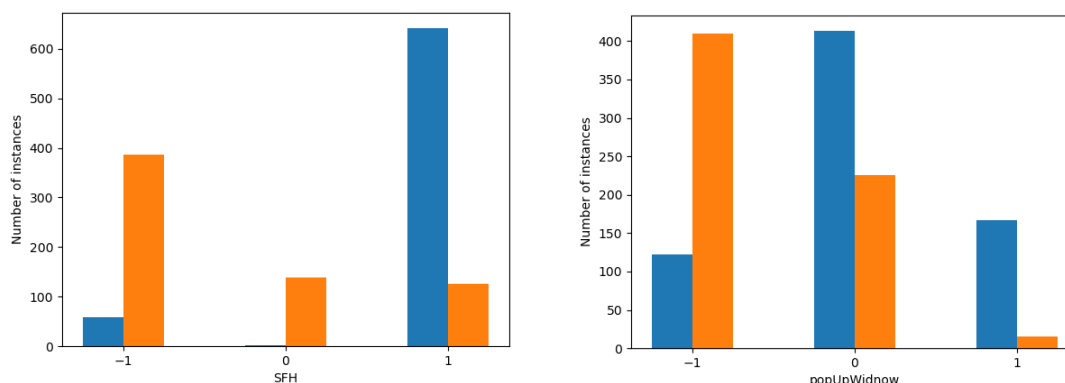


Figure 1-2: Examples of bias in the dataset with discrimination. The blue histograms represent the phishing instances, and the orange histograms represent the legitimate instances.

Whenever an instance has the value of -1 for the “SFH” or “popUpWindow” feature, a classifier trained on this dataset could be likely to predict this instance as a phishing instance. Similarly, if an instance has the value of 1 for the “SFH” or the value of 0 “popUpWindow” feature, a classifier trained on this dataset could likely predict this instance as a phishing instance.

One way to overcome this problem would be to generate more samples with the other, less preferred values, until a balance is achieved, should this be a problem for the classifier.

1.3.3 Bias Introduced During the Creation of the Dataset

The human perception influences what should and what should not be in the dataset. Another way to introduce bias to the dataset is not to know what is important and what is not important to insert into it.

In our case, this could be what websites were captured to be in the dataset or which features are chosen to represent a website. This is known as **selection bias**.

According to Wikipedia:

“Selection bias is the bias introduced by the selection of individuals, groups, or data for analysis in such a way that proper randomization is not achieved, thereby ensuring that the sample obtained is not representative of the population intended to be analyzed.”

1.3.4 Bias Introduced During the Preprocessing of the Dataset

By changing the dataset by removing samples or generating more samples from selected samples of the dataset, we have another “side” of **selection bias**, where the dataset could already represent a good generalization of the problem, but by introducing or removing samples, the dataset becomes skewed towards one or more classes.

1.4 Problem Statement

Countermeasures for phishing in social engineering deal with a constantly changing data stream that causes the classifier to malfunction.

This is especially problematic in the academic setting because there are not many datasets publicly available. Even when you create your own dataset, it is impossible to capture the dynamic changes in the data. To compensate for this, we decided to use an Adversarial Sampling Technique (AST) [12] to simulate an attacker. In the attack simulation, one must be careful **not** to introduce bias. An attacker with the same goal, knowledge, and influence as the simulation should be able to make a successful attack.

We also need to try to compensate the misclassification in the classifier. For this, we implemented a modified version of the Adversary-Aware Online SVM (AAOSVM)

[19]. It can change the sample scores during training and takes into account the adversarial nature of the data.

Objectives:

- *To create a baseline:* Compare batch classifiers from sklearn with incremental SVM from sklearn and with AAOSVM [19] performances.
- *Data standardization and pre-processing:* Investigate the effects of normalization on sklearn classifiers on its performances.
- *Data mining:* Investigate the effects of generated labels by clustering on generated data bias metrics.
- *Validation:* Investigate the effects of making the AAOSVM more dynamic on bias metrics and performance.
- *Assessment:* Investigate the effects of selection bias by using support vectors to generate more data.

Research Question 1: If support vectors are used as AST seed samples, will it cause more misclassification?

Research Question 2: Is there a class imbalance in the generated data by AST and does it negatively affect the classification algorithm?

Research Question 3: How does the AAOSVM behave in the worst-case scenario, with changing scores?

1.5 Key Contributions

The specific aim of this research is to raise awareness about the consequences of bias in machine learning, especially in areas where the data that is publicly available is hard to find and does not consider the context in which it is meant for such as phishing.

With the help of four datasets and with four experiments, we present an investigation on the effect of bias at the data preprocessing level. An AST will be used to mimic an attacker and an AAOSVM as a model of the end user.

The algorithm from [12] was reproduced according to what is presented in the paper. This AST used to model the attacker will go against the AAOSVM to see how that would change its behavior. The objective of this work is to shed light on the role of selection bias brought by the AST.

The remainder of this thesis is organized as follows. Chapter 2 (Background) consists of an overview of previous works. Chapter 3 (Methodology) outlines the experiments in terms of used datasets, assumptions, settings, algorithms, performance, and bias metrics. Chapter 4 (Results and Discussion) encapsulates the results of the experiments and discussion. Chapter 5 (Conclusion and Future Works) contains suggestions for further research, refinements, and improvements.

CHAPTER 2

BACKGROUND

2.1 Challenges of Phishing Datasets

In the beginning, people did all the work of selecting which websites were legitimate, and these people were experts or the users themselves. As the attacker is always trying to exploit certain traits in human behavior [11], [20], the experts would look for flaws such as misspelling and suspicious links, to blacklist those websites manually.

Some considerations must be made regarding when a phish is considered successful or not and the amount of untrained people that fall for them. Phishing could be considered successful by clicking on a link or by giving information to a website, for example [20]. Untrained people have a high rate of falling in a phishing attempt (about 52%) [20], but even trained people might misclassify websites (sometimes misclassifying real websites as well, out of suspicion) [21]. Maybe that has to do with training, as a report from a company of awareness training shows that the percentage of people that falls for phishing decreases significantly after only 90 days of training and gets as low as 5% after a year of training [22].

While it is important to train as much people as possible, with the growth of the Internet and the number and complexity of the attacks, it became virtually impossible to defend against them just by manually selecting each website.

2.2 Creation of the Dataset

Services like PhishTank^① started being used to blacklist and whitelist, respectively, websites so users and other services, such as google.com, could use them to know which websites to trust and which to avoid. This approach is insufficient as there is zero-day attacks and similar ones that can get through it [7], but necessary, as many datasets are built from it to be used [23].

2.3 Extracting Information from the Dataset

Many companies use ML tools to extract information from a vast amount of data. These tools are especially useful when dealing with problems such as phishing [24]. Supervised learning is a method that involves the system learning to map inputs and outputs based on existing input and output pairs. Supervised learning consists of techniques including SVM, linear regression, decision trees (DT), and neural networks (NN) [25]. There are several issues to consider when using supervised learning [26]. These issues are bias-variance tradeoff, functional complexity, dimensionality of the input space, and noise in the output values. Bias-variance tradeoff refers to the tradeoff between the bias and variance of a learning algorithm [27]. Bias refers to the error from erroneous assumptions in the learning algorithm. High bias can result in under-fitting. Variance refers to the error from sensitivity to small fluctuations in the training set. High variance can result in overfitting [28]. The prediction error of a learned classifier is related directly to the sum of the bias and variance of a learning algorithm.

¹ <https://phishtank.com/>

2.3.1 Feature Engineering

When it comes to phishing, many works focus on either defining features that will improve the performance of the machine learning models or enhancing the existing machine learning models [12], [29].

Sometimes it is a mixture of those two. For example, Niakanlahiji *et al.* introduced PhishMon [30], a machine learning framework to detect phishing websites that depends on certain features. Although it is a recent development, is scalable, and independent of third parties, it fails to consider the very nature of phishing attacks.

The problem with works that focus mainly on the features is that attackers can decipher which features are being used to decide what is a phishing attempt and what is legitimate and mimic it.

A good example would be the work of Shirazi *et al.* who tried to determine the most important features that would distinguish a phishing website from a legitimate one [9]. They tried to avoid anything that was convoluted, like DNS routing, or that could be compromised later – at least from an academic point of view, like third party services.

Another possibility would be to focus on something such as URL and the reasoning that phishing websites' URLs can be identified by a trained person with a certain ease [31], then URL shortening comes along and your work becomes not so valid anymore.

2.3.2 Enhancing Machine Learning Models

Jiang *et al.* developed a Convolutional Neural Network (CNN) that automatically extract features from the URL [32]. It combines deep neural network with natural language processing and threat intelligence to do so. That could potentially be robust

against an adversarial attack, especially because it uses incremental updates, but it did not explore that possibility [32].

Pereira *et al.* distinguished legitimate from phishing domains with precision and accuracy with the use of graphs [33]. Domain Generation Algorithms (DGA) was used to simulate an attacker. Although good classification was obtained in an adversarial environment, only the domain part of the website was used. From making graphs of domains from DGAs, one can see that there are some trends [33].

A few more considerations noted by Shirazi *et al.* [12] attackers have full control over the URL, except the Second Domain Level (SDL); therefore, any solution that does not account for or does not have room for considering the website content would be disregarded in the real world. Considering that your information still has not been stolen by just entering the website, the content of the website will determine if it will or not be stolen.

Figuroa *et al.* devised an AAOSVM that uses game theory to help in the classification process [19]. There are a few problems with it. First, they create their own dataset using natural language processing, not accounting for possible introduced bias. Second, it uses a function that is nowhere to be found and it is based on a library that has poor documentation, making reproduction near impossible.

2.4 Bias in the Dataset

In one way or another, Chiew *et al.* noted some ways a dataset could have bias introduced during and from its assembly step [23], namely (1) how big it is (interferes with the standard error), (2) where are the samples coming from, (3) how popular are the websites being used in the dataset, (4) without due description, the dataset could be using

a high volume of the same brand, (5) improper distribution of website categories, and (6) not enough languages (English, Spanish, etc.).

The focus of Shirazi *et al.* [9] is to try to create an unbiased dataset. Shirazi *et al.* points to two main causes of bias in the dataset: (1) too many features with not enough arguments to sustain those choices of features, and (2) many datasets are based on URL or content – [12] considered the potential bias in at least one of the used datasets; the remark made was about the use of URL length. As a sidenote, (1) Shirazi *et al.* also said that content-based dataset is not efficient due to the number of training features, (2) that third-party servers violate user privacy, and (3) while URL-based datasets have a good indicator of phishing attacks, their features make them unlikely to be used in the future as the attackers have complete control over the URL (except the domain part, at least for now).

2.5 Benchmarking Phishing Datasets

It is not easy to find publicly available datasets for phishing and many researchers use their own dataset for their experiments. This makes benchmarking for anti-phishing techniques very difficult [23]. The two main reasons for not finding publicly available datasets in this area are because (1) it is dealing with sensitive information and cannot be or stay publicly available for long, and (2) it was hosted somewhere but it was moved or deleted without a trace (ironically, it seems to be the case for the dataset from [23]).

CHAPTER 3

METHODS

3.1 Creation of the AAOSVM

In this thesis, we base our implementation of the SVM algorithm with Sequential Minimal Optimization (SMO) inspired by Charest implementation [34][35]. We describe the SVM using the following definitions.

Definition 1. Support vectors ($\vec{\alpha}$): instances that are closer to the hyperplane and influence the position and orientation of the hyperplane.

Definition 2. Objective function $W(\vec{\alpha})$: the hyperplane function that defines the class label of an instance, calculated as the RHS on **Eq. 3-1**.

$$\max_{\vec{\alpha}} W(\vec{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N y_i y_j \vec{x}_i \vec{x}_j \alpha_i \alpha_j \quad \text{Eq. 3-1}$$

where N is the number of instances in the dataset and the second term of the RHS is multiplying with itself. Maximizing $W(\vec{\alpha})$ while respecting the constraint **Eq. 3-2** constitutes the basis of the SVM.

$$\sum_{i=1}^N y_i \alpha_i = 0 \quad \text{Eq. 3-2}$$

where $0 \leq \alpha_i \leq C, i = 1, \dots, m$, and m is the number of instances.

The support vectors (α) are rounded to zero or a constant (C) if they are close to one of those values.

The next steps are to condense all the functions into the SVM class for later modifications, implement windowing, implement sample “check” (only poorly classified samples will make the model retrain), and join all of these before making changes to the SVM itself. This is brought about by SVM windowing and the definition of training criteria described below.

Definition 3. Bias or threshold b : the dislocation of the hyperplane from the origin.

Definition 4. The weight vector (\vec{w}): the normal vector of the hyperplane and the minimal distance to the hyperplane, defined as **Eq. 3-3**,

$$\vec{w} = \sum_{i=1}^N y_i \alpha_i \vec{x}_i \quad \text{Eq. 3-3}$$

where N is the number of instances in the window.

During the training, the AAOSVM always looks at a pair of samples at a time, so the update on the weight vector is done using **Eq. 3-4**. This saves computational power by only computing the change of the two samples observed and then added to the old vector.

$$\vec{w} = \vec{w}^{old} + y_1(a_1^{new} - \alpha_1^{old})\vec{x}_1 + y_2(a_2^{new} - \alpha_2^{old})\vec{x}_2 \quad \text{Eq. 3-4}$$

When optimizing, the SVM uses a vector of prediction errors to determine which sample will make the best pair to train on. The **SVM windowing** is implemented with the size of 100 samples, i.e., the maximum number of samples that the model knows is 100. The window slides one sample at a time, i.e., with each new sample, and if the window is full, the last one is discarded. Although the model is trained on a sliding window, it keeps its “knowledge” (values of α and b).

Definition 5. The training criteria: trains the classifier whenever samples that are considered poorly classified, as expressed in **Eq. 3-5**.

$$y_i \times (\vec{w}_{i-1} \cdot \vec{x}_i + b_{i-1}) < \varepsilon \quad \text{Eq. 3-5}$$

where \vec{w}_{i-1} and b_{i-1} are the weight and bias term from the previous training iterations.

The variable ε has the value of 0.6 which adopted this value on their algorithm [36]. This threshold sets the scaling of \vec{w} and could have been any positive number [18].

Finally, before making major changes on the SVM itself, we merged the sliding window with the sample selection.

3.2 Experimental Settings

All experiments presented in this thesis adopts the framework shown in **Figure 3-1**. Each experiment was repeated 10 times and the results reported are an average across 10 iterations.

3.2.1 Proposed Framework

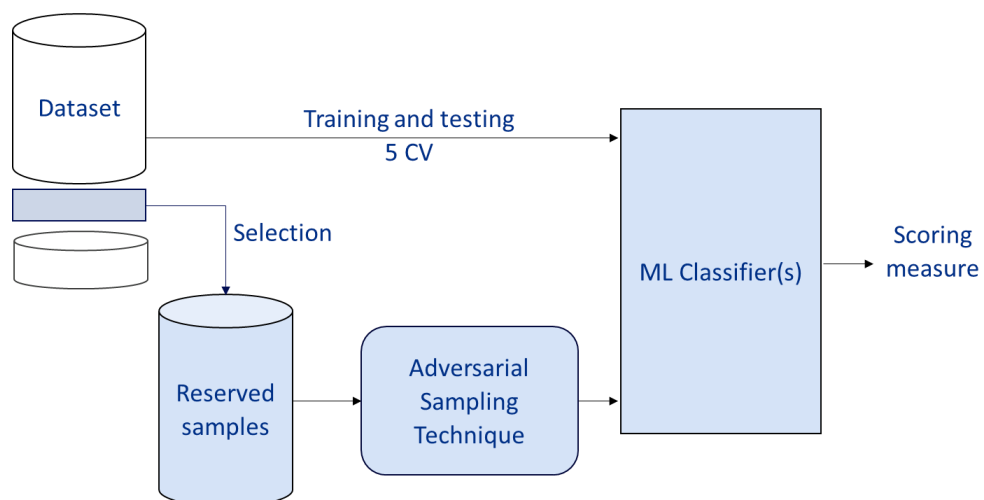


Figure 3-1: Overview of the framework adopted in this thesis.

All the class labels are verified and changed, if necessary, to +1 as a phishing instance and -1 as a legitimate instance. As part of our validation, all the dataset but 200 instances is used for 5-fold cross validation.

All the classifiers that are used to compare against the AAOSVM have parameters set as listed in **Table 3-1**.

Table 3-1: Parameters used in the classifiers.

Classifier	Parameters
Nearest Neighbors (KNN)	K = 3
Linear SVM (SVC, kernel="linear")	C =100, max_iter =10000, random_state = 42
RBF SVM (SVC, kernel="rbf")	Gamma = 2, C = 100, max_iter =10000, random_state=42
Decision Tree	max_depth = 5, random_state = 42
Random Forest	max_depth = 5, n_estimators =10, max_features = 1, random_state = 42
Naive Bayes	-
Gradient Boost	random_state = 42
OSVM (SGDOneClassSVM)	random_state = 42
AAOSVM	C = 100, m =100, Gp = 250, Em =10, Er = 10, Ym = 10, Yr = 10, s = 0.6, kernel_type = "linear", k = 3, max_optimization = 2

Some classifiers need a seed to initialize. This is given by the “random_state” attribute. If no seed is given, every iteration (run) of every dataset could lead to a different performance that has nothing to do with the input data. The reason for them to have a fixed initial random state is that we do not want to introduce a potential bias to the

experiments. Whenever an experiment does not follow these settings, it will be mentioned.

3.2.2 Bias Metrics

Class Imbalance (CI) is used to measure the imbalance of class labels after the AST as shown in **Eq. 3-6**.

$$CI = \frac{n_p - n_l}{n_p + n_l} \quad \text{Eq. 3-6}$$

where n_p is the number of phishing instances and n_l is the number of legitimate instances. The value of CI ranges from -1 to 1, where -1 signifies that there is only legitimate instances, 0 signifies that there is a perfect balance of labels, and 1 signifies that there is only phishing instances.

Difference in Positive Proportions of True Labels (DPPTL) is also used to track the imbalance of class labels after the AST as shown in **Eq. 3-7**.

$$DPPTL = \frac{n_a^p}{n_a} - \frac{n_d^p}{n_d} \quad \text{Eq. 3-7}$$

where n_a is the number of all the samples that have a certain value on a specific attribute, n_a^p is the same as n_a but are labeled as phishing, n_d is the number of all the samples that **do not** have that certain value on a specific attribute, and n_d^p is the same as n_d but are labeled as phishing. It needs to be calculated for each manipulated feature.

A positive DPPL value indicates that there is a preference towards that certain value in that specific attribute. This is referred to as *positive bias*. Similarly, a negative DPPL value indicates that there is a preference towards other values other than the certain value in that specific attribute. This is referred to as *negative bias*.

DPPTL can also be used to check Demographic Parity (DP) as shown in **Eq. 3-8** [37], which checks if the classifier uses an attribute to predict the labels or not:

$$DP = P(\hat{Y} = \hat{y}) = P(\hat{Y} = \hat{y}|A = a) \quad \text{Eq. 3-8}$$

where \hat{Y} are all the predicted labels and A are all the values of a specific attribute.

In this thesis, we employ the Euclidean distance (d) as shown in **Eq. 3-9**:

$$d(x_i, x'_i) = \sqrt{\sum_{j=1}^a (x_i^j - x'_i{}^j)^2} \quad \text{Eq. 3-9}$$

It is calculated for each original sample that has at least one sample from the generated samples to be classified as a legitimate sample. Only the smallest distance from the original sample to the generated samples where that original sample generated is kept.

3.2.3 Performance Metrics

The following performance metrics are employed to determine the classifier performance. The classifiers were measured and compared performances in Accuracy (ACC). Accuracy is measured to see the correctness of the models. Accuracy is represented in **Eq. 3-10**:

$$ACC = \frac{TP + TN}{TP + FN + TN + FP} \quad \text{Eq. 3-10}$$

True Positive Rate (TPR) or Recall (R) is also known as sensitivity. It is trivial to achieve recall of 100% by making the model classify everything as phishing. A high sensitivity model is reliable when its result is negative since it rarely misclassifies the TP. A negative model outcome can be taken as TN. However, a positive outcome in a model with high sensitivity is not necessarily useful. TPR is captured in **Eq. 3-11**:

$$TPR \text{ or } Recall = \frac{TP}{TP + FN} \quad \text{Eq. 3-11}$$

The formula of Precision (PRE) is included to calculate the F1-score (F1) and is defined in **Eq. 3-12**:

$$PRE = \frac{TP}{TP + FP} \quad \text{Eq. 3-12}$$

F1-score (F1) is the harmonic mean of precision and recall. It is another way of looking at a model's accuracy. It is also commonly used to compare the model's performances. An important thing to note is that this measure is dependent on the class imbalance [40]. F1-score is represented in **Eq. 3-13**:

$$F1 = \frac{2 \times PRE \times REC}{PRE + REC} = \frac{2 \times TP}{2 \times TP + FP + FN} \quad \text{Eq. 3-13}$$

3.2.4 Instance Selection

We believe that the choice of which instances will be kept away from the training can bias how the model behaves. Take **Figure 3-2** as an example. If we had to select four instances to keep away from training, the ones that have been circled are good candidates. They are good candidates because two of them represent the class and two of them represent the ones that most likely will cause some trouble for the classifiers, as represented by the hyperplane in red.

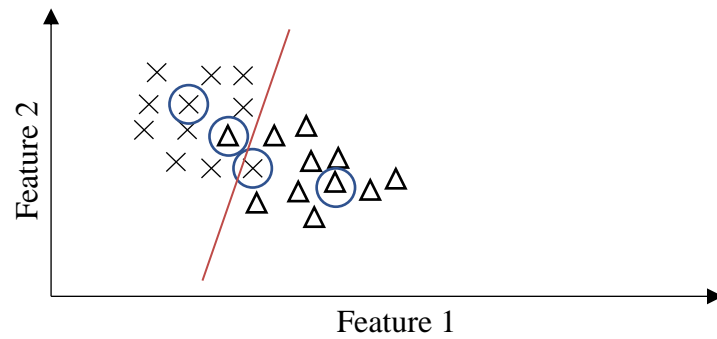


Figure 3-2: Desired instance selection.

Now imagine we did the selection shown in **Figure 3-3** instead. We most likely would never generate a new sample that would cause any trouble.

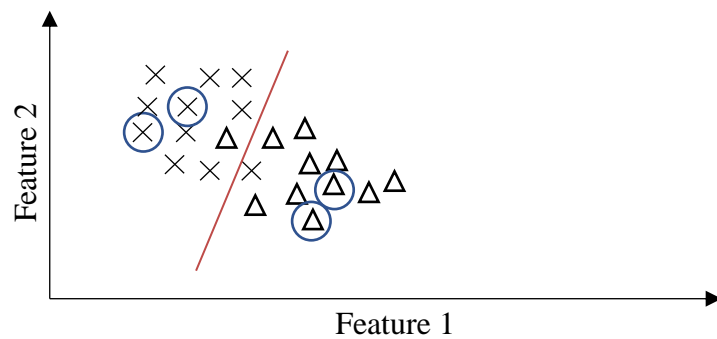


Figure 3-3: Undesired instance selection.

Our investigation on the effect of bias at the data preprocessing level works on the premise that **the AST will be used to mimic an attacker and the AAOSVM as a model of the end user**. We devised four types of experiments to test and show how an attack from an AST would influence each type of classifiers as follows:

1. **Experiment 1:** Evaluation of machine learning classifiers with both the datasets as they are and the datasets with normalized data with the use of minmax normalization.

2. **Experiment 2:** Evaluation of an online SVM classifier with both the datasets as they are and the datasets with normalized data with the use of minmax normalization.
3. **Experiment 3:** Evaluation of the AAOSVM on 200 randomly selected samples and investigate the effects of generated labels by clustering on generated data bias metrics.
4. **Experiment 4:** Evaluation of the AAOSVM on 200 samples selected based on support vectors from previous experiments and investigate the effects of:
 - a. generated labels by clustering on generated data bias metrics.
 - b. changing the scores of each sample with each training iteration.

3.3 Used Datasets

Four datasets that are publicly available on the Internet (Mendeley data and UCI repositories) are used, to which 3 are the same as [12]. The reasons these datasets were chosen are (i) to try to compare results with [12], (ii) to see how the algorithms will perform on a bigger dataset, (iii) to avoid unnecessary complexity, and (iv) to avoid introducing bias inadvertently when extracting features from emails, for example. **Table 3-2** summarizes important aspects of each dataset and shows their relevance by the number of papers that use it.

Since most of the phishing attacks involve going to a fake website at some point, we are going to try to help better the automation of detecting them before they can do any harm. Another thing to be pointed out is it works with these datasets; it should work with any other that has similar feature value types.

Table 3-2: Summary of the objective features in each dataset.

Dataset	Data shape (#)		Instances		Features			# of papers that use it
	Size	Features	Legitimate	Phishing	URL based	# binary	# trinary	
DS1a (DS1)	58,645	111	27,998	30,647	96	9	0	1
DS2	11055	30	6157	4898	8	20	10	184
DS3	1353	9	651	702	5	2	7	270
DS4	10000	48	5000	5000	27	23	6	11
DS1b (DS5)	88,647	111	58,000	30,647	96	9	0	1

Binary features are the ones that have only two values in the dataset. Similarly, trinary features are the ones that have only three values in the dataset.

Dataset 1 (DS1a and DS1b) [39] is the only one that differs from the ones used in [12], since the dataset created by Shirazi not available for public use. This dataset is the biggest and the most recent one, not only instance wise but feature wise as well. It has two versions – one to favor machine learning models and another to mimic real-world scenarios. Most of its features are about the number of certain characters in the URL and the total number of characters of parts of the URL.

Based on [23], this is the only one of the 4 used datasets that is significantly large enough to be used in anti-phishing research. From here onwards, DS1a will be called DS1 and DS1b will be called DS5.

Dataset 2 (DS2) [40], [41] has all the features of Dataset 3 plus twenty more, but it has about nine times more instances. We might see some correlation between the two.

All the features in Dataset 3 (DS3) [42], [43] are in the format of 1, 0, and -1. It also has three class labels, namely: “Legitimate”, “Suspicious”, and “Phishy”, websites, respectively. For the sake of reproducibility, all the “suspicious” labels are considered as “phishy”.

DS3 is the smallest dataset both in instances and features. Five of its features look at the URL for clues for whether a website is legitimate. These include looking at the presence of a symbol and the length of the URL. The other five features take a deeper look at the website. Those include checking if the URL will lead to another domain, how big is the traffic on the website, and if there is a DNS record.

In terms of features, this Dataset 4 (DS4) [44]–[46] is almost as balanced as DS3 in terms of how many features are dedicated to the URL alone and has about as many instances as DS2. It is the only dataset to have features with values of a continuous type.

3.4 Attacker Model

The threat – a person trying to phish someone through a website – is modeled as the AST. There are three different ways of categorizing the goal of an attack, as explored by [47], and it is as follows:

1. **Security violations:** Here there are three subcategories of violations *integrity*, *availability*, and *privacy*. The attacker seeks either to enter a system without being detected, or to make the system unavailable to the intended users, or to steal information.
2. **Attack specificity:** Here there are two subcategories, *targeted* or *indiscriminate*. The attacker seeks to fool one system (for a set of samples) or any system (for any given sample).

3. In **error specificity**, there are two subcategories, *specific* or *generic*. The attacker intends to get a sample classified as a certain type or just misclassified.

The attacker's goal in this thesis is categorized as violating integrity – for not being detected by the classifier, targeted, and specific, as we will try to bias the classifier with sets of adversarial generated samples (each set of samples will try to mimic a certain type).

The idea that there are attacks that come from the success from a previous attack is not unfounded [48]. It is possible that a real attacker could have the necessary information before the attack and that an attack could lead to another, but in a different way. The attacker's knowledge in this thesis is the 200 phishing websites that the classifier does not know, all the other phishing websites and the features that the classifier is looking for.

There are basically two types of adversarial attacks: evasion and poisoning [14], [47]. We modeled the attacker influence as using evasion attacks, where it tries to go undetected in the testing phase.

3.5 Adversarial Sampling Technique (AST)

For the AST, we used the Algorithm 1 from Shirazi *et al.* [12] with the difference that it takes the whole training dataset to generate new samples. It should not change anything from the original, as it was implicit that the Algorithm 1 from [12] could have had access to the whole dataset.

3.5.1 Conceptual Overview

For a given instance and some selected features, it will go over the dataset looking for all the unique values in those features of the given type. With all these unique values, it will create all the possible combinations and each combination will be a new sample. The idea is that if a value was found in a phishing instance, for example, then it could be used in another slightly different one.

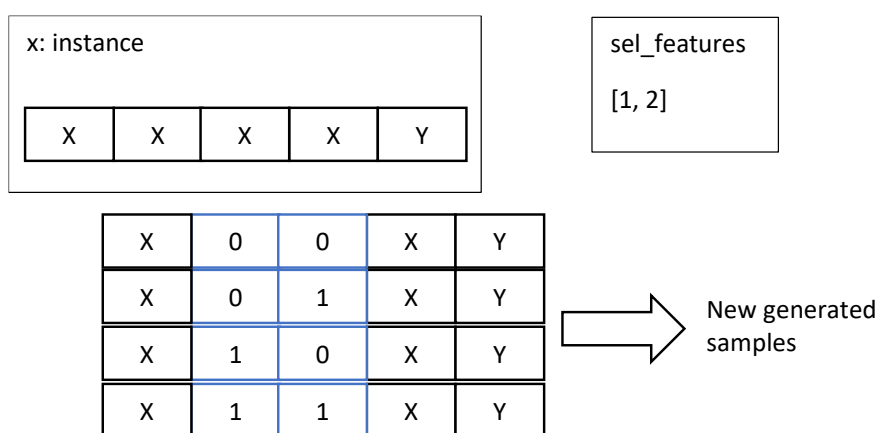


Figure 3-4: Algorithm 1 from [12], illustrated. “x” is an instance with 4 features and a label, and the selected features are the second and the third one. “X” is any value and “Y” is the label of that instance. In this example, all the unique values in columns 1 and 2 with the same label as “x” of the dataset are 0 and 1.

To better explain the algorithm, **Figure 3-4** illustrates how the algorithm works on an example. In the example, the dataset is made of instances of four features, each with binary values. For a given sample and a list of selected features – the second and third one in the example, the algorithm generates more instances from the unique values of all the instances that have the same “Y” value as the “x” instance. It creates all possible combinations of those values and replaces them in the selected features, repeating the other values in the other features as well as the label.

To test how the class imbalance brought by new samples would change and possibly affect the model's performance—should it be used as training sample—the labels were also generated in two other ways. One way is to invert the original labels and the other one is to cluster them into two groups. It is worth mentioning that we are dealing with a dynamic environment.

3.5.2 Feature Selection

The feature selection in [12] was done randomly on purpose and so it is in this thesis. That could lead to selection bias, for in the words of Wikipedia,

“Selection bias is the bias introduced by the selection of individuals, groups, or data for analysis in such a way that proper randomization is not achieved, thereby ensuring that the sample obtained is not representative of the population intended to be analyzed.”

Because of the nature of the features in some datasets such as DS1, it could not be done up to four features as it was in [12]. Each number between the parentheses in **Figure 3-5** is the number of samples generated by a single instance. In the last case of **Figure 3-5**, we are dealing with three manipulated features, which will generate more than 100 million samples, and this process will not only be repeated two more times but will also become more computationally expensive once another feature is manipulated.

```
Going through DS1 1 time
1 0 (2,)
1 1 (2,)
1 2 (2,)
2 0 (18,)
2 1 (18,)
2 2 (18,)
3 0 (520452,)
```

Figure 3-5: Generation of samples. Number of samples generated for each seed sample, according to number of features manipulated.

3.5.3 Sample Seed Selection

Shirazi *et al.* [12] did not say how they reserved the 200 samples for the AST. We assume it is randomly selected. That could lead to selection bias. We tried two more ways of selecting the sample seeds to see how it affects the models' predictions.

Algorithm 1 Dataset Split

Input: $\{X, y, split, type, random_state\}$
Output: $X_{train}, X_{test}, y_{train}, y_{test}, selected$

```

1: Initialize  $selected \leftarrow \emptyset$ 
2: if  $type$  is a class then
3:   initialize random number generator with  $random\_state$ 
4:   while  $length(selected) < split$  do
5:     get random index using random number generator
6:     if  $y[index] == type$  then
7:       add index to  $selected$ 
8:     end if
9:   end while
10:  use  $selected$  to split  $X$  into  $X_{train}$  and  $X_{test}$ 
11:  use  $selected$  to split  $y$  into  $y_{train}$  and  $y_{test}$ 
12: else if  $type$  is a list of indexes then
13:   if  $split > length(type)$  then
14:     use  $type$  to select  $y'$ 
15:     use  $type$  to select  $X'$ 
16:     add all of  $type$  into  $selected$ 
17:      $rad := 1$ 
18:      $dist := 2$ 
19:     while  $split > length(type)$  do
20:       for each  $index$  and  $sample$  in  $X$  that is labeled as phishing do
21:         for each  $sample'$  in  $X'$  that is labeled as phishing do
22:            $c \leftarrow$  calculate distance from  $sample$  to  $sample'$ 
23:           if  $c > 0$  and  $c \leq rad$  and  $split > length(type)$  then
24:             add  $index$  to  $selected$ 
25:           end if
26:           if  $split == length(type)$  then
27:             break
28:           end if
29:         end for each
30:         if  $split == length(type)$  then
31:           break
32:         end if
33:       end for each
34:        $rad := \sqrt{dist}$ 
35:        $dist += 1$ 
36:     end while
37:   else if  $split == length(type)$  then
38:     use  $type$  to split  $X$  into  $X_{train}$  and  $X_{test}$ 
39:     use  $type$  to split  $y$  into  $y_{train}$  and  $y_{test}$ 
40:   else
41:     initialize random number selector with  $random\_state$ 
42:      $selected \leftarrow split$  samples from  $type$  using random number selector
43:     use  $selected$  to split  $X$  into  $X_{train}$  and  $X_{test}$ 
44:     use  $selected$  to split  $y$  into  $y_{train}$  and  $y_{test}$ 
45:   end if
46: end if

```

Figure 3-6: Dataset Split.

The other two ways of selecting the sample seeds is using the support vectors from the linear SVM from sklearn and the support vectors from AAOSVM. This resulted in the lists that had various lengths. To keep it standardized at 200 samples, we used **Algorithm 1**. This algorithm assures that 200 samples are always kept out to be used in the AST. If the number of support vectors is less than 200, it uses the minimal distance from the support vectors to other samples to complete the selection of 200 samples.

3.6 Adversarial Classification

For adversarial classification, we used an AAOSVM based of [19]. The general idea of its workings can be seen on **Figure 3-7**.

Our assumptions from the paper to create the dataset

1. The websites were never changed after the training started, i.e., all the messages that had been maliciously modified were already modified. This means that the classifier actions did not affect the behavior of the adversary.
2. The number of clusters is arbitrarily defined as three, since there are thought to be three types existing, namely (Regular, Fraud), (Malicious, Fraud), and (Regular, Not Fraud).
3. The type of an instance is only which cluster it belongs to [24].

3.6.1 Optimization of the AAOSVM

The optimization in the AAOSVM (SMO) comes from [35].

Definition 6. Cluster Type (z): z is the cluster of which an instance can belong to. An instance can belong to any of three clusters, namely z_1 , z_2 , and z_3 ; the union of these clusters make Z . These clusters are created using the KMeans algorithm with random state parameter set to 42, found on the sklearn python library.

Definition 7. \vec{x}_j : the observed instance.

Definition 8. $p(z)$: this function computes the probability of type z . It is calculated by counting all the instances of type z and dividing by the number of instances in the window.

Definition 9. $p(M, z)$: this function computes probability of type z and $y = 1$. This represents the malicious type in cluster z . It is calculated by counting all the messages of type z that also have $y = 1$ divided by the number of messages in the window.

Definition 10. $\phi(M|z)$: this function computes the probability of instance \vec{x}_j to be malicious, given z , and is computed as **Eq. 3-16** [49]:

$$\phi(M|z) = \frac{p(M, z)}{p(z)} \quad \text{Eq. 3-14}$$

Definition 11. $\mu((y, z) | \vec{x}_j)$ is the consistent belief represented in **Eq. 3-15**.

$$\mu((y, z) | \vec{x}_j) = \begin{cases} \frac{p(z_j)\phi(M|z_j)}{p(R) + p(M)\phi(M|z)}, & \text{if } y = M \\ \frac{p(z_j)}{p(R) + p(M)\phi(M|z)}, & \text{if } y = R \\ 0, & \text{if } y = R \text{ and } z_j \cong z \end{cases} \quad \text{Eq. 3-15}$$

where z_j is the predicted cluster of \vec{x}_j , $p(R)$ is the probability of an instance to be regular, given the instances in the window, and $p(M)$ is the probability of an instance to be malicious, given the instances in the window.

Definition 12. Scores: The scores are one of two types: utility (ϵ) or cost (γ). Each type of score will be further divided to keep score of malicious and legitimate samples.

Definition 13. $\psi(\vec{x}_j)$: the function holds the prior knowledge that is based on probabilities and scores. It is defined in **Eq. 3-16**. It is updated every Gp periods or number of samples, as stated in [19]:

$$\psi(\vec{x}_j) = \frac{\sum_{z \in Z} \mu((M, z) | \vec{x}_j) \cdot (\epsilon_M + \gamma_M)}{\sum_{z \in Z} \mu((R, z) | \vec{x}_j) \cdot (\epsilon_R + \gamma_R)} \quad \text{Eq. 3-16}$$

Definition 14. $\Psi(\vec{x}_j)$: the function that is defined in **Eq. 3-17**. Adds prior knowledge to the training criteria (**Eq. 3-5**):

$$\Psi(\vec{x}_j) = \frac{1 + \psi(\vec{x}_j)}{w^T e + 2b} \quad \text{Eq. 3-17}$$

Now that $\Psi(\vec{x}_j)$ is defined, it is incorporated to **Definition 5**. The criteria used in **Figure 3-7** that decides if the SVM needs to be trained again is based off the intuition from **Eq. 3-18**:

$$y\hat{y}\Psi(\vec{x}) < v \quad \text{Eq. 3-18}$$

where $y \in \{+1, -1\}$ is the label of the instance, $\vec{x}, \hat{y} \in \{+1, -1\}$ is its predicted label, and $v \in (0,1]$ is a threshold value. From that, we can say that the only two ways of making the equation correct is to misclassify the sample or to have $\Psi(\vec{x}) < v$ (or a poorly classified sample).

Definition 15. Update parameter (u): a parameter that will tell the AAOSVM if it should change the scores or not.

Definition 16. Difference of errors (Δ_{error}): when training, if the training criteria is true, the AAOSVM will save the errors prior to training and compare with the new errors. This comparison is used as part of the conditions to decide which score should be updated, if $u == True$.

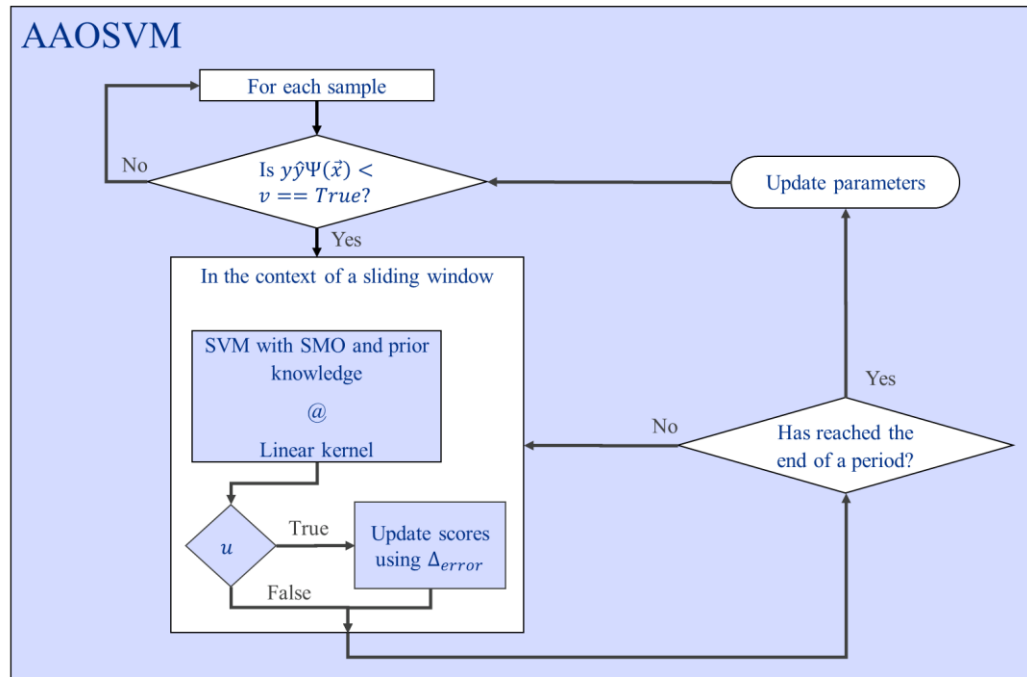


Figure 3-7: Overview of the inner workings of the AAOSVM of Dynamic Scoring.

Some changes needed to be made in the AAOSVM from [19]. Firstly, the *traceZeroes* procedure is not further explained, as “*based on input parameters from Gambit for QRE computation*” is not enough and is not publicly available; thus, we adopted the strategy of changing scores. Second, the true output of the algorithm is the optimal strategy for the classifier, as it would serve no purpose otherwise.

3.6.2 Bias in the Algorithm

From **Eq. 3-16** and **Eq. 3-15**, it is proven that the algorithm will be biased towards malicious messages. The more a type appears, the bigger the belief will be about that type. The bigger the belief that a message is malicious, the bigger $\Psi(\vec{x})$ will be; on the other hand, the bigger the belief that a message is regular, the smaller $\Psi(\vec{x})$ will be.

3.6.3 Scoring Each Sample

When training, the **proposed Algorithm 2** changes the scores for each label based on the true label and the difference between previous and actual error, depending on the value of u .

The intuition is that the score can double or zero its value if the Δ_{error} is too great or have a change that is proportional to the error difference. We expect the scores to either settle at zero or at around some value, as the error fluctuate and gradually go to a minimum.

To update the utility of a malicious sample, we decreased each sample's respective utility by itself, scaled by $\tanh (error)$, as shown in **Eq. 3-19**, if the *error* decreased as well. The intuition is that it will increase the utility of the malicious sample as the error goes down:

$$\epsilon_M -= \epsilon_M \times \tanh (error) \quad \text{Eq. 3-19}$$

The same happens for a regular sample, as shown in **Eq. 3-20**, if the *error* decreased. The intuition is that it will increase the utility of the regular sample as the error goes down:

$$\epsilon_R -= \epsilon_R \times \tanh (error) \quad \text{Eq. 3-20}$$

In a similar way, we update the costs as shown in **Eq. 3-21** and **Eq. 3-10**. Now, the costs go up as the error goes up:

$$\gamma_M += \gamma_M \times \tanh (error) \quad \text{Eq. 3-21}$$

$$\gamma_R += \gamma_R \times \tanh (error) \quad \text{Eq. 3-22}$$

Algorithm 2 Adversary-Aware Online SVM of Dynamic Scoring

Input: $\{\tau, m, v, G_p, C, k, u, \epsilon_R, \gamma_R, \epsilon_M, \gamma_M\}$
Output: $\text{predict}(x) = \text{sign}(w^T \cdot x + b)$

```

1: Initialize  $w_0 := 0, b_0 := 0, S \leftarrow \emptyset$ 
2: for each  $(x_\tau, y_\tau) \in T$  do
3:   if  $y_\tau(w_{\tau-1}^T \cdot x_\tau + b_{\tau-1})\Psi(x_\tau) < v$  then
4:      $\text{errors} := \text{predict}(S) - S_{\text{true\_labels}}$ 
5:     Find  $w', b'$  with prior knowledge SMO on  $S$ , with  $w_{\tau-t}$  and  $b_{\tau-t}$  as seed
     hypothesis, and  $\Psi(x_\tau)$ 
6:     if  $u == \text{True}$  then
7:       for each  $(\text{error}, \text{label}) \in \text{errors}, S$  do
8:         if  $\text{error decreased and label} == P$  then
9:            $\epsilon_M -= \epsilon_M \times \tanh(\text{error})$ 
10:        else if  $\text{error decreased and label} == L$  then
11:           $\epsilon_R -= \epsilon_R \times \tanh(\text{error})$ 
12:        else if  $\text{error increased and label} == P$  then
13:           $\gamma_M += \gamma_M \times \tanh(\text{error})$ 
14:        else if  $\text{error increased and label} == L$  then
15:           $\gamma_R += \gamma_R \times \tanh(\text{error})$ 
16:        end if
17:      end for
18:    end if
19:    Set  $w_\tau := w', b_\tau := b'$ 
20:  end if
21:  if  $|S| > m$  then
22:    Remove oldest example from  $S$ 
23:  end if
24:  if  $T \bmod G_p = 0$  then
25:    Initialize  $\{c_i\}_{i=1}^k = \text{getClusters}(S, k)$ 
26:     $p(t_i) = |\{x | x \text{ is a message in cluster } c_i\}| / |S|, \forall i \in \{1, \dots, k\}, \forall x \in S$ 
27:    update  $\Psi(x_i), \forall i \in S$ 
28:  end if
29:   $S.\text{add}(x_\tau, y_\tau)$ 
30: end for

```

Figure 3-8: Adversary-Aware Online SVM of Dynamic Scoring.

CHAPTER 4

RESULTS AND DISCUSSION

As explained in **Section 3.2.1**, the experiment results were obtained by removing 200 samples from each dataset. To better compare the results, the same 200 samples were removed from all classifiers. Every time an experiment was repeated, new 200 random samples were chosen. From the remainder of each dataset, a 5-fold cross validation was performed, measuring the time it took to do the cross validation. The only classifier that was scored on the adversarial samples was the AAOSVM and it was only scored on bias metrics. In the previous chapters we have shown in how many places bias could be introduced during the training or testing of a classifier and tried to minimize them as best as we could so we could analyze only the effects of the bias in the scope of this thesis.

4.1 Experiment 1 - Evaluation of Machine Learning Models

The classifiers used were Decision Tree (DT), Gradient Boosting (GB), Random Forest (RF), K-Nearest Neighbors (KNN), SVM with two different kernels: Linear (lin) and Gaussian (rbf), and online SVM on a moving window of 100 samples. They all are from the sklearn library. Except for **Figure 4-19** and **Figure 4-20**, all the graphs in this **Section 4.1** and its **subsections 4.1.1** and **4.1.2** show the average of the 10 runs, with each run having its own standard deviation from its cross validation.

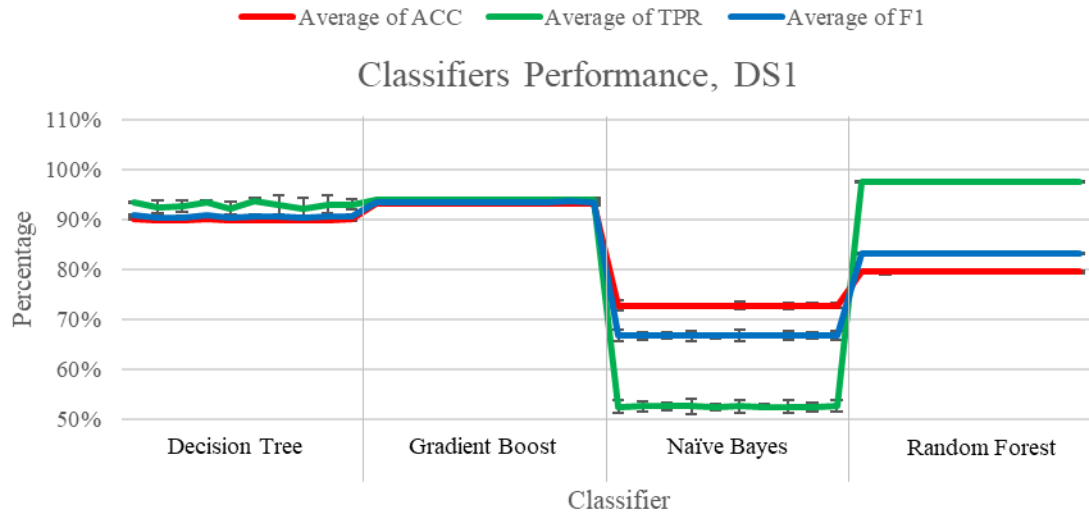


Figure 4-1: Performance metrics on machine learning models on dataset DS1.

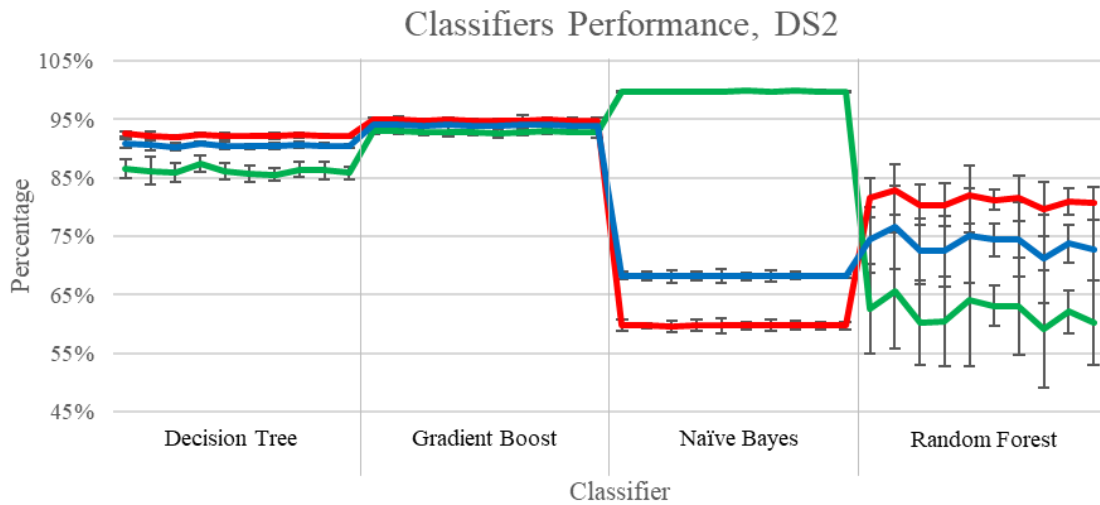


Figure 4-2: Performance metrics on machine learning models on dataset DS2.

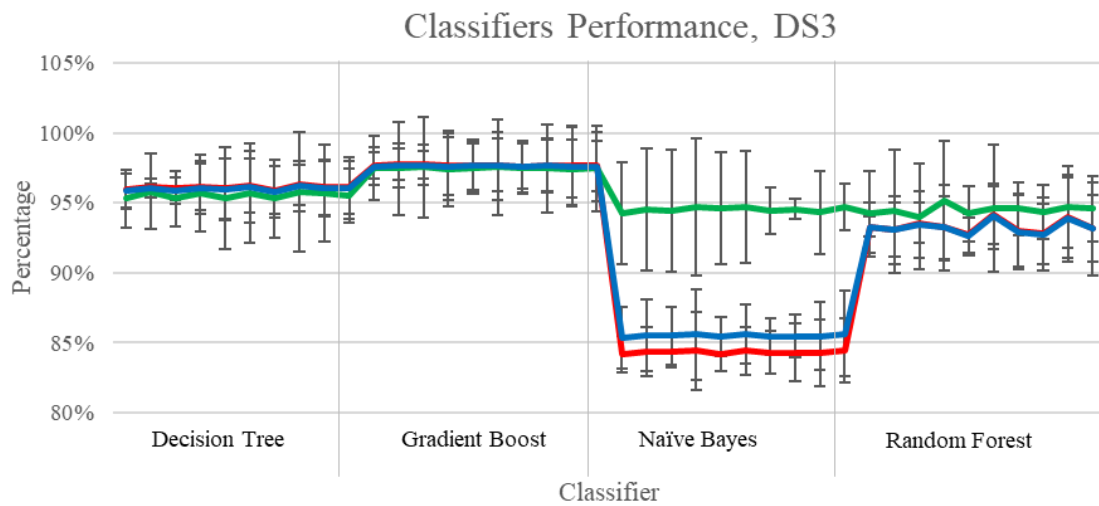


Figure 4-3: Performance metrics on machine learning models on dataset DS3.

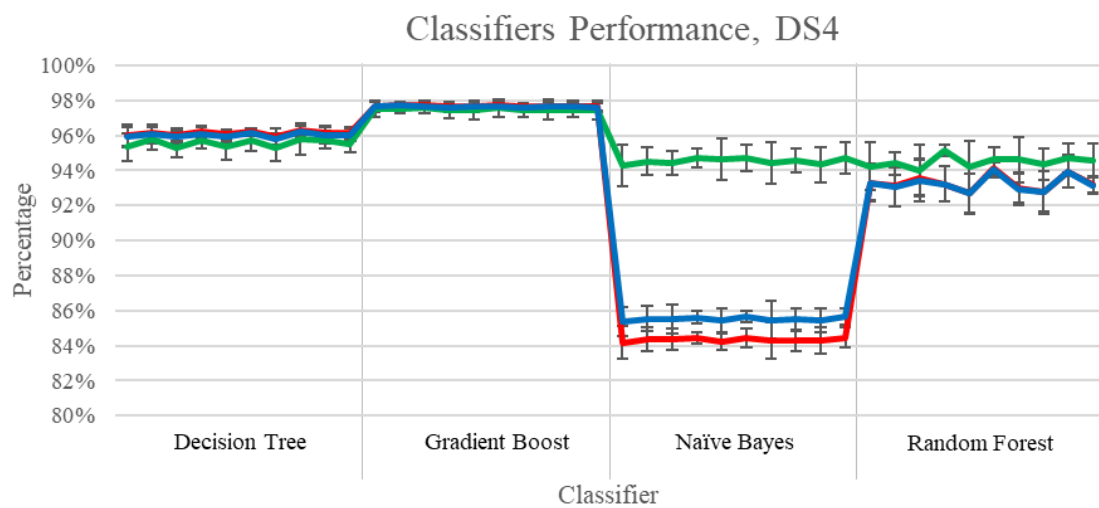


Figure 4-4: Performance metrics on machine learning models on dataset DS4.

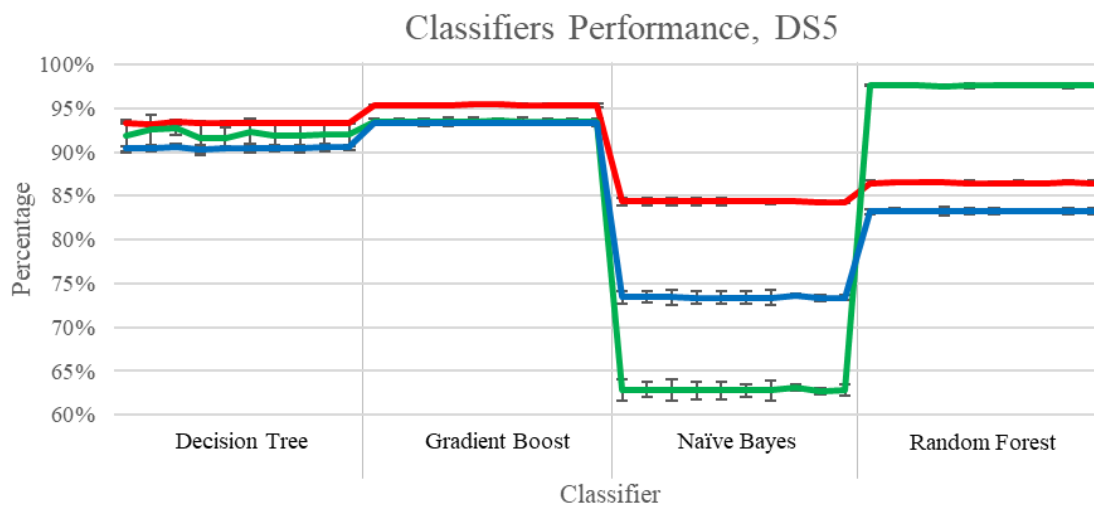


Figure 4-5: Performance metrics on machine learning models on dataset DS5.

From **Figure 4-5**, **Figure 4-2**, **Figure 4-3**, **Figure 4-9**, and **Figure 4-5**, we can see that Gradient Boost had the best performance overall on all datasets without normalization. **Figure 4-3** shows us the effect of DS3 on the classifiers' performance, a small dataset with only 9 features, when we remove 200 samples from it. We think that the reason for the DS3 to have such a variation is that 200 samples represent almost 15% of the total dataset and almost 30% of the total number of phishing instances.

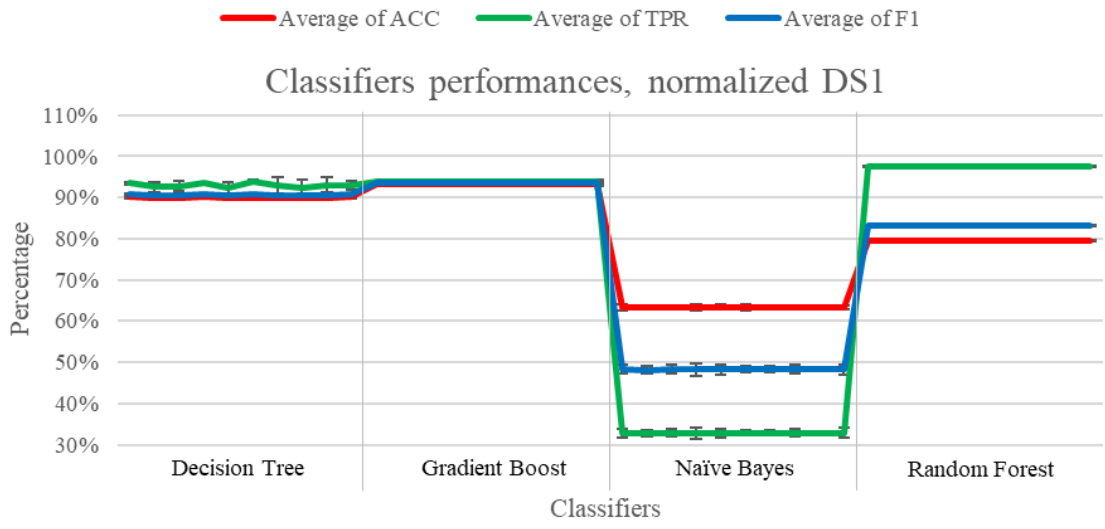


Figure 4-6: Performance metrics on machine learning models with normalized data of dataset DS1.

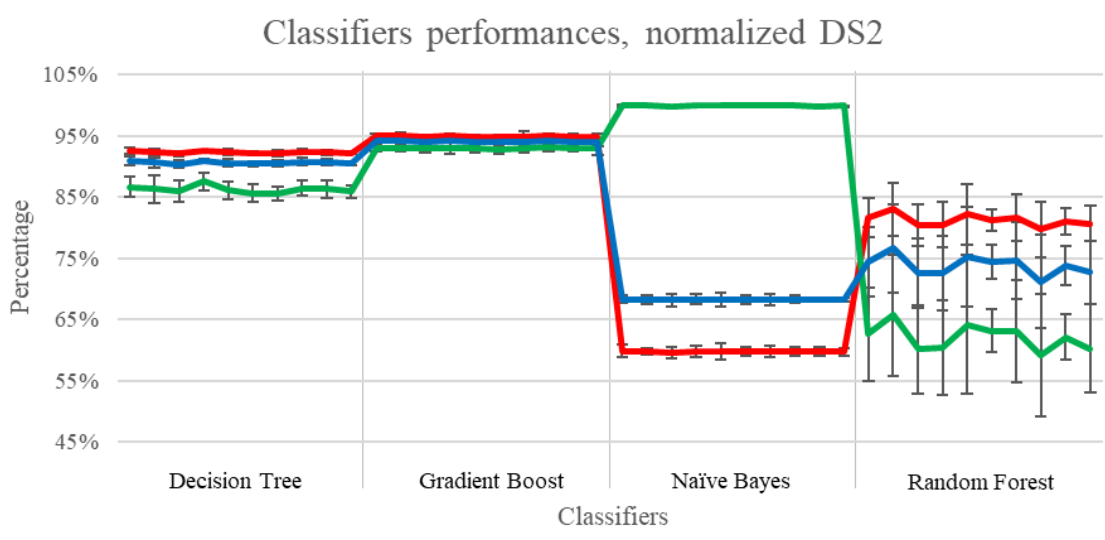


Figure 4-7: Performance metrics on machine learning models with normalized data of dataset DS2.

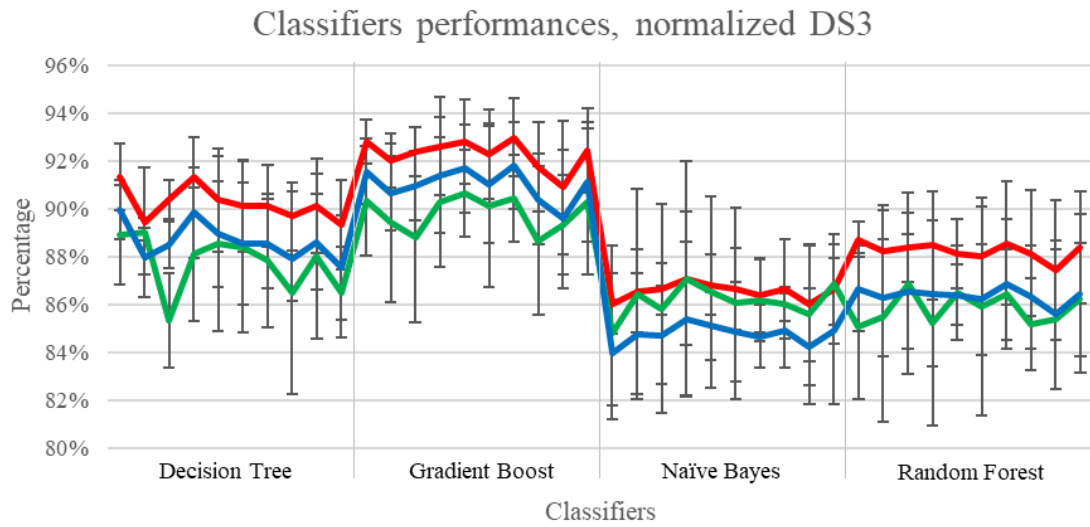


Figure 4-8: Performance metrics on machine learning models with normalized data of dataset DS3.

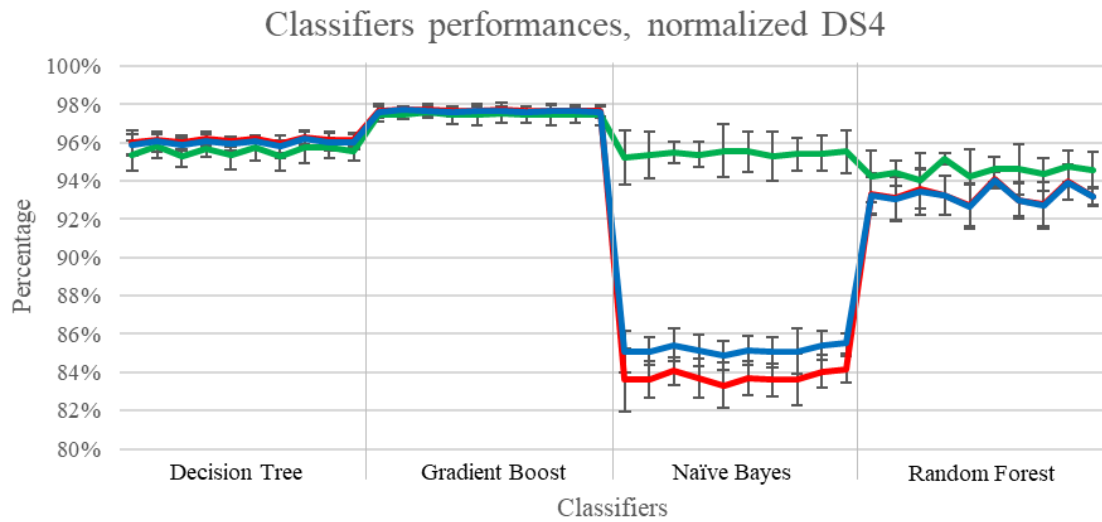


Figure 4-9: Performance metrics on machine learning models with normalized data of dataset DS4.

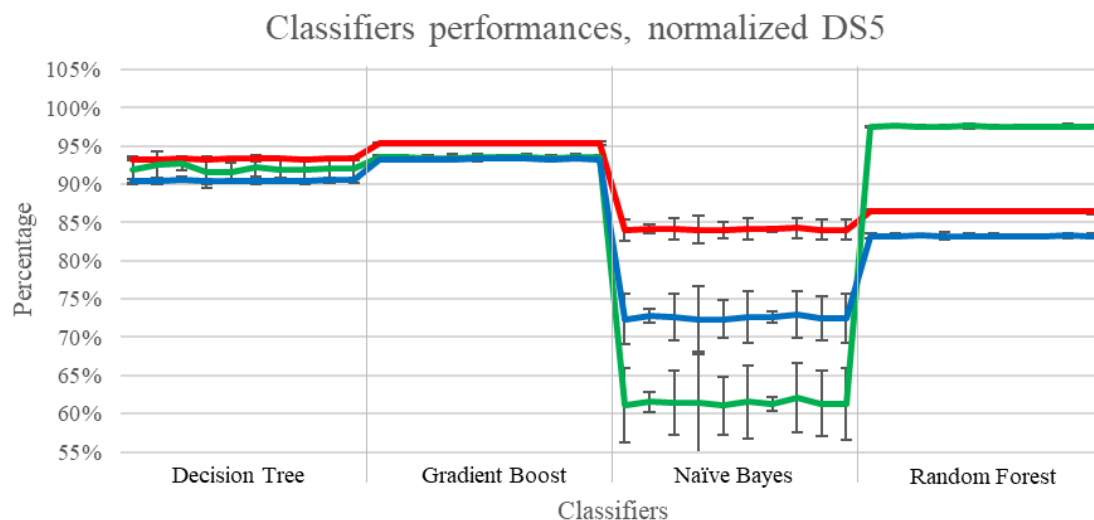


Figure 4-10: Performance metrics on machine learning models with normalized data of dataset DS5.

We can see in **Figure 4-8** that the normalization using minmax decreased the classifiers performance robustness for DS3, i.e., now the variations have a greater impact on the classifiers' performance. Normalization also had a bigger impact on Naïve Bayes standard variation on each run of DS5, as shown in **Figure 4-10**.

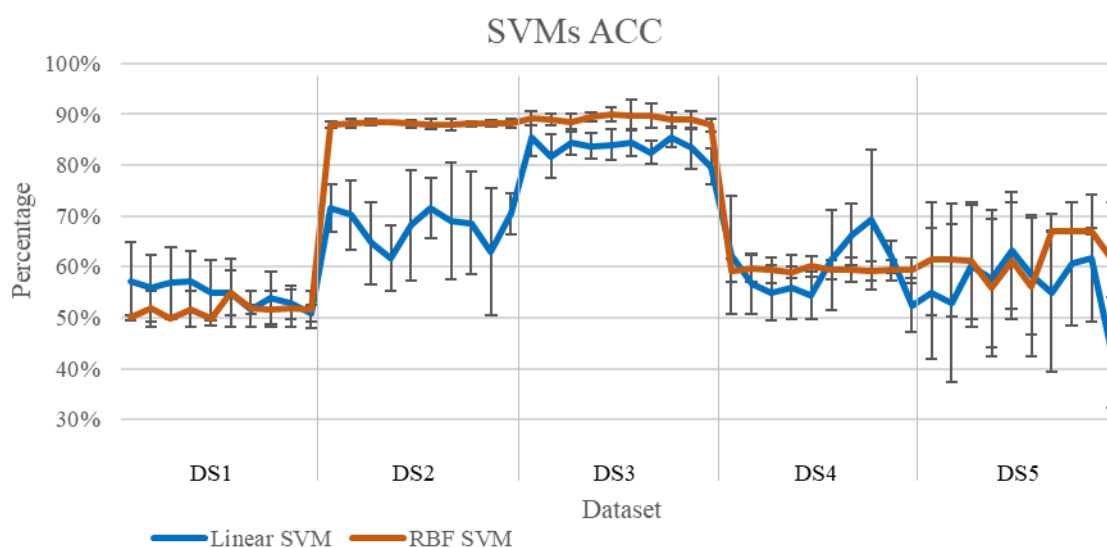


Figure 4-11: SVMs as compared on ACC.

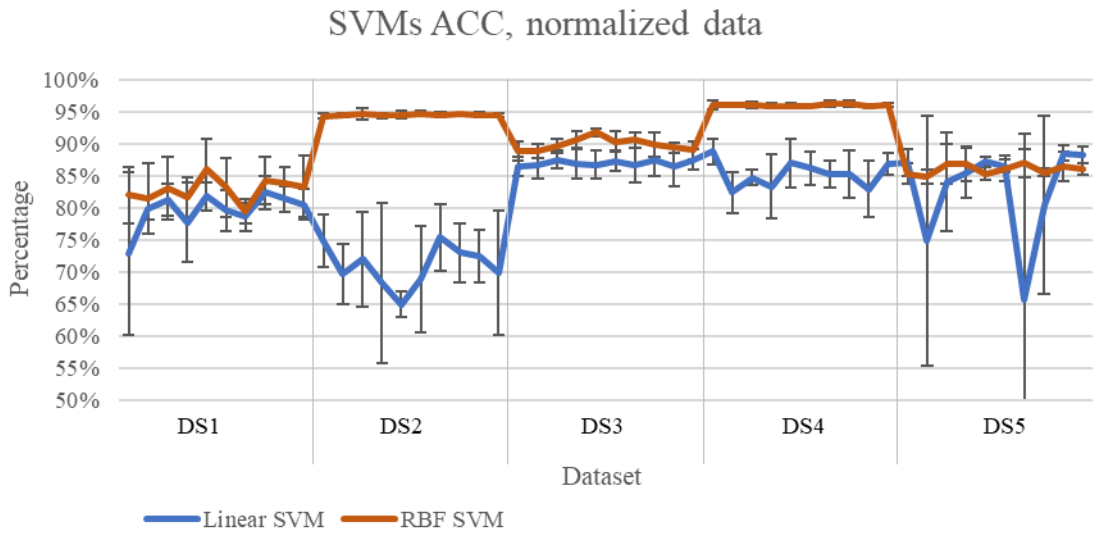


Figure 4-12: SVMs as compared on ACC with normalized data.

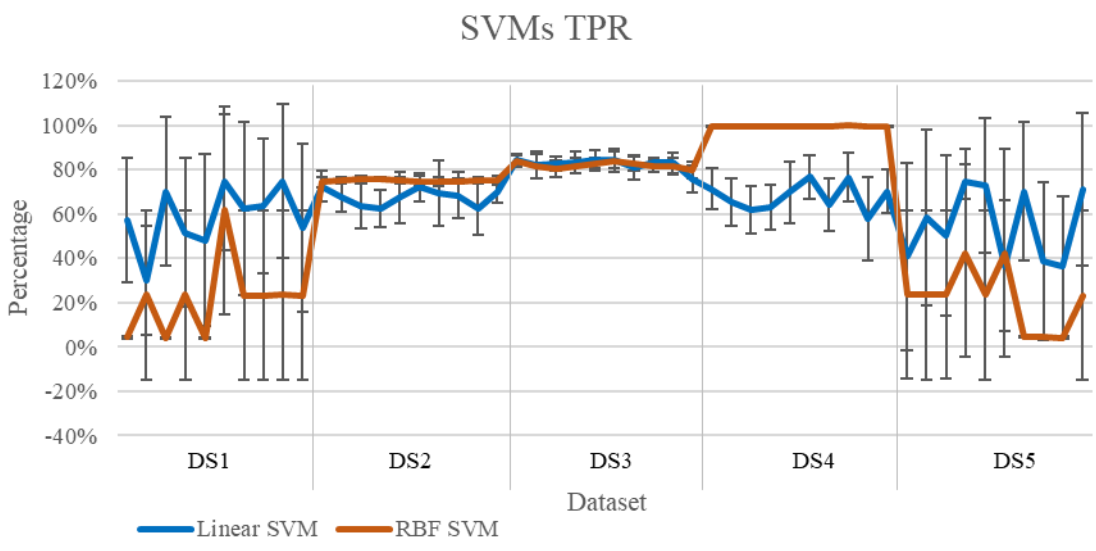


Figure 4-13: SVMs as compared on TPR.

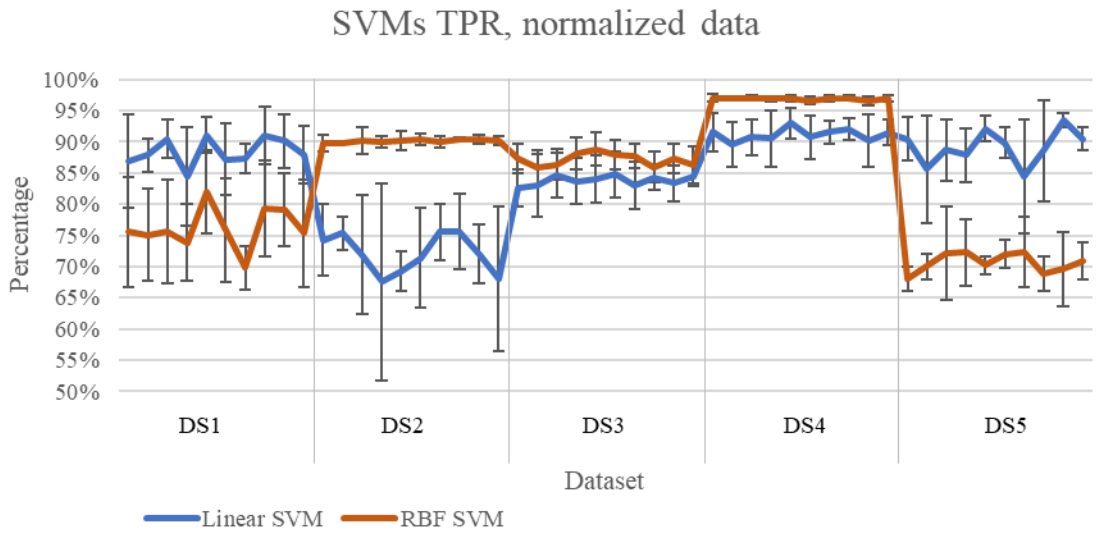


Figure 4-14: SVMs as compared on TPR with normalized data.

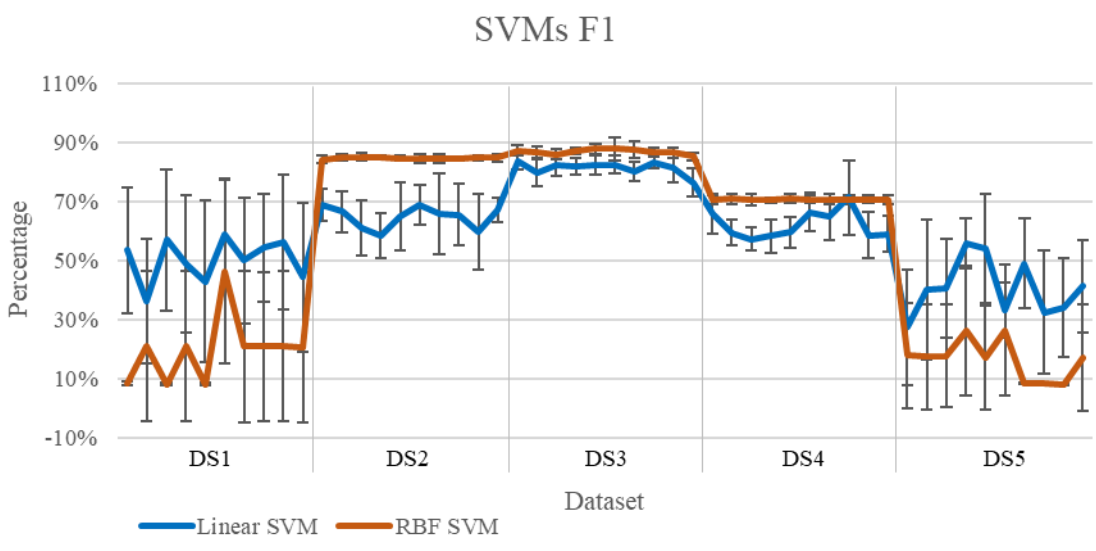


Figure 4-15: SVMs as compared on F1.

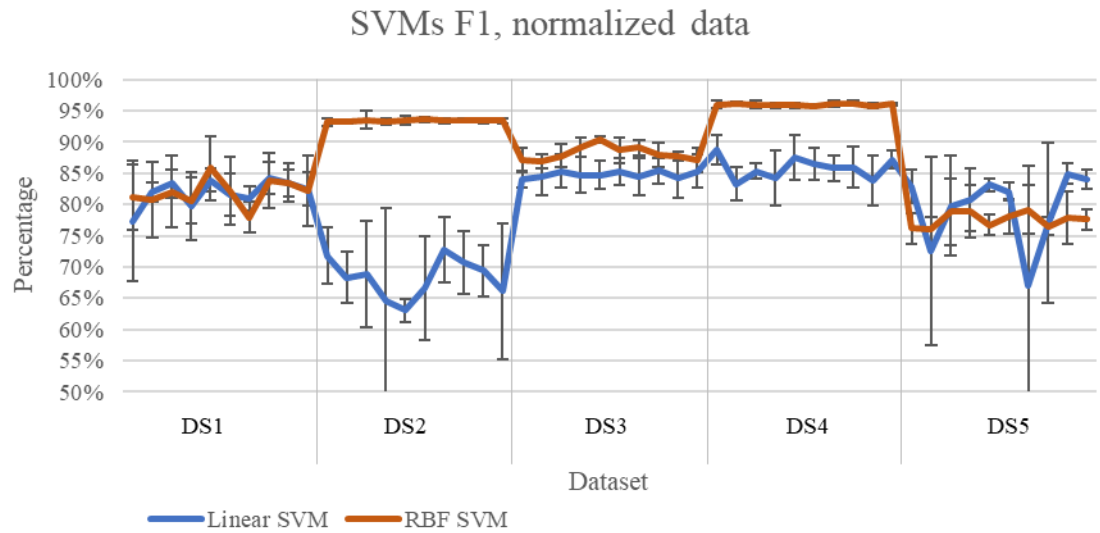


Figure 4-16: SVMs as compared on F1 with normalized data.

Looking at **Figure 4-11**, **Figure 4-12**, **Figure 4-13**, **Figure 4-14**, **Figure 4-15**, and **Figure 4-16**, we can see that the performance and standard deviation was greatly impacted by the normalization of the data. DS1, DS4, and DS5 had all the metrics improve with normalization of the data for both SVMs. DS2 had all the metrics improve with normalization of the data for both SVMs, but it was subtler for the Lin SVM. DS3 had all the metrics stay with about the same values, with or without the normalization of the data for both SVMs.

Other deductions that can be made is that Lin SVM is more sensitive to which the samples are being reserved. RBF SVM performance metrics had a variation of around 1% for datasets 1 and 3, for the same type of data, which could indicate that it would be more robust for those datasets, i.e., reserved samples do not affect its performance too much.

4.2 Experiment 2 - Performance of the Online SVM on a Moving Window

Now that the baseline for the SVM has been established, we wanted to see how an Online SVM (OSVM) would perform on a moving window. This will not only give a

more realistic baseline to the AAOSVM but will also show the reliability of an OSVM on a moving window and show the importance of the features being in the range of $[0,1]$ for a better general performance.

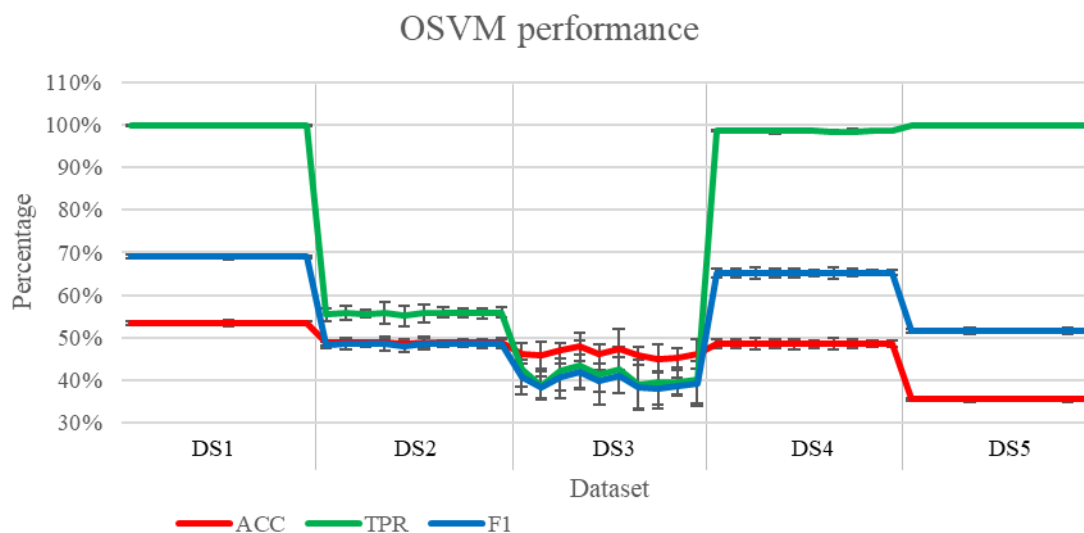


Figure 4-17: Performance metrics of OSVM.

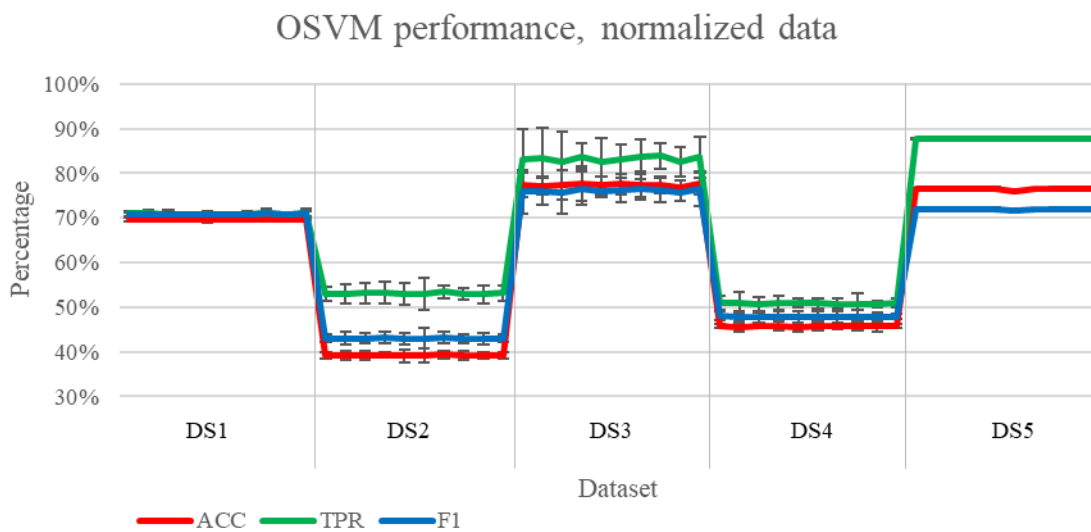


Figure 4-18: Performance metrics of OSVM with normalized data.

The only dataset that did not benefit from the normalization was DS2, probably because its data was already mostly binary. Based on the difference between the results

before and after normalization, we can see that when the features are engineered to be all in the range of [0,1], the performance of the SVM could be better.

This is especially evident for DS1 and DS5, where it goes from classifying every instance as phishing to trying to predict different labels correctly, having the accuracy go from below 55% to above 70%. DS3 also an increase in the accuracy, going from below 50% to above 75%. While DS4 accuracy decreased, at least the TPR was not as close to 100% as it was before, meaning that it was trying to predict different labels more often.

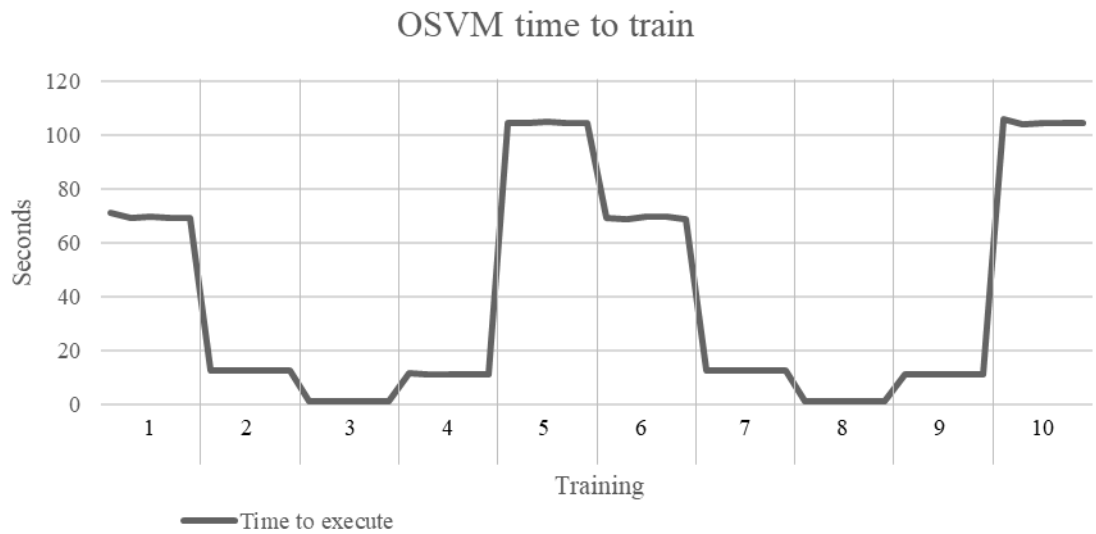


Figure 4-19: Time to execute each run of the cross validation of the OSVM.

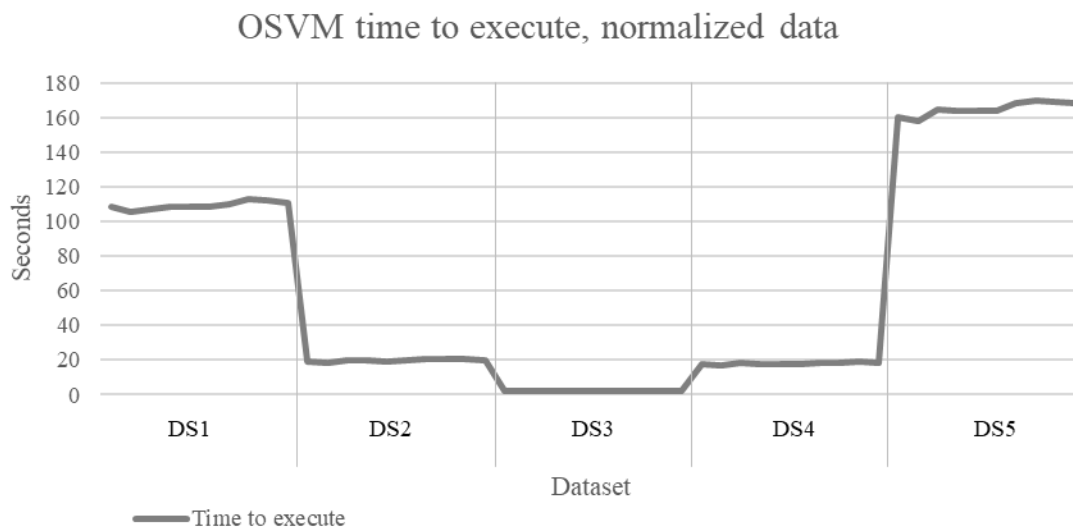


Figure 4-20: Time to execute each run of cross validation of the OSVM with normalized data, according to each dataset.

Although it is not in the scope of this thesis, we had an interesting find. During the analysis of the data, we found that the time to train before normalization was dependent on the run, while after normalization it was dependent on the dataset. When it comes to timing, there are two considerations to be done. One is how long it takes to train the model. Another is the number of features and the number of samples affect the training time.

4.3 Experiment 3 - SVM Becoming Adversary-Aware

For here and onwards, the experiments had access to only the data without normalization as it would have been in a real-world scenario. In this experiment, a random combination of features, up to 2 features, was used each time.

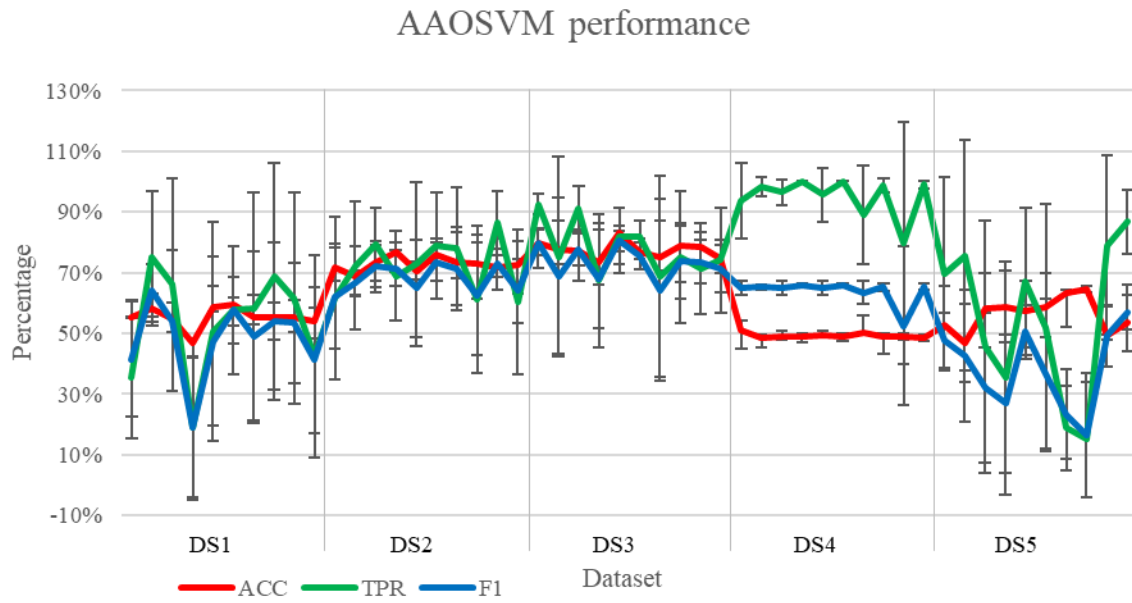


Figure 4-21: Performance metrics of AAOSVM, running 10x each dataset without 200 random samples.

Looking at **Figure 4-21**, we can see that the performance metrics are dependent on the run and dataset.

4.4 Experiment 4 - Changing the Scores

For here and onwards, the experiments were done with just one run and on holdout instead of 5-fold cross validation.

Table 4-1: Performance metrics of AAOSVM changing scores, running 1x each dataset without 200 random samples on a holdout validation.

Dataset	ACC	TPR	F1
DS1	11.09%	19.34%	13.85%
DS2	17.18%	17.04%	16.77%
DS3	16.45%	17.98%	16.26%
DS4	9.43%	19.08%	12.67%
DS5	12.86%	0.35%	0.64%

We can see from **Figure 4-21** and **Table 4-1** that changing the scores had a great impact on the AAOSVM. ACC went from around 50% to around 11% on DS1, from around 70% to around 16% on DS2 and DS3, from around 50% to less than 10% on DS4, and from around 60% to around 13% on DS5. In addition, it was even more critical on DS5, with TPR and F1 near 0, which means that it predicting almost every instance as being legitimate.

To analyze the changes in the scores, we picked DS4 and DS5, DS4 for being a balanced dataset on both class labels and feature types, and DS5 for being the biggest and the most imbalanced of them all. Because of the disparity of the number of samples needed for a score to reach zero, the scores were analyzed based on the first 200 samples.

It was already foreseen that once a score reaches zero, it has no way of coming back, i.e., it will stay at zero until the classifier is reset. As stated in **Section 3.6.3**, we thought that using tanh as the update function, the scores would go up and down, and either settles at zero or at around some value, as the error fluctuates and gradually goes to a minimum.

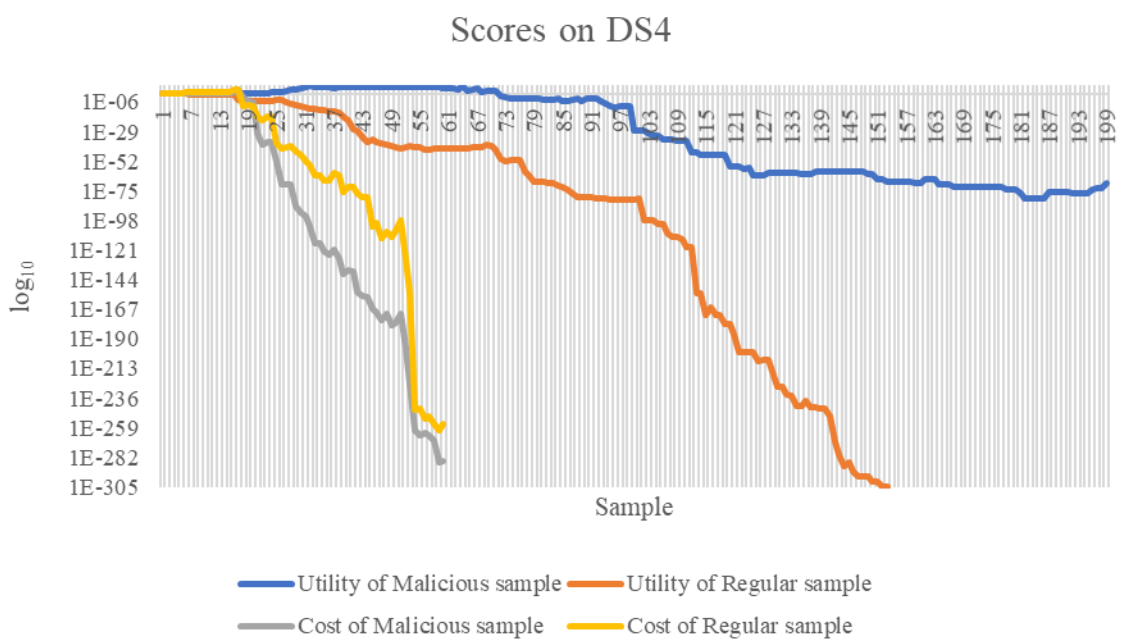


Figure 4-22: Score values for DS4, on 200 samples.

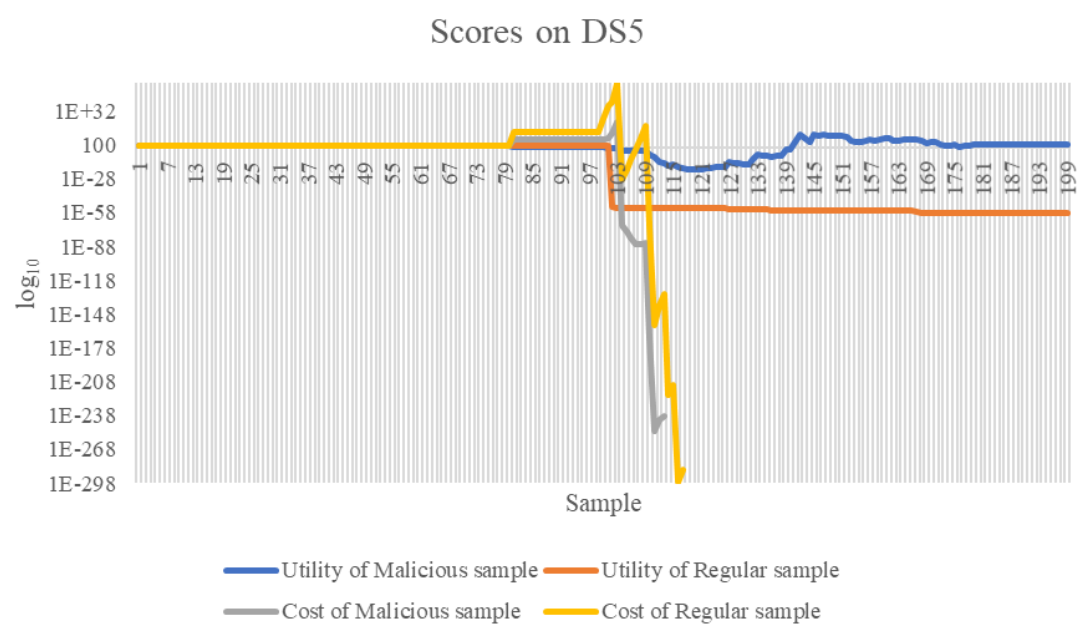


Figure 4-23: Score values for DS5, on 200 samples.

The intended desired behavior is the one displayed by the utilities on DS5 and the utility of malicious samples on DS4, as shown in **Figure 4-22** and **Figure 4-23**.

We did not foresee the number of samples that was needed for each score to reach zero nor that it would happen with the error still high. It is worth noting that once both scores for a class reach zero, $\Psi(\vec{x}_i)$ will be redefined as **Eq. 4-1**, where it no longer depends on the sample or the scores:

$$\Psi = \frac{1}{w^T e + 2b} \quad \text{Eq. 4-1}$$

The costs on DS4 and DS5 went to zero with less than 120 samples, as shown in **Figure 4-22** and **Figure 4-23**. The utility of regular samples on DS4 with less than 160 samples means that for DS4 the scoring had no effect on $\Psi(\vec{x}_i)$ after only around 1% of the samples. As both costs reach zero, they leave the scaling of the belief to just the utilities, which are updated based on the decrease in error or correct classification of the samples.

4.4.1 Significance of Experiment 3 and 4 – Bias Metrics

For the remainder of the results and discussion,

- “a” represents that the support vectors from the previous AAOSVM are being used to select the 200 samples that will be left out of training.
- “b” represents that the support vectors from the Linear SVM are being used to select the 200 samples that will be left out of training.
- “n” represents that the 200 samples that will be left out of training are randomly selected.

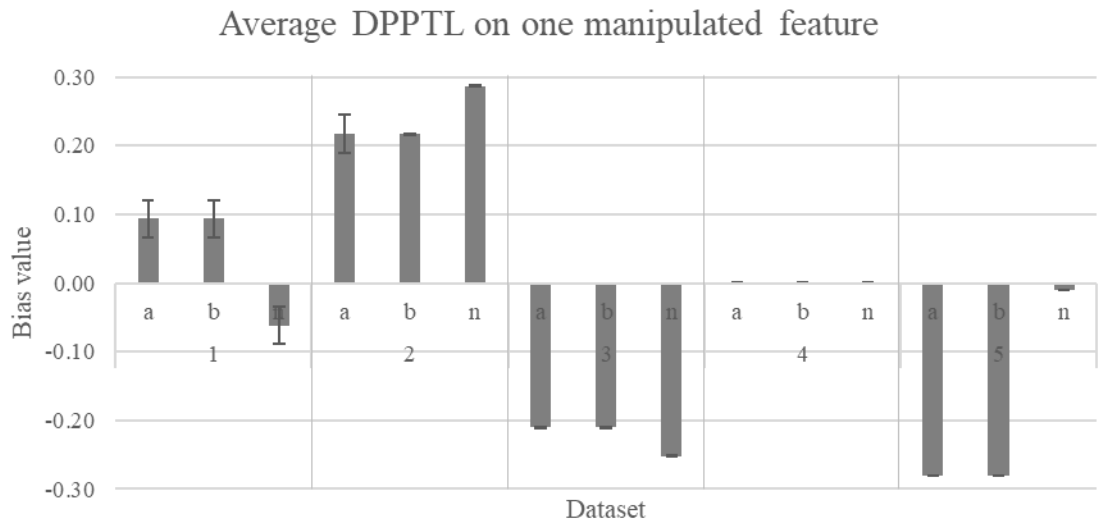


Figure 4-24: Average DPPTL on one manipulated feature.

As stated in **Section 3.2.2**, there were 5 metrics for measuring bias but only 2 are shown here and another 2 are shown in the next **Section**; this is because demographic parity was not attained on a single instance. An important thing to be mentioned is that not a single generated sample resulted in a false positive in DS4, even though the one with almost 0% TPR and F1 was DS5.

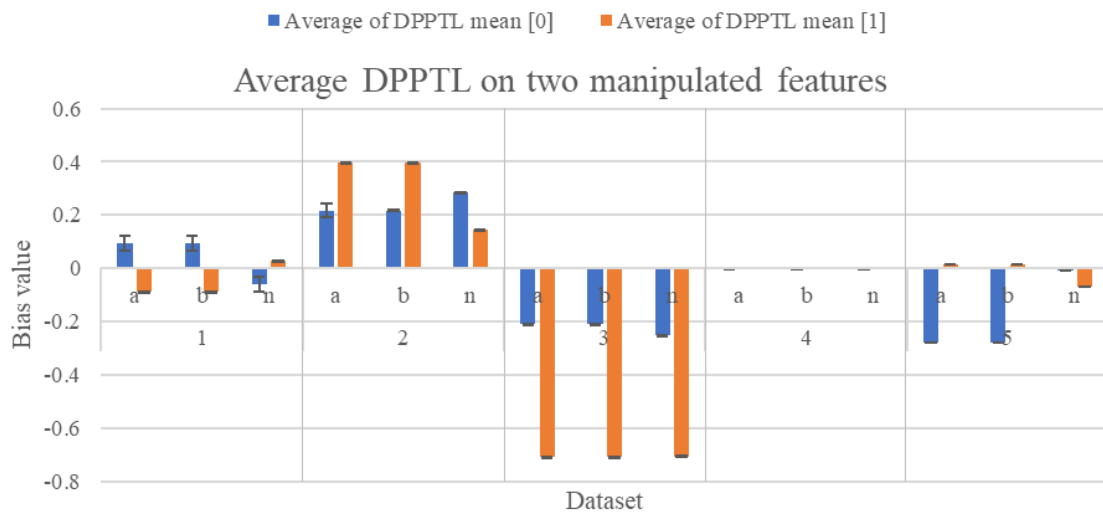


Figure 4-25: Average DPPTL on two manipulated features.

Demographic Parity and Difference in Positive Proportions of True Labels did not carry as much information as expected. We expected to see at least a different DPPTL value for each set of generated samples, but all the sets of generated samples in each dataset with a set condition generated an average of about four distinct values of DPPTL.

Table 4-2: Distance from the original sample. All the selected samples' type have the same values for each respective dataset.

Dataset	<i>Distance</i>
DS1	1
DS2	2
DS3	1
DS4	0
DS5	1

It did not matter what instances were kept out of the training. It did not matter if the AAOSVM changed scores. The minimal distance from the original sample remained constant for each dataset. All the generated samples from an instance had at least one sample that had the same minimal distance for the given dataset.

An important thing to consider is the cost of generating samples in the real world. While it is reasonable to consider Euclidean distance from an academic point of view, it is not reasonable in real life, as some features will be harder to modify than others and some datasets have half or more features with binary values while others have ninety percent of numerical values. Not to mention that, at least in the case of most website phishing, there is a need to modify one feature on every sample – named URL. It is also

important to consider how many features are being manipulated. At some point it will be similar to creating a new instance from scratch.

4.4.2 Significance of Experiment 3 and 4 - Testing for False Positives

Here, we have the results of percentage of false positives and class imbalance.

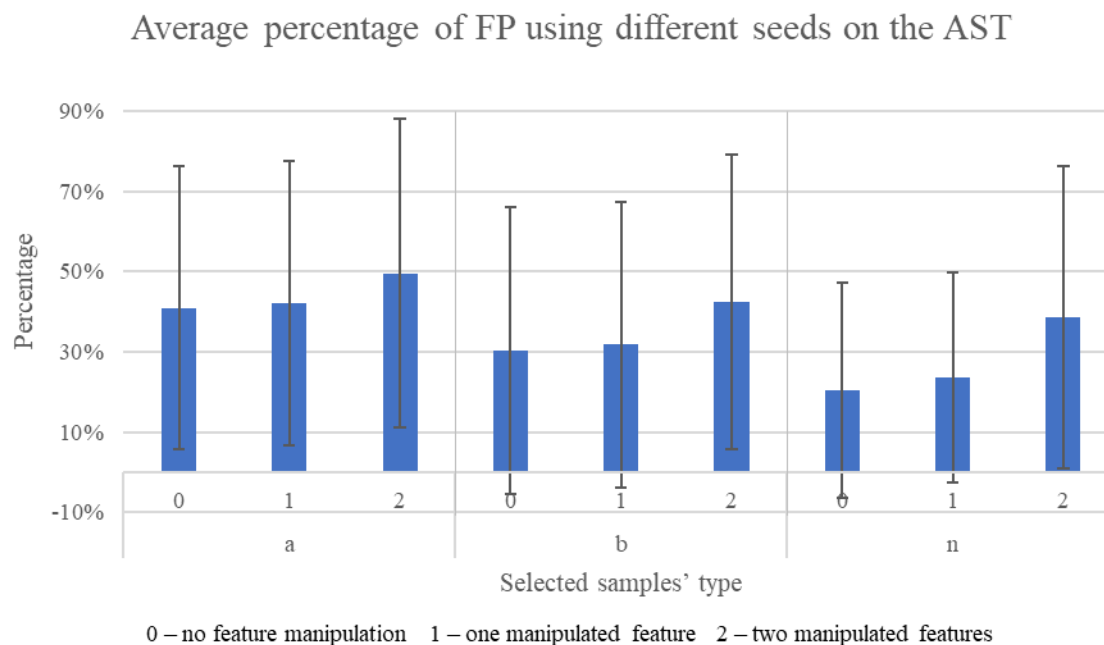


Figure 4-26: Average percentage of FP using different seeds on the AST for different number of manipulated features.

As stated in **Section 1.4**, we had three research questions. The first one was confirmed, as shown in **Figure 4-26**. If support vectors are used as AST seed samples, then the percentage of false positives can go up at least 10% in absolute number, or 25% relative to the randomly selected samples, and up to 20% in absolute number, or 100% relative to the randomly selected samples.

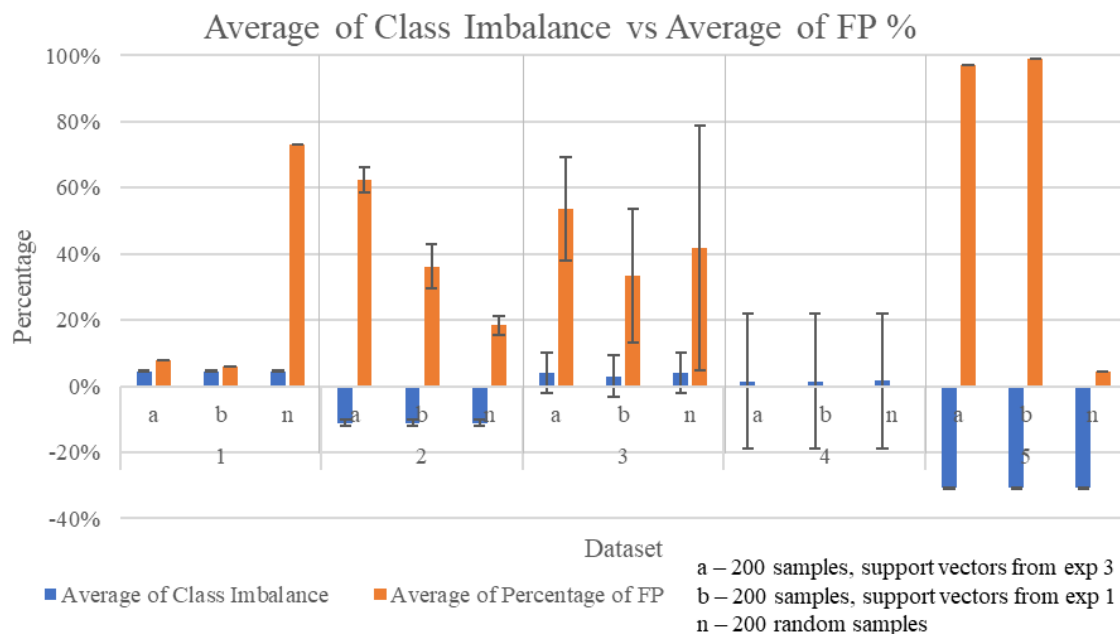


Figure 4-27: Comparative of Class Imbalance and Percentage of FP on each dataset for each type of sample seed.

The reason why **Figure 4-26** had a big variation in the standard deviation is shown in **Figure 4-9** because the percentage of FP varies for each dataset. The second one was dismissed as there is no correlation between the class imbalance and the percentage of false positives as shown in **Figure 4-9**. In all datasets that had false positives with the exception of DS3, the class imbalance was almost constant, with the standard variation less than 1%. The class imbalance in DS3 varied more because of its size. Although the class imbalance had some variations, it did not change with the selected samples' type but the dataset. The FP percentage varied according to the dataset and selected samples' type as shown in **Figure 4-9**.

On the one hand, small perturbations on some features can bypass the AAOSVM and bring down the accuracy. On the other hand, even in the worst case, where the AAOSVM had terrible performance during training and was facing generated samples

designed to fool the classifier, the percentage of false positives did not go over 50%. When it comes to online classifiers, adversarial attacks have an even bigger impact as they can bias the classifier towards one class or the other. When the AAOSVM is changing scores during training, its performance decreases, but the decrease in performance in training is not linearly correlated with its bias metrics values. We say this because while clearly the choice of the seed matters, as shown in **Figure 4-26**, the change in FP occurs in a similar way but with different proportions, except for DS3, in **Figure 4-9**.

Focusing on **Figure 4-9**, DS1 with seed “a” has a slightly higher percentage of FP than “b” but both are lower than “n”; DS2 with seed “a” has a higher percentage of FP than “b” but both are higher than “n”; DS3 follows a similar pattern of DS2 with the difference that on DS3 “b” has a lower percentage than “n”; DS5 with seed “a” has a slightly lower percentage of FP than “b” but both are higher than “n”

As mentioned in **Section 3.2**, DS2 and DS3 could have some correlation, but no experiment showed any obvious correlation.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

In this thesis, we have shown that the selection bias has an impact on the AAOSVM, while class imbalance does not have an impact on the AAOSVM. Now more experiments are needed to determine if that impact is extended to other classifiers and initial parameters.

As explained in **Section 4.2.3**, if support vectors are used as AST seed samples, it will cause more misclassification. There is a class imbalance in the generated data by AST, but it does not affect the classification algorithm.

When the AAOSVM is changing scores during training, its performance decreases, but the decrease in performance in training is not correlated linearly with its bias metrics values.

5.2 Future Work

5.2.1 Poisoning Attack

With a similar framework and the experiments as this thesis, one could test the effects of selection bias on the AAOSVM coming from a poisoning attack.

5.2.2 Proper Metrics

The metrics used in this thesis were adapted from works on bias related to social context. For adversarial context, we need to develop new metrics to measure the effects of adversarial bias.

5.2.3 Other Machine Learning Models

This thesis focused on SVMs. We need to see how this would behave in other types of machine learning models that take into account the adversarial context.

5.2.4 Tuning Machine Learning Models Parameters

It would also be interesting to see how every machine learning model would behave if they were to go against an attack while its parameters are tuned to a specific dataset.

5.2.5 Time to Train

As said in **Section 4.2**, we found an interesting dependence that is out of this scope. We propose an investigation of this dependence for future work.

5.2.6 Improving the AAOSVM Scoring

We suggest that anyone who is willing to improve on scoring method to try a different set of update functions and conditions to apply the update functions.

BIBLIOGRAPHY

- [1] V. G. Cerf, “Keeping the internet open,” *Communications of the ACM*, vol. 59, no. 9, pp. 7–7, Aug. 2016, doi: 10.1145/2980762.
- [2] L. Kleinman, “Dynamic Workforce Risk And The Remote Worker,” *Forbes Technology Council*.
<https://www.forbes.com/sites/forbestechcouncil/2020/04/27/dynamic-workforce-risk-and-the-remote-worker/?sh=2ef41def4ee6> (accessed Aug. 19, 2021).
- [3] G. Canova, M. Volkamer, C. Bergmann, and B. Reinheimer, “NoPhish App Evaluation: Lab and Retention Study,” *NDSS workshop on usable security*, Feb. 2015, doi: 10.14722/USEC.2015.23009.
- [4] W. Mazurczyk and L. Cavaglione, “Cyber reconnaissance techniques,” *Communications of the ACM*, vol. 64, no. 3, pp. 86–95, Mar. 2021, doi: 10.1145/3418293.
- [5] T. Benzel, “Cybersecurity research for the future,” *Communications of the ACM*, vol. 64, no. 1, pp. 26–28, Jan. 2021, doi: 10.1145/3436241.
- [6] F. Salahdine and N. Kaabouch, “Social Engineering Attacks: A Survey,” *Future Internet*, vol. 11, no. 4, p. 89, Apr. 2019, doi: 10.3390/fi11040089.
- [7] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, “Advanced social engineering attacks,” *Journal of Information Security and Applications*, vol. 22, pp. 113–122, Jun. 2015, doi: 10.1016/j.jisa.2014.09.005.
- [8] S. Lohani, “Social Engineering: Hacking into Humans,” *International Journal of Advanced Studies of Scientific Research*, vol. 4, no. 1, p. 10, 2019.
- [9] H. Shirazi, B. Bezawada, and I. Ray, “‘Kn0w Thy Doma1n Name’: Unbiased Phishing Detection Using Domain Name Based Features,” in *Proceedings of the 23rd ACM on Symposium on Access Control Models and Technologies*, Jun. 2018, pp. 69–75. doi: 10.1145/3205977.3205992.
- [10] APWG Phishing Activity Trends Report, “Phishing Activity Trends Report, 1st-3rd Quarters 2015,” APWG, 2015.

- [11] R. E. Indrajit, “Social Engineering Framework: Understanding the Deception Approach to Human Element of Security,” *International Journal of Computer Science Issues (IJCSI)*, vol. 14, no. 2, pp. 8–16, Mar. 2017, doi: 10.20943/01201702.816.
- [12] H. Shirazi, C. Anderson, B. Bezawada, I. Ray, and C. Anderson, “Adversarial Sampling Attacks Against Phishing Detection,” pp. 83–101, 2019, doi: 10.1007/978-3-030-22479-0_5.
- [13] K. Huang, M. Siegel, and S. Madnick, “Systematically Understanding the Cyber Attack Business,” *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–36, Sep. 2018, doi: 10.1145/3199674.
- [14] B. Biggio and F. Roli, “Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Oct. 2018, pp. 2154–2156. doi: 10.1145/3243734.3264418.
- [15] Q. Chen, S. R. Islam, H. Haswell, and R. A. Bridges, “Automated Ransomware Behavior Analysis: Pattern Extraction and Early Detection,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11933 LNCS, pp. 199–214, Aug. 2019, doi: 10.1007/978-3-030-34637-9_15.
- [16] H. Zhang, X. Xiao, F. Mercaldo, S. Ni, F. Martinelli, and A. K. Sangaiah, “Classification of ransomware families with machine learning based on N-gram of opcodes,” *Future Generation Computer Systems*, vol. 90, pp. 211–221, Jan. 2019, doi: 10.1016/J.FUTURE.2018.07.052.
- [17] D. Sculley and G. M. Wachman, “Relaxed online SVMs for spam filtering,” in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '07*, 2007, pp. 415–422. doi: 10.1145/1277741.1277813.
- [18] B. Schölkopf and A. J. Smola, *Learning with kernels support vector machines, regularization, optimization, and beyond*. 2002.
- [19] N. Figueroa, G. L’Huillier, and R. Weber, “Adversarial classification using signaling games with an application to phishing detection,” *Data Mining and Knowledge Discovery*, vol. 31, no. 1, pp. 92–133, 2017, doi: 10.1007/s10618-016-0459-9.
- [20] F. Stajano and P. Wilson, “Understanding scam victims: seven principles for systems security,” *Communications of the ACM*, vol. 54, no. 3, pp. 70–75, Mar. 2011, doi: 10.1145/1897852.1897872.

- [21] P. Kumaraguru, S. Sheng, A. Acquisti, L. F. Cranor, and J. Hong, “Teaching Johnny not to fall for phish,” *ACM Transactions on Internet Technology*, vol. 10, no. 2, pp. 1–31, May 2010, doi: 10.1145/1754393.1754396.
- [22] P. Carpenter and J. Huisman, “2020 Phishing Attack Landscape and Industry Benchmarking,” 2020.
- [23] K. L. Chiew, E. H. Chang, C. L. Tan, J. Abdullah, and K. S. C. Yong, “Building Standard Offline Anti-phishing Dataset for Benchmarking,” *International Journal of Engineering & Technology*, vol. 7, no. 4.31, pp. 7–14, 2018, doi: 10.14419/ijet.v7i4.31.23333.
- [24] Y. Zhou, M. Kantarcioglu, and B. Xi, “A survey of game theoretic approach for adversarial machine learning,” *WIREs Data Mining and Knowledge Discovery*, vol. 9, no. 3, May 2019, doi: 10.1002/widm.1259.
- [25] H. F. Yang, K. Lin, and C. S. Chen, “Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 437–451, Feb. 2018, doi: 10.1109/TPAMI.2017.2666812.
- [26] D. H. Wolpert, “The Lack of A Priori Distinctions Between Learning Algorithms,” *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, Oct. 1996, doi: 10.1162/neco.1996.8.7.1341.
- [27] S. Geman, E. Bienenstock, and R. Doursat, “Neural Networks and the Bias/Variance Dilemma,” *Neural Computation*, vol. 4, no. 1, pp. 1–58, Jan. 1992, doi: 10.1162/neco.1992.4.1.1.
- [28] S. Singh, “Understanding the Bias-Variance Tradeoff,” *Towards Data Science*, 2018. <https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229> (accessed Mar. 17, 2022).
- [29] H. Shirazi, B. Bezawada, I. Ray, and C. Anderson, “Directed adversarial sampling attacks on phishing detection,” *Journal of Computer Security*, vol. 29, no. 1, pp. 1–23, Feb. 2021, doi: 10.3233/JCS-191411.
- [30] A. Niakanlahiji, B.-T. Chu, and E. Al-Shaer, “PhishMon: A Machine Learning Framework for Detecting Phishing Webpages,” in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Nov. 2018, pp. 220–225. doi: 10.1109/ISI.2018.8587410.
- [31] R. Verma and K. Dyer, “On the Character of Phishing URLs,” in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, Mar. 2015, pp. 111–122. doi: 10.1145/2699026.2699115.

- [32] J. Jiang *et al.*, “A Deep Learning Based Online Malicious URL and DNS Detection Scheme,” 2018, pp. 438–448. doi: 10.1007/978-3-319-78813-5_22.
- [33] M. Pereira, S. Coleman, B. Yu, M. DeCock, and A. Nascimento, “Dictionary Extraction and Detection of Algorithmically Generated Domain Names in Passive DNS Traffic,” 2018, pp. 295–314. doi: 10.1007/978-3-030-00470-5_14.
- [34] J. Charest, “SVM,” 2019. <https://jonchar.net/notebooks/SVM/#Dual-form> (accessed Sep. 06, 2021).
- [35] J. Platt, “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines,” 1998. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/>
- [36] G. L’Huillier, R. Weber, and N. Figueroa, “Online phishing classification using adversarial data mining and signaling games,” in *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics - CSI-KDD ’09*, 2009, pp. 33–42. doi: 10.1145/1599272.1599279.
- [37] B. H. Zhang, B. Lemoine, and M. Mitchell, “Mitigating Unwanted Biases with Adversarial Learning,” *AIES 2018 - Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, vol. 18, pp. 335–340, Dec. 2018, doi: 10.1145/3278721.3278779.
- [38] C. K. I. Williams, “The Effect of Class Imbalance on Precision-Recall Curves,” *Neural Computation*, vol. 33, no. 4, pp. 853–857, Apr. 2021, doi: 10.1162/neco_a_01362.
- [39] G. Vrbančič, “Phishing Websites Dataset.” Mendeley Data, V1, 2020. doi: 10.17632/72ptz43s9v.1.
- [40] R. M. Mohammad, F. Thabtah, and L. McCluskey, “An Assessment of Features Related to Phishing Websites using an Automated Technique,” in *2012 International Conference for Internet Technology and Secured Transactions*, 2012, pp. 492–497.
- [41] R. M. A. Mohammad, L. McCluskey, and F. Thabtah, “Phishing Websites Data Set,” *UCI Machine Learning Repository*. University of California, School of Information and Computer Sciences, Irvine, CA, 2015. Accessed: Jul. 18, 2021. [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Phishing+Websites>
- [42] N. Abdelhamid, A. Ayes, and F. Thabtah, “Phishing detection based Associative Classification data mining,” *Expert Systems with Applications*, vol. 41, no. 13, pp. 5948–5959, Oct. 2014, doi: 10.1016/j.eswa.2014.03.019.

- [43] N. Abdelhamid, "Website Phishing Data Set," *UCI Machine Learning Repository*. University of California, School of Information and Computer Sciences, Irvine, CA, 2016. Accessed: Jul. 18, 2021. [Online]. Available: <https://archive.ics.uci.edu/ml/datasets/Website+Phishing>
- [44] C. L. Tan, "Phishing Dataset for Machine Learning: Feature Evaluation." Mendeley Data, V1, 2018. doi: 10.17632/h3cgnj8hft.1.
- [45] C. L. Tan, K. L. Chiew, N. Musa, and D. H. A. Ibrahim, "Identifying the Most Effective Feature Category in Machine Learning-based Phishing Website Detection," *International Journal of Engineering & Technology*, vol. 7, no. 4.31, pp. 1–6, 2018, doi: 10.14419/ijet.v7i4.31.23331.
- [46] K. L. Chiew, C. L. Tan, K. Wong, K. S. C. Yong, and W. K. Tiong, "A new hybrid ensemble feature selection framework for machine learning-based phishing detection system," *Information Sciences*, vol. 484, pp. 153–166, May 2019, doi: 10.1016/j.ins.2019.01.064.
- [47] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, Dec. 2018, doi: 10.1016/j.patcog.2018.07.023.
- [48] M. Jakobsson, "The Rising Threat of Launchpad Attacks," *IEEE Security and Privacy*, vol. 17, no. 5, pp. 68–72, Sep. 2019, doi: 10.1109/MSEC.2019.2922865.
- [49] R. S. Gibbons, "Game Theory for Applied Economists," *Game Theory for Applied Economists*, Sep. 2019, doi: 10.1515/9781400835881.