Effects of Selection Bias on Online Adversarial Aware SVM When Facing an Evasion Attack

Victor Barboza Morais
Program of Computer Science
Louisiana Tech University
Ruston, LA, USA
victorbarbozamo@gmail.com

Pradeep Chowriappa
Program of Computer Science
Louisiana Tech University
Ruston, LA, USA
pradeep@latech.edu

***Abstract*—In evasion attacks, the success of machine learning (ML) depends on its ability to detect an attack in an adversarial setting. It is important that these ML models are trained regularly to keep abreast with evolving evasion attacks. We present an investigation of the effects of selection bias and class imbalance in training an Adversary-Aware Online SVM (AAOSVM). We show that incorrect samples can compromise the model's ability to detect attacks. Comparison results showed better True Positive Rates (TPR) performance than the Online Support Vector Machine (OSVM), and that AAOSVM was sensitive to selection bias but not to class imbalance.**

***Index Terms*—Adversary-Aware, Machine Learning, Support Vector Machine, Bias, Phishing**

I. Introduction

A successful phishing attack leads to significant losses as the leak of sensitive information can have long-term ramifications [1]. Due to this, machine learning (ML) models and frameworks have been used to varied degrees of success [2], [3]. Many related works use ML algorithms in conjunction with data filtering techniques to create adaptive ML algorithms that boost algorithm performance [4]. However, there is limited research in this space to determine the effectiveness, evaluation, and impact of selection bias when considering the use of ML for defense against phishing attacks. Adversary-Aware Machine Learning (AAML) models take into consideration an attacker (simulated or not) when doing the training. 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.

As is evident in areas of privacy, security, and ethics [2], the hindrance to the research is the lack of validated real-world data. 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. We would benefit from 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 Adversarial Sampling Techniques (AST) to simulate a phishing attack. The generated data is used to test and analyze the vulnerability of the AAOSVM [3]. AST randomly chooses a data sample from the input dataset and modifies the data sample with the intent of causing an ML to misclassify the sample.

The attacker – a person trying to phish someone through a website – is modeled as the AST. We believe that the attacker's goal 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). We also need to try to compensate for the misclassification in the classifier. For this, we implemented a modified version of the Adversary-Aware Online SVM (AAOSVM) [5]. It can change the sample scores during training and takes into account the adversarial nature of the data. This paper is an investigation into the bias of AST against the AAOSVM in the context of phishing attacks.

II. Background

A. 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 [6]. There are several issues to consider when using supervised learning [7], [8]. The prediction error of a learned classifier is related directly to the sum of the bias and variance of a learning algorithm. When it comes to phishing, many works focus on either defining features that will improve the performance of the ML models or enhancing the existing ML models [3], [9].

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 [1]. 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. An example of focus on a feature that was compromised is [10]. They based their work on the URL and the reasoning that phishing websites' URLs can be identified by a trained person with a certain ease [10]. This is because now URL shortening services are widely available.

Jiang et al. developed a Convolutional Neural Network (CNN) that automatically extracts features from the URL [11]. 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 [11]. Pereira et al. distinguished legitimate from phishing domains with precision and accuracy with the use of graphs [12]. Domain Generation Algorithms (DGA) were 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 [12]. A few more considerations noted by Shirazi et al. [3] 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.

B. Selection Bias in SVM

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 [13], [14]. 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.

We believe that the choice of which instances will be kept away from the training can bias how the model behaves. Take Figure 1 as an example. It assists in describing selection bias in the context of the classification boundary. 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.

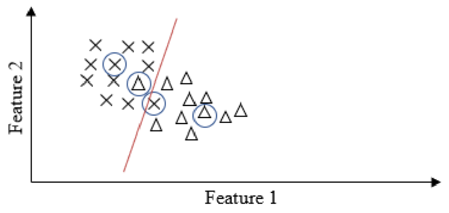

Fig. 1: Desired instance selection.

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. Selection bias within a training set could have downstream effects on machine learning models and it could be related to anomaly detection in a large network. Support vector machines (SVMs) are a type of supervised learning method. Our choice for using an SVM is because we wanted to use the support vectors to induce a selection bias.

III. Methodology

We will try to answer the following three questions in this paper.

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?

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 three types of experiments to test and show how an attack from an AST would influence each type of classifiers as follows:

Experiment 1: Evaluation of an Online SVM classifier.

Experiment 2: Evaluation of the AAOSVM on 200 randomly selected samples and investigate the effects of generated labels by clustering on generated data bias metrics.

Experiment 3: Evaluation of the AAOSVM on 200 samples selected based on support vectors from a previous experiment and investigate the effects of: (a) False Positives (FP) and Class Imbalance (CI) of generated labels by clustering on generated data. (b) Changing the scores of each sample with each training iteration.

A. Proposed Framework

All experiments presented in this work adopt the same framework. Each experiment was repeated ten times and the results reported are an average across ten iterations. To better compare the results, the same 200 samples were removed from all classifiers. Every time an experiment was repeated, 200 random samples were chosen. From the remainder of each dataset, a five-fold cross-validation was performed and the performance metrics were measured. The only classifier that was scored on the adversarial samples was the AAOSVM and it was only scored on bias metrics. We tried to minimize any source of bias as best as we could so we could analyze only the effects of selection bias.

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.

B. Datasets

Five datasets that are publicly available on the Internet (Mendeley data and UCI repositories) are used. 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.

TABLE I: Summary of the objective features in each dataset.

	Data shape (#)		Instances		Features		
Dataset	Size	Features	Legitimate	Phishing	URL based	# binary	# trinary
DS1a (DS1) [15]	58,645	111	27,998	30,647	96	9	0
DS2 [16], [17]	11055	30	6157	4898	8	20	10
DS3 [18], [19]	1353	9	651	702	5	2	7
DS4 [20]–[22]	10000	48	5000	5000	27	23	6
DS1b (DS5) [15]	88,647	111	58,000	30,647	96	9	0

C. Adversarial Sampling Technique (AST)

For the AST, we used the algorithm from Shirazi et al. [3] with the difference that it takes the whole training dataset to generate new samples. 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.

For example, a 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, 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 a 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. Because of the nature of the features in some datasets such as DS1, the feature selection could not be done up to four features as it was in [3]. For perspective, each instance could - depending on the selected features - be generating more than half a million samples when three features were manipulated.

Shirazi et al. [3] implied that 200 samples and four features would be enough to generate sufficient instances to make most trained models useless. Shirazi et al. [3] 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 one more way of selecting the sample seeds to see how it affects the models' predictions. The other way of selecting the sample seeds is by using the support vectors from the previous experiment with the AAOSVM. This resulted in lists that had various lengths. We kept it standardized at 200 samples 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.

To measure the skew (imbalance) of class labels after the AST we used Class Imbalance (CI) as shown in Eq. (1).

$$CI = \frac{n_p - n_l}{n_p + n_l} \tag{1}$$

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. Performance metrics are employed to determine the classifier's performance. The classifiers were measured and compared performances in Accuracy (ACC), True Positive Rate (TPR), and F1-score (F1).

D. Adversarial Classification

In this work, we base our implementation of the SVM algorithm on Sequential Minimal Optimization (SMO) inspired by Charest implementation [23], [24]. From that implementation, we created our version of an AAOSVM based on [5]. We thought it could be improved using a strategy of changing scores. We assume that the classifier's actions did not affect the behavior of the adversary. We describe the AAOSVM using the following definitions.

Definition 1. The training criteria: trains the classifier whenever samples that are considered poorly classified, as expressed in Eq. (2).

$$y_i \times (\vec{w}_{i-1} \cdot \vec{x}_i + b_{i-1}) < \varepsilon \tag{2}$$

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

The variable ε has the value of 0.6 which was the adopted value on [25]. This threshold sets the scaling of $\vec{w}$ and could have been any positive number [26].

Definition 2. 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. The intuition is that there are malicious fraudulent (true phishing), legitimate fraudulent (fake phishing for training purposes), and legitimate non-fraudulent (common) websites.

Definition 3. Probability 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 4. Probability p(M, z): this function computes the 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 5. Transform probability $\phi(M|z)$: this function computes the probability of instance $\vec{x}_j$ to be malicious, given z, and is computed as Eq. (3) [27]:

$$\phi(M|z) = \frac{p(M,z)}{p(z)} \tag{3}$$

Definition 6. Belief $\mu((y,z)|\vec{x}_j)$: is the consistent belief represented in Eq. (4).

$$\mu((y,z)|\vec{x}_j) = \begin{cases} \frac{p(z_j)\phi(M|z_j)}{P(R)+p(M)\phi(M|z)}, & \text{if } y = M \\ \frac{p(z_j)}{P(R)+p(M)\phi(M|z)}, & \text{if } y = R \\ 0, & \text{if } y = R \text{ and } z_j \cong z \end{cases} \tag{4}$$

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

Definition 8. Helper function $\psi(\vec{x}_j)$: the function holds the prior knowledge that is based on probabilities and scores. It is defined in Eq. (5). It is updated every fixed number of samples, as stated in [5]:

$$\psi(\vec{x}_j) = \frac{\sum_{z \in Z} \mu((M,z)|\vec{x}_j) \cdot (\epsilon_M + \gamma_M)}{\sum_{z \in Z} \mu((R,z)|\vec{x}_j) \cdot (\epsilon_R + \gamma_R)} \tag{5}$$

Definition 9. Knowledge function $\Psi(\vec{x}_j)$: the function that is defined in Eq. (6). Adds prior knowledge to the training criteria (Eq. (2)):

$$\Psi(\vec{x}_j) = \frac{1 + \psi(\vec{x}_j)}{w^T e + 2b} \tag{6}$$

Now that $\Psi(\vec{x}_j)$ is defined, it is incorporated to **Definition 1**. The criteria that decides if the SVM needs to be trained again is based on the intuition from Eq. (2):

$$y\hat{y}\Psi(\vec{x}) < v \tag{7}$$

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 are to misclassify the sample or to have $\Psi(\vec{x}) < v$ (or a poorly classified sample).

Definition 10. Update parameter (u): a parameter that will tell the AAOSVM if it should change the scores or not.

Definition 11. Difference of errors (Δ_{error}): when training, if the training criteria is true, the AAOSVM will save the errors prior to training and compare them with the new errors. This comparison is used as part of the conditions to decide which score should be updated, if $u == True$.

From Eq. (5) and Eq. (4), 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.

The AAOSVM was implemented with a sliding window 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 w and b).

To update the utility of a malicious sample, we decreased each sample's respective utility by itself, scaled by tanh(Δ_{error}), as shown in Eq. (8), if the error decreased as well. The intuition is that it will increase the utility of the malicious sample as the error goes down:

$$\epsilon_M - = \epsilon_M \times \text{tanh}(\Delta_{error}) \tag{8}$$

The same happens for a regular sample. In a similar way, we update the costs as shown in Eq. (9). Now, the costs go up as the error goes up:

$$\gamma + = \gamma \times \text{tanh}(\Delta_{error}) \tag{9}$$

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 goes to a minimum.

IV. Results and Discussion

Experiments one and two aimed at creating a baseline of OSVM and AAOSVM. As shown in Figure 2, we see how an OSVM performs on a moving window and establish a baseline for OSVMs (Figure 2). We observe that OSVM performed poorly on DS1, DS4, and DS5 based on TPR. It indicates that datasets that have many features that are numerical will be problematic for SVM.

TABLE II: Performance metrics of AAOSVM changing scores, running one time 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 2 and Table II that changing the scores had a great impact on the AAOSVM; ACC dropped significantly on all datasets. In addition, it was even more critical on DS5, with TPR and F1 near zero, which means that it predicted almost every instance as being legitimate.

Experiment three was done with just one run and on holdout instead of five-fold cross-validation. In experiment three, we

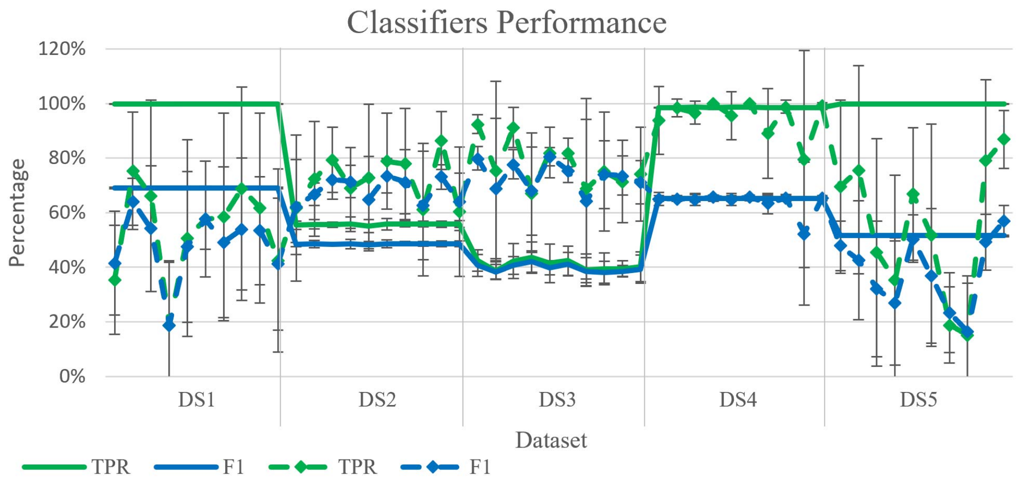

Fig. 2: Performance metrics of OSVM (solid line) and AAOSVM (dashed line with circle).

evaluate the AAOSVM under an AST attack. Each time the AST used a random combination of features, up to two features, to expand the reserved 200 samples manyfold.

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 manipulated instances.

The costs on DS4 and DS5 went to zero with less than 120 samples. As shown on Figure 3, the utility of regular samples on DS4 went to zero with less than 160 samples, which means that for DS4 the scoring had no effect on $\Psi(\vec{x}_i)$ after only around 1% of the samples. Both the utilities on DS5 and the utility of malicious samples on DS4 have stabilized at some number other than zero, which was the desired behavior. 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. We thought that using tanh as the update function, the scores would go up and down, and either settle at zero or at around some value, as the error fluctuates and gradually goes to a minimum.

We did not foresee the number of samples that were 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 to not have $\psi(\vec{x}_i)$ in it and it no longer depends on the sample nor the scores.

TABLE III: Average percentage of FP using different seeds on the AST for different numbers of manipulated features.

	Average of FP (%)	
# of features	support vector	random
0	40.90 ± 35.30	20.40 ± 26.77
1	42.10 ± 35.48	23.60 ± 26.06
2	49.60 ± 38.54	38.60 ± 37.75

TABLE IV: Comparative of Class Imbalance and Percentage of FP on each dataset for each type of sample seed.

Average of Class Imbalance vs Average of FP (%)		
	Support vectors as seed	
Dataset	Class Imbalance	False Positives
DS1	4.48 ± 0.19	8.00 ± 0.00
DS2	-11.17 ± 0.96	62.33 ± 3.70
DS3	3.88 ± 6.12	53.67 ± 15.55
DS4	1.39 ± 20.40	0.00 ± 0.00
DS5	-30.86 ± 0.12	97.00 ± 0.00
	Random instances as seed	
Dataset	Class Imbalance	False Positives
DS1	4.47 ± 0.19	73.00 ± 0.00
DS2	-11.20 ± 0.93	18.33 ± 2.87
DS3	4.04 ± 6.19	41.83 ± 36.92
DS4	1.59 ± 20.38	0.00 ± 0.00
DS5	-30.85 ± 0.12	4.50 ± 0.00

V. Conclusions

We had three research questions. The first one was confirmed, as shown in Table III. 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. The second one was dismissed as there is no correlation between the class imbalance and the percentage of false positives as shown in Table IV. 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 Table IV.

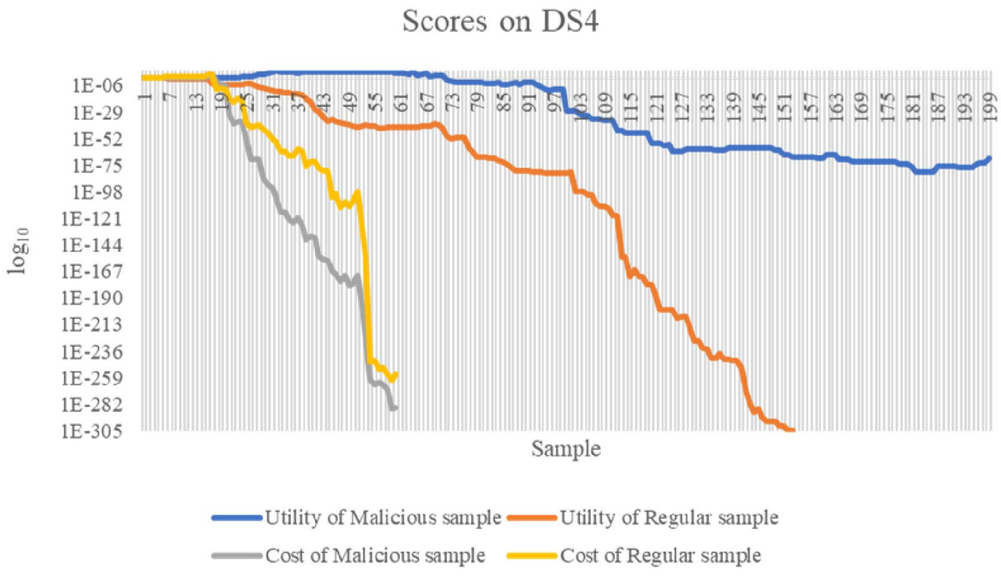

Fig. 3: Score metrics on DS4, for the first 200 samples.

As for the last question: on 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 a terrible performance during training and was facing generated samples designed to fool the classifier, the average percentage of false positives did not go over 50%, as shown on Table III. 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 Table III, the change in FP occurs in a similar way but with different proportions, except for DS3, in Table IV.

In this work, we have shown that 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. 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.

References

[1] H. Shirazi, B. Bezawada, and I. Ray, ""Kn0w Thy Doma1n Name": Unbiased Phishing Detection Using Domain Name Based Features," in *Proceedings of the 23nd ACM on Symposium on Access Control Models and Technologies*. New York, NY, USA: ACM, jun 2018, pp. 69–75. [Online]. Available: https://dl.acm.org/doi/10.1145/3205977.3205992

[2] T. Benzel, "Cybersecurity research for the future," *Communications of the ACM*, vol. 64, no. 1, pp. 26–28, jan 2021. [Online]. Available: https://dl.acm.org/doi/10.1145/3436241

[3] H. Shirazi, C. Anderson, B. Bezawada, I. Ray, and C. Anderson, "Adversarial Sampling Attacks Against Phishing Detection," pp. 83–101, 2019. [Online]. Available: https://www.researchgate.net/publication/334213370

[4] A. Bergholz, J. De Beer, S. Glahn, M.-F. Moens, G. Paaß, S. Strobel, F. Iais, S. A. Germany, K. U. Leuven, and H. Belgium, "New Filtering Approaches for Phishing Email," Tech. Rep.

[5] 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.

[6] 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. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/widm.1259

[7] D. H. Wolpert, "The Lack of A Priori Distinctions Between Learning Algorithms," *Neural Computation*, vol. 8, no. 7, pp. 1341–1390, oct 1996. [Online]. Available: https://direct.mit.edu/neco/article/8/7/1341-1390/6016

[8] 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. [Online]. Available: https://direct.mit.edu/neco/article/4/1/1-58/5624

[9] H. Shirazi, B. Bezawada, I. Ray, and C. Anderson, "Directed adversarial sampling attacks on phishing detection," *Journal of Computer Security*, no. 1, pp. 1–23, feb.

[10] R. Verma and K. Dyer, "On the Character of Phishing URLs: Accurate and Robust Statistical Learning Classifiers," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*. San Antonio, TX, USA: ACM, mar 2015, pp. 111–122. [Online]. Available: https://dl.acm.org/doi/10.1145/2699026.2699115

[11] J. Jiang, J. Chen, K.-K. R. Choo, C. Liu, K. Liu, M. Yu, and Y. Wang, "A Deep Learning Based Online Malicious URL and DNS Detection Scheme," 2018, pp. 438–448. [Online]. Available: http://link.springer.com/10.1007/978-3-319-78813-5_22

[12] 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. [Online]. Available: http://link.springer.com/10.1007/978-3-030-00470-5_14

[13] F. Vella, "Estimating Models with Sample Selection Bias: A Survey," *The Journal of Human Resources*, vol. 33, no. 1, p. 127, 1998. [Online]. Available: https://about.jstor.org/terms https://www.jstor.org/stable/146317?origin=crossref

[14] J. J. Heckman, "Selection Bias and Self-selection," in *Econometrics*.

London: Palgrave Macmillan UK, 1990, pp. 201–224. [Online]. Available: http://link.springer.com/10.1007/978-1-349-20570-7_29

[15] G. Vrbančič, "Phishing Websites Dataset," 2020.

[16] 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.

[17] R. M. A. Mohammad, L. McCluskey, and F. Thabtah, "Phishing Websites Data Set," Irvine, CA, 2015. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Phishing+Websites

[18] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based Associative Classification data mining," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5948–5959, oct 2014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0957417414001481

[19] N. Abdelhamid, "Website Phishing Data Set," Irvine, CA, 2016. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Website+Phishing

[20] 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.

[21] C. L. Tan, "Phishing Dataset for Machine Learning: Feature Evaluation," 2018.

[22] K. L. Chiew, C. L. Tan, K. Wong, K. S. 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. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0020025519300763

[23] J. Charest, "SVM," 2019. [Online]. Available: https://jonchar.net/notebooks/SVM/Dual-form

[24] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Microsoft, Tech. Rep., 1998. [Online]. Available: https://www.microsoft.com/en-us/research/publication/sequential-minimal-optimization-a-fast-algorithm-for-training-support-vector-machines/

[25] 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*. New York, New York, USA: ACM Press, 2009, pp. 33–42. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1599272.1599279

[26] B. Schölkopf and A. J. Smola, *Learning with kernels support vector machines, regularization, optimization, and beyond*, 2002.

[27] R. S. Gibbons, *Game Theory for Applied Economists*. Princeton University Press, jul 1992. [Online]. Available: https://www.degruyter.com/document/doi/10.1515/9781400835881/html