Exploring Feature Normalization and Temporal Information for Machine Learning Based Insider Threat Detection

Pedro Ferreira*, Duc C. Le† and Nur Zincir-Heywood‡
*†‡*Faculty of Computer Science, Dalhousie University*, Halifax, Nova Scotia, Canada
**Department of Electrical Engineering, University of Brasília*, Brasília, Brazil
Email: *pedroferreira@ieee.org, †lcd@dal.ca, ‡zincir@cs.dal.ca

***Abstract*—Insider threat is one of the most damaging cybersecurity attacks to companies and organizations. In this paper, we explore different techniques to leverage spatial and temporal characteristics of user behaviours for insider threat detection. In particular, feature normalization (scaling) techniques and a scheme for representing explicit temporal information are explored to improve the performance of the machine learning based insider threat detection. The results show that these data characteristics have different effects on different classifiers, where Standard Scaler with Random Forest classifier produces the best performance.**

***Index Terms*—Insider Threat Detection, Machine Learning, Spatial and Temporal Data Characteristics**

I. Introduction

Insider threat is a growing concern in the area of cybersecurity. In the last year, 53% of the companies reported to have faced at least one insider threat attack. Moreover, 90% of the organizations consider themselves vulnerable to insider threats and 66% of them believe that they are most likely to face an insider threat attack than an external one [1].

However, even though most of the organizations are aware of the risks of insider threat attacks, the insider threat detection remains challenging due to its nature. Insider threat attacks have a wide range, varying from Intellectual Property Theft and Data Leak to IT Sabotage [2]. They can be performed both intentionally and unintentionally. They account for just a small fraction of user's activities, being usually blended with non-malicious actions. Thus, they often require the employment of sophisticated control and intensive monitoring frameworks [1]. As the insider threat detection task can be seen through the prism of a classification and/or clustering problem, the application of classical machine learning becomes a suitable alternative [3]–[5]. This paper will focus on exploring spatial and temporal information present on user's activity logs to improve the results of the application of classical machine learning in insider threat detection. To this end, different methods of feature normalization as well as different time granularities will be exploited via sliding window approach. Additionally, feature concatenation will be studied over a longer historical time window as to provide a context for the learning algorithms and benefit from the temporal correlations of the actions. A parameter sensitivity analysis will be conducted and the results will be compared in terms of performance and computational cost.

In the following, section II reviews the literature on insider threat detection. Section III discusses the methodology employed in this paper. Section IV, presents the evaluations performed and the results obtained. Finally, in section V, conclusions are drawn and future works are discussed.

II. Related Work

Insider threat detection attracts attention not only from cybersecurity research community but also organizations and government agencies. General guidelines to insider threat detection and prevention have been released by CERT insider threat centre and U.S. National insider threat task force [2], [6]. Recent surveys by Liu et al. [5] and Homoliak et al. [7] summarize the literature in insider threat research. Particularly, in [5], Liu et al. reviewed the research in detection of malicious insider and relevant threats, such as malware and intrusions, which can lead to different types of insider problems. Furthermore, a structural taxonomy for the insider threat related research is presented in [7].

Due to the advancement of machine learning (ML) and data mining techniques, they have been widely adopted for many intrusion detection tasks [3], [4]. For insider threat detection, different ML methods have been employed [5], [7], [8], including but not limited to graph-based [9], [10] and mixture of models [11]–[13], stream online learning techniques [14], [15], and supervised learning algorithms [16], [17].

An associated aspect to any application of ML methods is data preprocessing [4], [18]. In [19], Habeeb et al. reviewed the data pre-processing techniques in application of big data in anomaly detection. In [20], Ramírez-Gallego et al. surveyed the data pre-processing techniques in stream data mining – a method that has been applied extensively in insider threat detection [10], [14], [15]. A general review of data preprocessing for anomaly based network intrusion detection is presented by Davis et al. in [21]. Among the preprocessing techniques, a particular example is feature normalization, which has been examined in general intrusion detection by Wang et al. in [22].

On the other hand, cybersecurity in general and insider threat detection in particular are highly related to the human factors. Hence, several works have explored different ways

978-3-903176-24-9

of employing temporal information. Most of the time, this is a representation of user behaviours (actions) over time, for improving the performance of the ML based detectors. Examples of techniques relying on temporal information include Information tap [23], [24], Moving average [14], [25], and Markovian models [26], [27].

In this research, we examine the effect (if any) of different feature normalization techniques. Moreover, we introduce explicit temporal information on ML methods for insider threat detection. To this end, different sizes of sliding windows over time are evaluated to show how much temporal information is best suited for improving the system performance.

III. Methodology

This section provides an overview of the approach followed in this research in terms of the dataset, learning algorithms, normalization methods and the sliding window approach employed.

A. Dataset

Although insider threats constitute one of the main cybersecurity issues for companies and government agencies, publicly available real world data of insider threat events is very scarce [28]. This is due both to the confidentiality of the records of users' activities in the workplace and to the high-cost associated with the manual detection of a set of actions as malicious. Since malicious insider actions are a very small portion of users' activities, a human expert looking for insider threat actions would spend most of his/her time checking regular non-malicious users' activities. Furthermore, as many of the malicious actions performed by insiders rely on accessing data [5], [28], the insider threat detection is also heavily influenced by the organizational structure and the data access rights and roles. Thus, some solutions for detecting insider threats are specific to a particular organization.

As an option to overcome the scarcity of real world insider threat data, CERT [28] has simulated such environments and made the generated data publicly available for the research community. In this paper, the release 5.2 of CERT Insider Threat Detection Dataset (CERT 5.2 dataset), simulating the activities of an organization with 2000 employees over a 74-week period, is utilized [28], [29]. The CERT 5.2 dataset presents a detailed log for emails, web history, file and device access, logins and logoffs and the time of each of these events. Also, it presents four different types of malicious insider actions, ranging from data exfiltration to IT sabotage, and different types of intellectual property thefts.

In order to obtain representative features, the dataset is preprocessed and the features are chosen as summary statistics of the logged actions. These consist of: (i) frequency statistics – the ones that record the number of times that an action was performed over a given time period; and (ii) descriptive statistics – the mean and the standard deviation of the distribution of a target variable over a given time period. Additionally, users' profile information is encoded categorically in [17].

For *a priori* analysis, three different time durations are chosen, namely session, daily and weekly, and for each duration, a particular dataset is built by aggregating each user's actions over a given time period. The week-based dataset presents 1092 features and the malicious insider events account for 0.38% of the actions, while in the daily dataset, there are 824 features and 0.19% of the actions are performed by malicious insiders. On the other hand, for the session-based dataset, there are 221 features and malicious actions accounting for 0.18% of the events in the dataset. In order to explore the temporal correlations present in the features, a sliding window analysis is performed. The process that is employed to build the sliding window is discussed in depth in subsection III-D.

B. Machine Learning Algorithms

For the detection of insider threats, three different classical supervised ML algorithms, namely Logistic Regression, Random Forest and Multilayer Perceptron, are employed. An overview of these algorithms are given below. A more detailed description for them can be found in [18].

1) Logistic Regression: A Logistic Regression [30] can be seen as a linear classifier for a binary dependent variable. It consists of finding the vector of weights, w, that minimizes, over all samples, the sum of squared error between the prediction function $\sigma(w^T x)$ and the true value of the label for each sample x. However, Logistic Regression is a very simple model, always producing a linear decision boundary. Since its outputs can be easily described in terms of its inputs, Logistic Regression provides us a highly interpretable model, therefore making it suitable for cybersecurity purposes.

2) Random Forest: A Random Forest [31] is a classifier composed by an ensemble of decision trees trained with different training subsets obtained through bagging from the main training set. The output is a majority vote of the decisions of individual decision trees. Moreover, a Decision Tree classifier can be interpreted as a non-linear classifier that splits the input space in different regions aiming to maximize the information gain in each split. Decision Trees can draw complex decision boundaries and also provide very interpretable results. Thus, they are also suitable for cybersecurity purposes.

3) Multilayer Perceptron: A Multilayer Perceptron [18] is a neural network consisting of at least three layers - input, hidden and output - and a non-linear activation function. It learns from data through backpropagation. This is an efficient algorithm that, given an error metric to minimize, calculates how much to change each layer's weights exploiting the mathematical properties of the chain rule. Since the Multilayer Perceptron can be understood as a universal function approximator, it can discriminate the data even in scenarios with extremely complex decision boundaries. This makes it useful, for benchmarking purposes, for the insider threat detection problem, since the real boundary of the problem for the features chosen is unknown. However, Multilayer Perceptron is non-transparent, where it is not possible to express the output directly as a simple function of the inputs.

C. Data Normalization

For the application of ML in multidimensional datasets, feature normalization is an important step in data preprocessing, when the aim is to employ any algorithm based in l-norm error minimization. Therefore, as two of the chosen ML algorithms are Logistic Regression and Multilayer Perpectron, feature normalization is necessary for our particular problem. Since the true distribution of each feature is not known beforehand, different feature normalization methods are applied to the data before the ML training step. The three methods, namely Standard Scaling, Maximum Absolute Value Scaling and Quantile Transform, used in this research are described below. The reason we chose these three methods is based on the previous work performed in [22].

1) Standard Scaler: Standard Scaler (SS) is a featuring scaling method that normalize each feature removing its mean and scaling its variance to one. As the determination of the normalized value just depends of the mean and the variance, it presents advantages such as being linear, reversible, fast and highly scalable. On the other hand, Standard Scaler also has some disadvantages such as it manifests a high sensitivity to outliers and is more suitable for normally distributed data. In addition, standard scaling also can struggle to handle sparse data, since after its application the scaled data produced will be dense. [32], [33]. For each observation X_i from a sample with mean $\bar{X}$ and standard deviation σ, its normalized version $\hat{X}_i$ can be determined from equation 1.

$$\hat{X}_i = \frac{X_i - \bar{X}}{\sigma} \tag{1}$$

2) Maximum Absolute Value Scaler: Maximum Absolute Value Scaler (MA) is a feature scaling method that normalizes each feature by dividing each sample for the maximum absolute value of the feature. MA is a linear, reversible, and scalable method that can also handle sparse data. On the other hand, MA, as Standard Scaler, is still very sensitive to outliers and each feature presents a non-zero mean, potentially making it necessary for the addition of a bias term in the models [32], [33]. For each observation X_i in a sample X with maximum absolute value X_{MA}, each normalized observation $\hat{X}_i$ can be determined from equation 2.

$$\hat{X}_i = \frac{X_i}{X_{MA}} \tag{2}$$

3) Quantile Transform: Quantile Transform (QT) is a feature scaling method that utilizes quantile information to scale each feature through the application of an inverse cumulative distribution function. As a quantile based transform, it is robust to marginal outliers. Also, it is a non-linear and weakly-reversible transformation. However, a test set value which is out of the bounds of the cumulative distribution function of the training set cannot be perfectly recovered. This may distort the linear correlations shown between different samples of the same feature. Over and above that, Quantile Transform may have performance issues on account that it both struggles to handle sparse data, producing a non-sparse output, and can exhibit scalability issues once that finding a cumulative distribution function demands sorting all the values of each feature [32], [33]. The Quantile Transform can be determined from taking the following steps:

- Sort the samples in order to estimate its cumulative distribution function.
- For a sample in the input range, its Quantile Transform coincides with its cumulative distribution function value.
- For samples outside of the input range, its Quantile transform is given by the bounds of cumulative distribution function.

D. Sliding Window

If we analyze the insider threat detection problem from the perspective of a time series problem, it can be reduced to detecting whether a user performed a malicious action during a time period given the time series of each feature. Since it is reasonable to suppose that, for a given company, the actions performed are quite consistent, we should expect a high auto-correlation in the features [23], [24], [34].

As an effort to assist the learning algorithm in capturing this auto-correlation, we employ a sliding window approach where each observation provided to the algorithm is drawn from a sliding window. In this case, the sliding window contains features not just for the current observation but also for a fixed size of past observations. The sliding window approach ensures a smooth transition between the samples. We can also highlight that the decision of using all features from current and past observations is highly desirable since it may allow a near to optimal automatic weight assignment for past and current versions of each feature and provide a highly interpretable result.

Sliding Window based data preprocessing is build on the CERT 5.2 weekly dataset. Each sample is composed by the user's profile information and by all the features of the activities of each period in the sliding window. In other words, for a sliding window of size 2, each sample at time t will be composed of the user's profile information, for the features of the activities of week t and week $t-1$. One can easily realize that, in this approach, the number of features grows approximately linearly with the window size. In this case, our goal is to explore whether the information about past behaviour provided to the classifiers aids in detecting malicious actions over time.

For the Sliding Window dataset, the series of actions over a sliding window is labeled as non-malicious, if none of the individual actions are malicious. Equivalently, it is labeled as malicious, if any of the individual actions is malicious. It should be noted here that, due to this AND-construction effect, an increase in the window size also increases the ratio of malicious samples in the population. This reduces the class imbalance.

IV. Evaluations

This section presents the experiments performed and discusses the results obtained.

A. Training Set

Given the privacy limitations, data for building and evaluating insider threat detection methods are extremely rare [7]. In this research, we employ a training set composed of records of 400 users (20% of the total users) from the first 37 weeks of the CERT r5.2 dataset [28]. The aim is to "mimic" a real-world situation, where the labelled data is rare. Hence in this case, the work assumes labelled data from only a small amount of user (20%) and a time period (first half of data duration). Since our main goal is detecting whether a user has performed a malicious action in a certain time period or not, we aggregate all four malicious classes into one single class, thus reducing the class imbalance presented in the dataset. In order to further balance the training dataset in terms of malicious and normal actions, the insider class was over-sampled in the composition of the training set. Therefore, all users that presents any malicious behaviour in the first 37 weeks are included in the training set, making the training set account for 34% of the malicious users in comparison to 18% of the normal users. Table I presents a summary of the training and test datasets employed.

TABLE I
OVERVIEW OF TRAINING AND TEST DATASETS EMPLOYED

Data granularity	**Training**		**Testing**	
	Normal	Malicious	Normal	Malicious
Session	103648 (99.42%)	602 (0.58%)	473159 (99.76%)	1122 (0.24%)
Day	71931 (99.36%)	463 (0.64%)	328082 (99.76%)	775 (0.24%)
Week	14330 (98.94%)	153 (1.06%)	67016 (99.47%)	355 (0.53%)

B. Performance Metrics

As this work deals with strongly skewed data, accuracy is not an appropriate performance metric for the algorithms. In this case, a classifier biased to the normal class would still obtain a very high accuracy although it is unable to detect any insider threat. On the other hand, insider detection rate and insider detection precision appears as natural choices for the metrics to be taken under consideration. The main goal of the detection system is to detect as many malicious actions as possible with the highest precision possible. F1-Score is also a suitable choice since it is the harmonic mean of precision and detection rate (recall). It penalizes biased results in favour of balanced ones, producing low scores for systems with: (i) high detection rate and low precision – which would require a lot of effort from an expert to identify the true insiders from the misclassified normal users; and (ii) systems with low detection rate and high precision – which would again require a lot of effort from the expert to identify the misclassified malicious users in the normal users pool.

Therefore, in this paper, the insider detection rate, the insider detection precision and the F1-Score will be employed as performance metrics. For completeness, normal user detection rate will also be provided. In doing so, our aim is to understand when a given algorithm outperforms another and whether it just trades detection rate for precision or vice versa. The results are presented both for instance-based and user-based detection. A test user is considered an insider if he/she performs at least one malicious action during the testing weeks. The insider detection rate, insider detection precision and F1-Score can be determined from equations 3, 4, 5 respectively, where TP stands for the number of true malicious samples detected, FP is the number of normal samples classified as malicious, and FN is the number of malicious samples classified as normal. In the following, the detection rates are reported both based on instances (IDR) and users (UDR).

$$DR = \frac{TP}{TP + FN}, \quad (3) \qquad Precision = \frac{TP}{TP + FP}, \quad (4)$$

$$F1Score = 2\,\frac{DR \cdot Precision}{DR + Precision}, \quad (5)$$

The computational cost of the algorithms are given by the median of 10-runs execution time. To ensure comparability between the execution times, all of the algorithms are benchmarked on the same computer.

C. Implementation and Hyper-Parameter Tuning

The dataset processing steps are implemented in Python 3.7. The ML algorithms and the feature normalization methods are employed using the Scikit-learn library [35]. In terms of hyper-parameters for the ML algorithms, the Logistic Regression is run with default parameters using *lbfgs* [36] as solver. For the Random Forest classifier, the number of individual decision trees is set to 200. Each individual tree has a number of features available for training up to the basis-2 logarithm of all features. Finally, a Multilayer Perceptron is applied using two hidden layers: the first with 545 neurons and the second with 272 neurons with *Adam* [37] as solver, 10^{-4} as the $l2$ regularization penalty, 10^{-8} as *Adam*'s epsilon and 125 as the maximum number of iterations allowed.

D. Impact of Feature Normalization on Learning Algorithms

Before exploiting the temporal correlations present in the data, we analyzed the impact of feature normalization over the performance of the ML algorithms. This not only enables the decision of which normalization method to use, but also provides information about the spatial arrangement of the data.

Table II shows that, for a Logistic Regression Model, a Quantile Transform of the features seems to perform better than the other two normalization methods. This indicates that, for a linear model, how a user behaves in comparison with other users seems to be more significant than its absolute behaviour. Moreover, we can also see that the performance of Quantile Transform, in relation to the other transforms, quickly improves with a reduction of the time granularity. This is expected from the standpoint of a regression towards the mean. Since a session is the highest granularity, it is expected that some sessions may have extreme values even for a non-malicious user. So, it can be hard to distinguish a malicious user from a normal one just based on one session information. Although, when we aggregate the sessions into daily basis, it is expected that the user's behaviour over a day might be a good signature of his/her long-term behaviour. We can easily extend such an argument for a weekly analysis as well.

TABLE II
INSTANCE BASED RESULTS OF ML ALGORITHMS

ML Algorithm	Scaler	Session					Day					Week				
		Insider IDR	Normal IDR	Precision	F1-Score	CC	Insider IDR	Normal IDR	Precision	F1-Score	CC	Insider IDR	Normal IDR	Precision	F1-Score	CC
Logistic Regression	**SS**	30.66%	99.48%	13.59%	0.1839	23.86s	46.58%	99.03%	11.06%	0.1759	60.85s	58.53%	98.42%	16.74%	0.2592	15.33s
	QT	20.00%	99.60%	11.52%	0.1431	39.99s	48.98%	99.37%	16.49%	0.2438	98.44s	59.09%	99.06%	26.82%	0.3643	24.83s
	MA	22.99%	99.69%	19.83%	0.1960	23.86s	32.10%	99.36%	11.35%	0.1645	60.86s	43.66%	99.02%	21.04%	0.2778	15.55s
Random Forest	**SS**	31.96%	99.98%	84.01%	0.4615	56.30s	40.87%	99.95%	76.65%	0.5191	58.71s	50.00%	99.98%	96.32%	0.6577	9.13s
	QT	31.66%	99.98%	82.23%	0.4539	73.90s	44.32%	99.90%	71.47%	0.5252	80.51s	51.01%	99.98%	96.29%	0.6662	19.92s
	MA	31.89%	99.97%	77.19%	0.4493	55.51s	42.29%	99.95%	71.77%	0.5222	42.17s	48.92%	99.97%	92.42%	0.6392	9.90s
Multilayer Perceptron	**SS**	38.93%	98.66%	6.91%	0.1164	198.57s	54.09%	99.03%	12.50%	0.2008	199.62s	64.22%	98.94%	24.84%	0.3557	34.31s
	QT	36.84%	98.71%	7.07%	0.1162	333.39s	48.69%	99.25%	14.01%	0.2162	425.21s	64.59%	98.82%	24.05%	0.3462	65.47s
	MA	37.19%	98.79%	9.03%	0.1344	547.61s	56.47%	98.81%	12.16%	0.1949	378.58s	62.19%	98.56%	19.85%	0.2972	69.98s

TABLE III
USER BASED RESULTS OF ML ALGORITHMS

ML Algorithm	Scaler	Session					Day					Week				
		Insider UDR	Normal UDR	Precision	F1-Score	CC	Insider UDR	Normal UDR	Precision	F1-Score	CC	Insider UDR	Normal UDR	Precision	F1-Score	CC
Logistic Regression	**SS**	92.15%	91.26%	27.85%	0.4262	23.86s	95.07%	88.52%	22.92%	0.3691	60.85s	93.38%	91.86%	29.12%	0.4434	15.33s
	QT	79.69%	93.15%	29.50%	0.4295	39.99s	95.53%	93.15%	33.32%	0.4938	98.44s	86.30%	94.45%	36.08%	0.5079	24.83s
	MA	84.76%	95.25%	39.85%	0.5387	23.86s	88.15%	94.37%	36.10%	0.5113	60.86s	86.46%	95.90%	43.57%	0.5769	15.55s
Random Forest	**SS**	85.62%	99.32%	81.92%	0.8362	56.30s	84.61%	99.30%	82.88%	0.8325	58.71s	74.30%	99.93%	97.69%	0.8435	9.13s
	QT	85.07%	99.27%	81.72%	0.8297	73.90s	85.84%	99.23%	80.59%	0.8297	80.51s	75.84%	99.90%	96.29%	0.8482	19.92s
	MA	86.30%	99.10%	77.50%	0.8153	55.51s	84.92%	99.13%	79.65%	0.8156	42.17s	73.23%	99.86%	95.13%	0.8268	9.90s
Multilayer Perceptron	**SS**	91.23%	87.49%	21.11%	0.3417	198.57s	96.30%	91.16%	28.08%	0.4345	199.62s	93.69%	94.99%	40.28%	0.5628	34.31s
	QT	90.46%	84.98%	18.25%	0.3021	333.39s	93.84%	89.15%	24.48%	0.3863	425.21s	90.76%	93.15%	32.38%	0.4764	65.47s
	MA	88.92%	90.18%	26.95%	0.4051	547.61s	95.07%	90.75%	27.56%	0.4252	378.58s	93.84%	94.60%	39.25%	0.5503	69.98s

On the other hand, a careful analysis of Multilayer Perceptron results in Tables II and III raises concerns on the feasibility of the algorithm for insider threat detection. Although it has high insider detection rates, it also presents the worst precision rate of all three classifiers. Since the precision is low, there would be a very large number of false positives for each true positive, which alongside with its high computational cost, makes Multilayer Perceptron undesirable. On the other hand, Random Forest-based detection, shown in Tables II and III, reveals the best performance and scalability of the evaluated algorithms. Random Forest also shows the best results in terms of precision, where $>80\%$ precision means that for every 5 alarmed users, at least 4 of them are true insider attackers.

TABLE IV
p-VALUES OF PAIRWISE T-TESTS OF WEEKLY RANDOM FOREST BASED DETECTION METRICS

Feature Scaling	Instance				User			
	SS		QT		SS		QT	
	Insider IDR	Normal IDR	Insider IDR	Normal IDR	Insider UDR	Normal UDR	Insider UDR	Normal UDR
QT	0.1899	0.2265	-	-	0.3988	0.6623	-	-
MA	0.1743	0.4828	0.9699	0.1552	0.7047	0.6208	0.5904	0.9058

Given that it is not l-norm dependent, its performance seems to be independent from the normalization method chosen [38]. In order to confirm this observation, a Pairwise T-Test is performed for each pair of feature normalization method chosen. Since the sliding window approach is employed for weekly data, just the weekly-based results from the Random Forest detector is tested. In the following, we report these T-Test results just for the descriptive series of the aforementioned tests. From the results in Table IV, it is clear that all p-values are greater than the p-values of the confidence intervals (0.1, 0.05 and 0.01). Therefore, we cannot reject the null hypothesis for all of these comparisons. Thus, for weekly data, the choice of feature normalization method does not affect the performance of a Random Forest based insider detection system. Hence, it is convenient to employ Standard Scaler for Random Forest to perform the classification faster. Given the poor performance of Multilayer Perceptron, we only use Logistic Regression and Random Forest for the rest of the experiments performed.

E. Sliding Window Sensitivity Analysis

As discussed in section III-D, a Sliding Window approach is employed to feed the ML algorithms not only with the current user activities, but also with the previous activities. Ideally, it is expected that this past information could be used to help the ML algorithm to build a context for each user's action. This in return may provide the context to better identify whether an action is malicious or not.

In order to investigate whether and how the detection systems can benefit from such an approach, both Random Forest and Logistic Regression classifiers are applied to the sliding week dataset utilizing the tuned hyper-parameters detailed in IV-C. For the Logistic Regression, a Quantile Transform is applied on the features so that it can benefit from the performance boost shown in section IV-D. As discussed earlier, Standard Scaler is employed on the features for Random Forest. The sliding window size is varied and, for each window size, a dataset is assembled according to the discussions in III-D. A training set is formed following the process described in IV-A.

The results of this approach are presented in Figure IV-E and given in detail in Tables V and VI. A careful analysis of the results shows that Random Forest and Logistic Regression have different behaviours when fed with past information.

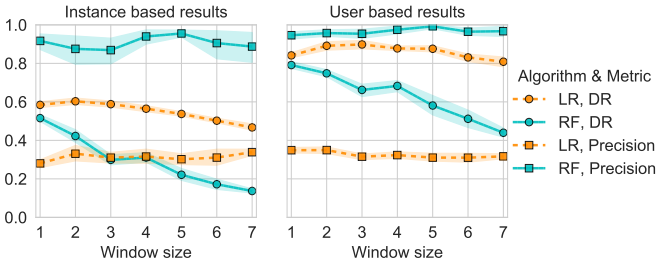

Fig. 1. Insider Threat Detection Performance vs Sliding Window Size

TABLE V
INSTANCE BASED RESULTS OF DIFFERENT WINDOW SIZES

Window Size	Logistic Regression				Random Forest			
	Insider IDR	Normal IDR	Precision	F1-Score	Insider IDR	Normal IDR	Precision	F1-Score
1 Week	58.44%	99.20%	28.03%	0.3775	51.56%	99.97%	91.69%	0.6578
2 Weeks	60.30%	99.15%	33.03%	0.4229	42.27%	99.95%	87.54%	0.5636
3 Weeks	58.82%	99.00%	31.04%	0.4048	29.88%	99.96%	86.90%	0.4406
4 Weeks	56.46%	98.98%	30.61%	0.4018	31.07%	99.98%	93.97%	0.4645
5 Weeks	53.72%	98.93%	30.19%	0.3840	22.12%	99.99%	95.54%	0.3560
6 Weeks	50.23%	98.98%	31.06%	0.3793	17.19%	99.98%	90.56%	0.2851
7 Weeks	46.72%	99.9%	33.86%	0.3914	13.7%	99.98%	88.73%	0.2359

Logistic Regression shows improvement in performance when the information of the current week and the week before (a sliding window of size 2) is used. This improves both its instance-based and user-based precision and insider detection rates. This performance starts to deteriorate for bigger window sizes, reaching even below the baseline performance of a current week only data model (sliding window of size 1). On the other hand, the performance of Random Forest always decreases when it is fed with multiple weeks of sliding window data. However, the precision for Random Forest seems to increase both for user and instance based detection when sliding windows of sizes 3 to 5 are used.

The above results indicate that the concatenation of user data over many weeks does not seem to improve the detection performance of the ML algorithms as much as we expected. This might be because of the features chosen to be highly correlated with the malicious (normal) activity in the first place. In order to test this, we analyze the importance of the features using the Random Forest classifier. Since this classifier has the ability to choose a subset of important features from a given set of features [31]. A sample of the most important features for a Random Forest based Insider Threat Detection system is presented for a sliding window of size 2 in Table VII. These results show that the most important features seem to belong to different periods of the same features. We can assume that these are modeled as independent features by Random Forest. Thus, the current and past values of a given feature are highly correlated. In other words, as we move forward in time, the information gain on the past feature space gets smaller and smaller. Furthermore, this also explains the performance drop faced by Random Forest based detection system as a consequence of the presence of highly correlated features [39], [40]. Thus, the utilization of the concatenation of features via the sliding window technique to produce new features does not seem to be beneficial for the datasets employed in this work.

TABLE VI
USER BASED RESULTS OF DIFFERENT WINDOW SIZES

Window Size	Logistic Regression				Random Forest			
	Insider UDR	Normal UDR	Precision	F1-Score	Insider UDR	Normal UDR	Precision	F1-Score
1 Week	84.13%	95.00%	49.34%	0.4928	79.14%	99.87%	94.62%	0.8611
2 Weeks	89.13%	94.69%	50.20%	0.5014	74.83%	99.90%	95.74%	0.8392
3 Weeks	89.82%	93.77%	46.69%	0.4667	66.21%	99.91%	95.39%	0.7804
4 Weeks	87.75%	94.13%	47.28%	0.4721	68.28%	99.95%	97.49%	0.8017
5 Weeks	87.58%	93.76%	45.84%	0.4570	58.10%	99.99%	99.24%	0.7292
6 Weeks	83.10%	94.08%	45.18%	0.4505	51.21%	99.95%	96.47%	0.6649
7 Weeks	80.86%	94.53%	45.53%	0.4550	43.97%	99.96%	96.72%	0.6035

TABLE VII
6 MOST IMPORTANT FEATURES OF THE RANDOM FOREST INSIDER THREAT DETECTION FOR A SLIDING WINDOW OF SIZE 2

Feature Name	Feature Importance
#acts, supervisor's PC, week t	0.0076
#acts, supervisor's PC, week $t-1$	0.0070
HTTP url depth, weekend, "other" cat., week t	0.0060
HTTP url depth, weekend, "other" cat., week $t-1$	0.0053
#HTTP acts, shared PC, weekend, "other" cat., week t	0.0048
#HTTP acts, shared PC, weekend, "other" cat., week $t-1$	0.0048

V. CONCLUSION AND FUTURE WORKS

In this paper, a thorough exploration of spatial and temporal characteristics of the CERT 5.2 dataset was conducted in order to better understand the data. In doing so, our aim is to enhance the performance of insider threat detection systems based on ML classifiers by providing insights on how to apply feature normalization and sliding window-based temporal information in ML for insider threat detection. Our results confirm that the deployment of Standard Scaler with Random Forest classifier produces a high performance in terms of different metrics and low computational cost in terms of efficiency. Moreover, our analysis shows that Random Forest seems to be robust to different feature normalization techniques. This in return makes it very scalable to different environments. On the other hand, the use of temporal information via sliding window approach seems to affect different classifiers differently. While it is improving Logistic Regression, it does not seem to have the same effect on Random Forest classifier.

As future work, we aim to conduct further investigations in the application of different techniques for combining features. In doing so we aim to explore a more effective translation of the temporal correlations that might be present in the data. Moreover, another possible exploration could be the employment of other ML classifiers such as genetic algorithms that could aggregate observations automatically.

ACKNOWLEDGMENT

This research is supported by Natural Science and Engineering Research Council of Canada (NSERC). Pedro Ferreira is funded by Mitacs Globalink Internship program. Duc C. Le gratefully acknowledges the supports from the Killam trusts, Mitacs, and the province of Nova Scotia. This research is conducted as part of the Dalhousie NIMS Lab at: https://projects.cs.dal.ca/projectx/.

REFERENCES

[1] Holger Schulze, "Insider Threat 2018 Report," CA Technologies, Cybersecurity Insiders, Tech. Rep., 2018.

[2] M. L. Collins, M. C. Theis, R. F. Trzeciak, J. R. Strozer, J. W. Clark, D. L. Costa, T. Cassidy, M. J. Albrethsen, and A. P. Moore, "Common sense guide to mitigating insider threats," The CERT Insider Threat Center, Tech. Rep. CMU/SEI-2015-TR-010, 2016.

[3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.

[4] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*, 1st ed. Boston, MA, USA: Auerbach Publications, 2011.

[5] L. Liu, O. De Vel, Q.-L. Han, J. Zhang, and Y. Xiang, "Detecting and Preventing Cyber Insider Threats: A Survey," *IEEE Communications Surveys & Tutorials*, 2018.

[6] National Insider Threat Task Force. (2017) 2017 NITTF insider threat guide. The US National Counterintelligence and Security Center. [Online]. Available: https://www.dni.gov/files/NCSC/documents/nittf/NITTF-Insider-Threat-Guide-2017.pdf

[7] I. Homoliak, F. Toffalini, J. Guarnizo, Y. Elovici, and M. Ochoa, "Insight into insiders and IT: A survey of insider threat taxonomies, analysis, modeling, and countermeasures," *ACM Computing Surveys*, vol. 52, no. 2, pp. 30:1–30:40, Apr 2019.

[8] Defense Advanced Research Projects Agency. Anomaly detection at multiple scales (ADAMS). [Online]. Available: https://www.darpa.mil/program/anomaly-detection-at-multiple-scales

[9] W. Eberle, J. Graves, and L. Holder, "Insider threat detection using a graph-based approach," *J. Applied Security Research*, vol. 6, no. 1, 2011.

[10] P. Parveen, J. Evans, B. M. Thuraisingham, K. W. Hamlen, and L. Khan, "Insider threat detection using stream mining and graph mining," in *IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 2011, pp. 1102–1110.

[11] T. E. Senator, E. Chow, I. Essa, J. Jones, V. Bettadapura, D. H. Chau, O. Green, O. Kaya, A. Zakrzewska, E. Briscoe, R. I. L. Mappus, H. G. Goldberg, R. McColl, L. Weiss, T. G. Dietterich, A. Fern, W.-K. Wong, S. Das, A. Emmott, J. Irvine, J.-Y. Lee, D. Koutra, A. Memory, C. Faloutsos, D. Corkill, L. Friedland, A. Gentzel, D. Jensen, W. T. Young, B. Rees, R. Pierce, D. Huang, M. Reardon, and D. A. Bader, "Detecting insider threats in a real corporate database of computer usage activity," in *19th ACM SIGKDD*, 2013.

[12] P. A. Legg, O. Buckley, M. Goldsmith, and S. Creese, "Automated insider threat detection system using user and role-based profile assessment," *IEEE Systems Journal*, vol. 11, no. 2, 2017.

[13] B. Bose, B. Avasarala, S. Tirthapura, Y. Y. Chung, and D. Steiner, "Detecting insider threats using radish: A system for real-time anomaly detection in heterogeneous data streams," *IEEE Systems Journal*, 2017.

[14] A. Tuor, S. Kaplan, B. Hutchinson, N. Nichols, and S. Robinson, "Deep learning for unsupervised insider threat detection in structured cybersecurity data streams," in *AAAI Workshop on Artificial Intelligence for Cyber Security*, 2017.

[15] D. C. Le, S. Khanchi, A. N. Zincir-Heywood, and M. I. Heywood, "Benchmarking evolutionary computation approaches to insider threat detection," in *Genetic and Evolutionary Computation Conference*, 2018.

[16] G. Gavai, K. Sricharan, D. Gunning, J. Hanley, M. Singhal, and R. Rolleston, "Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data," *Journal of Wireless Mobile Networks, Ubiquitous Computing, & Dependable Applications*, vol. 6, no. 4, 2015.

[17] D. C. Le and A. N. Zincir-Heywood, "Machine learning based insider threat modelling and detection," in *IFIP/IEEE Symposium on Integrated Network and Service Management*, April 2019.

[18] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.

[19] R. A. A. Habeeb, F. Nasaruddin, A. Gani, I. A. T. Hashem, E. Ahmed, and M. Imran, "Real-time big data processing for anomaly detection: A survey," *International Journal of Information Management*, vol. 45, pp. 289 – 307, 2019.

[20] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017.

[21] J. J. Davis and A. J. Clark, "Data preprocessing for anomaly based network intrusion detection: A review," *Computers & Security*, vol. 30, no. 6, pp. 353 – 375, 2011.

[22] W. Wang, X. Zhang, S. Gombault, and S. J. Knapskog, "Attribute normalization in network intrusion detection," in *10th International Symposium on Pervasive Systems, Algorithms, and Networks*, Dec 2009, pp. 448–453.

[23] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood, "A hierarchical som-based intrusion detection system," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 4, pp. 439–451, 2007.

[24] D. Song, M. I. Heywood, and A. N. Zincir-Heywood, "Training genetic programming on half a million patterns: an example from anomaly detection," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 225–239, June 2005.

[25] P. Parveen and B. Thuraisingham, "Unsupervised incremental sequence learning for insider threat detection," in *IEEE International Conference on Intelligence and Security Informatics*, 2012, pp. 141–143.

[26] T. Rashid, I. Agrafiotis, and J. R. Nurse, "A new take on detecting insider threats," in *International Workshop on Managing Insider Security Threats*, 2016, pp. 47–56.

[27] D. C. Le and A. N. Zincir-Heywood, "Evaluating insider threat detection workflow using supervised and unsupervised learning," in *IEEE Security and Privacy Workshops*, 2018.

[28] J. Glasser and B. Lindauer, "Bridging the gap: A pragmatic approach to generating insider threat data," *IEEE Security and Privacy Workshops*, pp. 98–104, 2013.

[29] CERT and ExactData, LLC. Insider threat test dataset. Software Engineering Institute. [Online]. Available: https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=508099

[30] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, ser. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[31] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, 2001.

[32] D. Hoaglin, F. Mosteller, and J. Tukey, *Understanding robust and exploratory data analysis*, ser. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, 1983.

[33] I. Borg and P. J. F. Groenen, *Modern Multidimensional Scaling: Theory and Applications, Second Edition (Springer Series in Statistics)*, 2nd ed., ser. Springer Series in Statistics. Springer, 2005.

[34] W. Enders, *Applied Econometric Times Series*, ser. Wiley Series in Probability and Statistics. Wiley, 2014.

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

[36] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, Aug. 1989. [Online]. Available: https://doi.org/10.1007/bf01589116

[37] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.

[38] V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston, "Random forest: a classification and regression tool for compound classification and QSAR modeling," *Journal of Chemical Information and Computer Sciences*, vol. 43, no. 6, pp. 1947–1958, Nov. 2003. [Online]. Available: https://doi.org/10.1021/ci034160g

[39] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, "Conditional variable importance for random forests," *BMC Bioinformatics*, vol. 9, no. 1, Jul. 2008. [Online]. Available: https://doi.org/10.1186/1471-2105-9-307

[40] C. Strobl, J. Malley, and G. Tutz, "An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests." *Psychological Methods*, vol. 14, no. 4, pp. 323–348, 2009. [Online]. Available: https://doi.org/10.1037/a0016973