Outlier Detection in Random Subspaces over Data Streams: An Approach for Insider Threat Detection

Diana Haidar and Mohamed Medhat Gaber

Abstract Insider threat detection is an emergent concern for industries and governments due to the growing number of attacks in recent years. Several Machine Learning (ML) approaches have been developed to detect insider threats, however, they still suffer from a high number of false alarms. None of those approaches addressed the insider threat problem from the perspective of stream mining data where a concept drift or an outlier is an indication of an insider threat. An outlier refers to anomalous behaviour that deviates from the normal baseline of community's behaviour and is the focus of this paper. To address the shortcoming of existing approaches and realise a novel solution to the problem, we present RandSubOut (Random Subspace Outliers) approach for insider threat detection over real-time data streaming. RandSubOut allows the detection of insider threats represented as localised outliers in random feature subspaces, which would not be detected over the whole feature space, due to dimensionality. We evaluated the presented approach as an ensemble of established distance-based outlier detection methods, namely, Micro-cluster-based Continuous Outlier Detection (MCOD) and Anytime OUTlier detection (AnyOut), according to evaluation measures including True Positive (TP) and False Positive (FP).

1 Introduction

Insider threat detection is an emergent concern for academia, industries, and governments due to the growing number of attacks in recent years. Insiders

Diana Haidar
Birmingham City University, Birmingham, UK e-mail: diana.haidar@bcu.ac.uk

Mohamed Medhat Gaber
Birmingham City University, Birmingham, UK e-mail: mohamed.gaber@bcu.ac.uk

are current or former employees, contractors or business partners of an organisation who have authorised access to the network system, data, or sensitive information (e.g. trade secrets, organisation plans, and Intellectual Property) [15]. Their authorised access makes the organisation prone to insider threats that could have a vital negative impact on the system. Insider threats can be categorised into two groups, which are malicious insider threats and unintentional insider threats. Malicious insider threats are caused by insiders who exploit their privileges with the intention to compromise the confidentiality, integrity, or availability of the system and data. Their motivation would be to disclose or modify organisation's information for financial gain, for the advantage of a competitor company with the promise of a higher-paying job, or for revenge [4]. Unintentional insider threats are caused by insiders who accidentally harm the organisation's assets. This includes human mistakes and errors that are the result of faulty system design or the non-malicious negligence of insiders.

Legg *et al.* [14] stated that according to the 2011 Cybersecurity Watch Survey, 58% of attacks are associated outsiders, while 21% are associated insiders. However, the risk of insider attacks is much more than that of outsider attacks, because insiders have authorised access to sensitive data that could be used to damage the company's reputation, brand integrity, or customer confidence. An important example is the insider attack of Edward Snowden [21], a contractor hired by the National Security Agency (NSA). Snowden, in May 2013, stole approximately 1.7 million documents containing secret data from the NSA and revealed it to the public via newspapers and mass media. This incident was reported as the biggest US intelligence leakage, and has caused increased concern amongst governments and companies. Moreover, there exist various security measures for outsider threats, which is not the case for insider threats, such as firewalls, Intrusion Detection/Prevention Systems (IDS/IPS), and antiviruses.

The continuous streaming of insider threat data generated from various sources (e.g. security logs, traffic data, web requests, and email headers) increases the volume and velocity of this data acquisition. To handle the big data streams and detect insider threats, several machine learning approaches have been developed, however, they still suffer from a high number of false alarms. None of those approaches addressed the insider threat problem from the perspective of *stream mining* data where a *concept drift* or an *outlier* is an indication of an insider threat. We can recognize two types of drifts: a normal drift and a concept drift. A normal drift is the change in data distribution over time; the normal change in user's behaviour over time according to tasks assigned, developed knowledge level and other elements [24]. A concept drift may manifest in different patterns such as sudden/abrupt, incremental, gradual, and reoccurring concepts [9]. The focus of this paper is on *outliers* which refer to anomalous behaviour that deviates from the normal baseline of user's behaviour or community (i.e. a group of users having the same role) behaviour.

Hence, we present RandSubOut (Random Subspace Outliers) approach for insider threat detection over real-time data streaming. The idea is to construct an ensemble of $p+1$ outlier detection base learners, including p learners over random subspaces of feature pairs and one learner over all feature space. RandSubOut allows the detection of insider threats represented as localised outliers in random feature subspaces, which would not be the case over the whole feature space. The main contributions of this paper are summarised as follows:

- extraction of new feature types from the synthetic insider threat data by CMU-CERT [1] [11] to generate community behaviour profiles;
- presentation of RandSubOut for Insider Threat Detection with MCOD and AnyOut outlier detection base learners;
- detection of insider threats represented as distance-based outliers using RandSubOut; and
- evaluation of the performance of RandSubOut according to the evaluation measures including TP and FP.

The rest of this paper is organised as follows. Section 2 summarises the related work. Section 3 introduces some outlier detection methods over a data stream and describes RandSubOut for Insider Threat Detection. Experiments and results are discussed in Section 4. Finally, Section 5 concludes the paper and suggests future work.

2 Related Work

There exists a significant body of research on applying ML approaches for detecting insider threats. A great research trend is towards utilising probabilistic and k Nearest Neighbours (k-NN) approaches. The probabilistic ML approaches include Naive Bayes [25][19], Bayesian Network (BN) [3], and Hidden Markov Model (HMM) [12]. Others chose to combine BN and other ML algorithms, such as HMM [20] and Stochastic Gradient Boosting [5], with the aim to improve their models' performance. While, the k-NN approaches [7][8][18] are distance-based so any deviation of a user's behaviour from her/his normal baseline or the baseline of the community's behaviour flags anomalous behaviour. In addition, there exists research on implementing Fuzzy Inference Systems [24][6] to evaluate the degree of insider threat.

However, neither of those approaches addressed the insider threat problem from the perspective of stream mining data with concept drift, except for [16]. Parveen *et al.* [16] proposed an ensemble of One Class Support Vector Machine (OCSVM) to model a time series of daily logs. It maintains a k number of ensemble models having the minimum prediction error. The ensemble of OCSVM reported higher accuracy and almost half the number of

[1] https://www.cert.org/insider-threat/tools/

false positives compared to traditional OCSVM. The authors extended their work in a future paper [17] where the ensemble approach was applied on unsupervised Graph-based Anomaly Detection (GBAD). The results proved that the ensemble approach based on supervised OCSVM outperformed that based on unsupervised GBAD which reported a higher false rate.

Parveen's approach [16] models normal user's behaviour acquired from data streams over an ensemble of OCSVM classifiers in order to find a decision boundary between normal class label $c = 1$ and abnormal class label $c = 0$ (i.e. malicious insider threats). However, the idea of RandSubOut is to detect malicious insider threats represented as distance-based outliers continuously over real-time data streams. It also allows to detect localised outliers over random feature subspaces which may not be detected over the whole feature space as addressed in [16]. In addition, the paper [16] evaluated the proposed approach over 1998 DARPA Intrusion Detection data set [2] which is not specifically developed for insider threat detection. Unlike our approach RandSubOut which is evaluated over synthetic insider threat data set generated by CMU-CERT.

3 Outlier based Insider Threat Detection

This section describes an established distance-based outlier detection methods, as well as the proposed approach RandSubOut utilizing those methods for insider threat detection.

3.1 Outlier Detection Methods

The state of the art introduces different outlier detection methods over a data stream. The demonstration paper [10] provides a comparison of four distance-based outlier detection methods including STORM [1], Abstract-C [22], COD [13], and MCOD [13]. According to [10], COD and MCOD have lower space and time requirements than [1] and [22]. Moreover, the experimental evaluation in [10] shows better performance for MCOD compared to COD over benchmark tested data sets. All of the four algorithms in addition to a further distance-based outlier detection method called AnyOut [2] have been implemented in the open-source tool for Massive Online Analysis (MOA)[3] . Hence, we select MCOD and AnyOut to test their performance on synthetic insider threat data sets generated by the CERT Division at Carnegie Mellon

[2] http://www.ll.mit.edu/ideval/data/1998data.html

[3] http://moa.cms.waikato.ac.nz/

University (CMUCERT). A description of the selected algorithms is provided below.

MCOD

MCOD (Micro-cluster-based Continuous Outlier Detection) is an event-based approach that introduces the concept of evolving micro-clusters. The micro-clusters are the regions containing inlier instances with no overlapping. This allows to evaluate the range queries for new instance x with respect to the centers of micro-clusters instead of all the preceding active instances in the current window w. Thus, reducing the space and time requirements. Let S represent the set of instances that doesn't belong to any micro-cluster. For each new instance x, MCOD selects the center of the nearest micro-cluster c. If the $dist(x, c) \leq \frac{r}{2}$ such that r is the distance parameter for outlier detection, x is assigned to the corresponding micro-cluster. Otherwise, a range query for x is evaluated with respect to the set S of instances that doesn't belong to any micro-cluster. Let b represent the number of neighbours $N \subseteq S$ of x where $\forall n \in N, dist(x, n) \geq \frac{r}{2}$ and let k represent the number of neighbours parameter. If $b > \theta k$ such that parameter $\theta \geq 1$, then a new micro-cluster with center x is created and the neighbours N are assigned to this micro-cluster. A micro-cluster whose size decreases below $k + 1$ is deleted and a range query similar to that described for x is performed for each of its former instances. An instance x is flagged as an outlier, if there exists less than k instances in either S or in each micro-cluster of center c that satisfies the following: $dist(x, c) \leq \frac{3}{2}r$.

AnyOut

AnyOut (Anytime Outlier detection) is a cluster-based approach that suggests a hierarchy of clusters in tree structure named ClusTree where upper level clusters include fine grained information about lower levels. It represents a cluster by a Cluster Feature tuple $CF = (n, LS, SS)$, where n is the number of instances in the cluster, LS and SS are respectively the linear sum and the squared sum of those instances. The compact structure of the tree using CF tuples reduces space requirements. The idea of AnyOut is to traverse the tree in top-down manner and compare the current instance x_i to the clusters at each level until the arrival of new instance x_{i+1} interrupts the descent down the tree. At the moment of interruption, it inserts x_i to the cluster node e arrived into in the ClusTree and provides its degree of outlierness. Thus, maintaining real-time feedback. AnyOut defines two scores to reflect the degree of outlierness of an instance x_i. Mean outlier score is $dist(x_i, \mu(e))$

where $\mu(e)$ is the mean of e that x_i is inserted to. Density outlier score is $1 - gaus(x_i, e)$ where $gaus(x_i, e)$ is the Gaussian probability density of x_i for $\mu(e)$.

3.2 RandSubOut Approach for Insider Threat Detection

Features extracted from system and network logs describe the activity of a community. The deviation of such activity from its normal pattern according to a certain determined threshold could correlate to a malicious insider behaviour [23]. Those features are treated as *priori indicators*. For example, increased removable media activity which could correlate to IP theft; logon after working hours or from a different county location; vast number of file events like downloads, uploads, or email forwards. Another indicator is related to text categorization where specific keywords, phrases or emails could correlate to malicious insider activities. An illustrative case could be browsed suspicious websites and on the top of the list is WikiLeaks and job hunting sites.

The feature space identified in this research consists of a combination of system and network logs. The feature set assesses each user's logged activities, devices, and assigned attribute values. It is categorized into four feature types sorted according to session slots:

- *Frequency-based features* (e.g. freq. (i.e. frequency) of logon, freq. of connecting a device, freq. of copying files to removable media device, etc.);
- *Time-based features* (e.g. logon after working hours, device usage after working hours, duration of connecting a device, etc.);
- *Boolean features* (e.g. logon from new pc flag= 0,1, email-bcc non-empty, email-to or -cc or -bcc include a non-employee, etc.);
- *Attribute-based features* (e.g. freq. of visiting a particular URL, freq. of sending emails to a particular user, etc.).

We propose an approach for insider threat detection named RandSubOut (Random Subspace Outliers). The aim of RandSubOut is based on detecting localised outliers in random subspaces of feature pairs. Those localised outliers represent insider threats, which would not be detected over the whole feature space. For instance, let f represent the number of features and $p = f$ represent the number of feature subspaces. The idea is to construct an ensemble of $p+1$ base learners, including p learners over random feature subspaces and one learner over all feature space. Each learner in the ensemble will build a model, so based on a voting mechanism of those models, an alarm may be generated reporting a malicious insider threat. In the following, a detailed description of RandSubOut is provided.

Over each feature subspace:

At each window slide $wIter$, RandSubOut creates an outlier temporal list $OutTempList$ of the detected outliers $OutSet$, each associated with a temporal factor $tempF$ and the window slide $wIter$. A temporal factor $tempF$ counts the number of windows that the outlier survived in (i.e. the outlier did not turn into an inlier). For example, if an outlier $out \in OutTempList$ remains as an outlier at next window slide $wIter + 1$, $tempF$ associated to out is incremeted by 1.

RandSubOut then creates a list of flag alerts $SubFlagAlertList$ associated to the window slides $wIter$ for the corresponding feature subspace. A flag alert $flagAlert$ is a boolean variable assigned to $\{0, 1\}$. We define a vouch factor $vouchF$ as a parameter which confirms whether an outlier is a positive (i.e. a malicious insider threat). At each window slide $wIter$, we check if $tempF = vouchF$, then the outlier survived for a $vouchF$ number of windows and $flagAlert$ is assigned 1 for the window slide $wIter$ that the outlier is confirmed as a positive at. Thus, minimizing the number of false alarms (i.e. false flag alerts). Algorithm 1 gives a a pseudo code to describe the steps of the procedure over each feature subspace. Note that $outMethod$ represents the distance-based outlier detection method utilized as a base learner. This refers to either MCOD or AnyOut in this paper.

Over the ensemble of random feature subspaces:

In total, RandSubOut creates a list of votes $EnsembleVoteList$ which counts the number of times a $flagAlert$ is assigned to 1 at each window slide $wIter$ over all the random feature subspaces. This count is maintained in a variable $vote$. Let $voteF$ represent a vote factor which confirms whether a flag alarm should be generated as a total vote of the ensemble. If $vote \geq voteF$, then a an alarm is generated at this window slide $wIter$. A description for the procedure over the ensemble of random feature subspaces is provided in a pseudo code in Algorithm 2.

4 Experiments

In this section, we present the experiments performed to evaluate RandSubOut with MCOD and AnyOut based learners over CMU-CERT data. The results are displayed and discussed in order to select the optimal tuned parameters for detecting malicious insider threat represented as outliers.

Algorithm 1 Over each feature subspace

```
1: wIter ← 0
2: found ← 0
3: while nbrProcInst < nbrInst do
4:     outMethod.process(newInst)
5:     if (nbrProcInst mod w) = 0 then
6:         OutSet ← outMethod.getOutliersFound()
7:         for Outlier out : OutSet do
8:             if out ∈ OutTempList.getOutliers() then
9:                 tempF ← OutTempList.getOutTuple(out).getTempF()
10:                OutTempList.getOutTuple(out).setTempF(tempF + 1)
11:                found ← 1
12:            end if
13:            if found = 0 then
14:                tempF ← 1
15:                OutTempList.addOutTuple(out, tempF, wIter)
16:            end if
17:        end for
18:        for Outlier out ∈ OutTempList.getOutliers() do
19:            if OutTempList.getOutTuple(out).getTempF()= vouchF then
20:                if SubFlagAlertList.getFlagAlert(wIter)= 0 then
21:                    flagAlert ← 1
22:                    SubFlagAlertList.setFlagAlert(wIter, flagAlert)
23:                end if
24:                OutTempList.removeOutTuple(out)
25:            else
26:                outWIter ← OutTempList.getOutTuple(out).getWIter()
27:                if wIter − outWIter + 1 = vouchF then
28:                    OutTempList.removeOutTuple(out)
29:                end if
30:            end if
31:        end for
32:        wIter ← wIter + 1
33:    end if
34: end while
```

Algorithm 2 Over the ensemble of random feature subspaces

```
1: for FeatureSubspace FSubspace : FeatureSpace do
2:     for WindowIterSlide wIter : SubFlagAlertList.getWIters() do
3:         tempVote ← EnsembleVoteList.getVote(wIter)
4:         vote ← tempVote+ SubFlagAlertList.getFlagAlert(wIter)
5:         EnsembleVoteList.setVote(wIter, vote)
6:     end for
7: end for
```

4.1 Experimental Setup

We extracted a set of the features defined previously from the CMU-CERT data sets. This feature set builds up a community behaviour profile for each community (e.g. computer programmers, mathematicians, test engineers,etc.) sorted according to session slots. A session slot is per 4 hours to handle the detection of insider threat as soon as possible near to real-time. Since the range of feature values varies widely, we performed feature scaling in the range of $[0, 1]$.

As discussed before, a malicious insider threat may manifest in different patterns of concept drift or an outlier which is the focus of this paper. We consider a malicious insider threat scenario that maps to outliers, where a user uses a USB drive at markedly higher rate than previous activity to steal data. The total number of instances $nbrInst = 3000$ in a community behaviour model represents the number of session slots. The period of the malicious insider threat scenario maps to a start at $lowerBound = 922$ and an end at $upperBound = 1323$.

We evaluated RandSubOut as an ensemble of MCOD base learners and an ensemble of AnyOut base learners according to TP (i.e. true alarms that correlate to malicious insider threat) and FP (i.e. false alarms that correlate to normal instances detected as outliers) measures.

For MCOD, Table 1 displays a description of the tuned parameters. For AnyOut, the experiments revealed that varying the values of the parameters $trainSetSize$, $confAggr$, $conf$, and $threshold$ does not have a significant effect on the results, so they were assigned to fixed values. However, we tuned other parameters as described in Table 2.

Table 1 MCOD Tuned Parameters.

$k = \{50, 60, 70\}$	number of neighbours parameter
$r = \{0.3, 0.4, 0.5, 0.6, 0.7\}$	distance parameter for outlier detection
$w = \{100, 150, 200\}$	window size

Table 2 AnyOut Tuned Parameters.

$trainSetSize = 500$	training set size
$confAggr = 2$	size of confidence aggregate
$conf = 4$	initial confidence value
$threshold = 0.4$	outlier score threshold
$oScoreAggr = \{2, 8\}$	size of outlier score aggregate
$w = \{100, 200\}$	window size

The vouch factor $vouchF$ defined previously is assigned $vouchF = 2$ for MCOD, so it confirms whether an outlier is a positive after it survives for 2 window slide iterations. However, it is tuned to $vouchF = \{2, 4\}$ for AnyOut, because it affects the results, which is not the case for MCOD where no flag alarms are generated then. The vote Factor $voteF$ is assigned $voteF = 2$, so it confirms whether a flag alarm should be generated as a total vote of the ensemble.

4.2 Results and Discussion

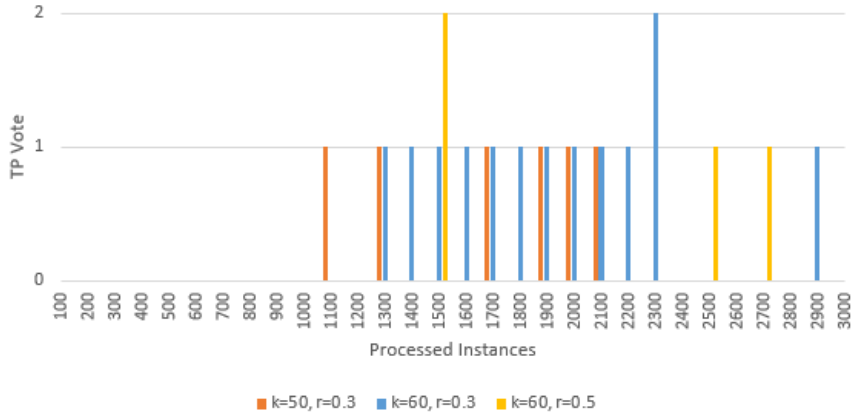

Fig. 1 TP votes generated by RandSubOut with MCOD over $w = 100$.

Fig.1, Fig.2, and Fig.3 show the number of TP votes generated by RandSubOut using MCOD base learners over window size $w = 100$, $w = 150$, and $w = 200$ respectively with varying values of parameters k and r. Note that the parameters which didn't show TP votes are not displayed in figures. The window slide iteration $wIter$ ends at the number of processed instances $nbrProcInst$ and starts at $nbrProcInst - w$ (e.g. for $w = 100$, the first window slide iteration $wIter = 1$ ends at $nbrProcInst = 100$ and starts at $nbrProcInst - w = 0$). In order to satisfy real-time or near real-time detection of a malicious insider threat (i.e. TP), we should consider the flag alarms generated at real-time or near real-time. For instance, in Fig.1, if $vouchF = 2$, then a flag alarm of TP votes should be generated between $lowerVouchBound = ceil(lowerBound + w \times (vouchF - 1)) = floor(922 + 100 \times 1) = 1100$ and $upperVouchBound = floor(upperBound + w \times vouchF) = ceil(1323 + 100 \times 1) = 1500$. Fig.1 shows that the alarms started to be flagged at $lowerVouchBound = 1100$, however more flag alarms

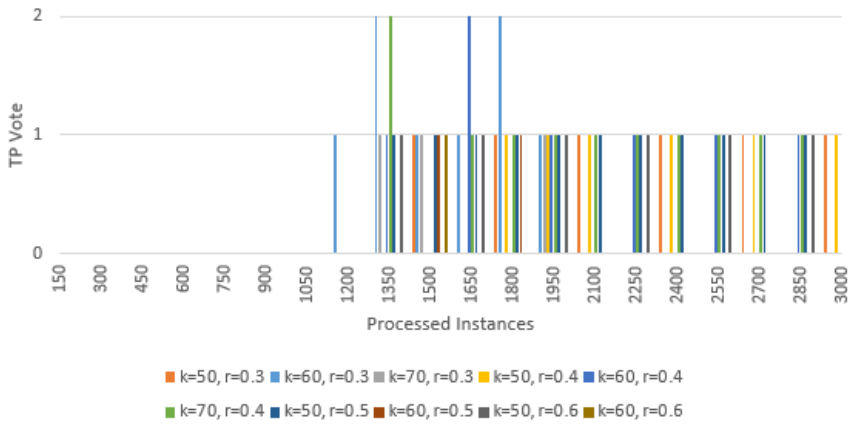

Fig. 2 TP votes generated by RandSubOut with MCOD over $w = 150$.

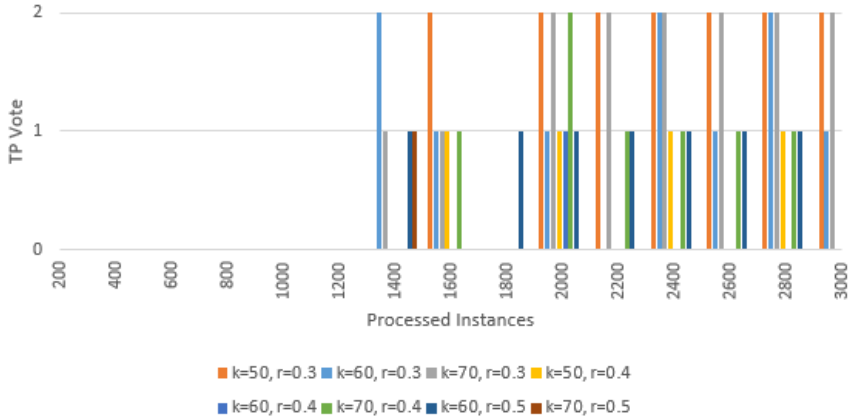

Fig. 3 TP votes generated by RandSubOut with MCOD over $w = 200$.

are generated after $upperVouchBound = 1500$. This is justified by that instances assigned as inliers by RandSubOut may turn into outliers after a number of window slide iterations. In MCOD, an instance may not belong to any micro-cluster but still is an inlier, thus considered a potential outlier. RandSubOut considers a flag alarm as a true if it signals a $vote \geq voteF$ and it is generated between $lowerVouchBound$ and $upperVouchBound$. This means that at least two feature subspaces voted for a flag alarm at this window slide iteration $wIter$. Thus, minimizing the number of false alarms. To select the optimal parameter values, we consider the parameter values which satisfies a true flag alarms ONLY generated between $lowerVouchBound$ and $upperVouchBound$. Fig. 1 for $k = 60, r = 0.5, w = 100$ reports a true alarm with $vote \geq voteF; voteF = 2$ at $nbrProcInst = 1500$ (i.e.

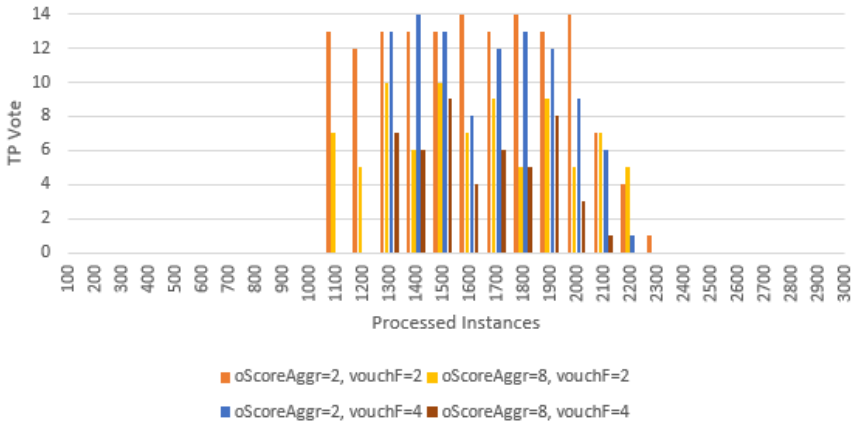

Fig. 4 TP votes generated by RandSubOut with AnyOut over $w = 100$.

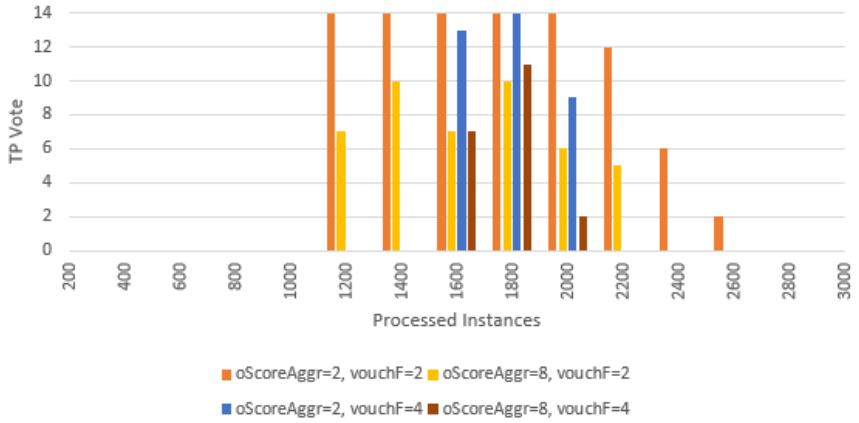

Fig. 5 TP votes generated by RandSubOut with AnyOut over $w = 200$.

$wIter = nbrProcInst \div w = 1500 \div 100 = 15$ such that $wIter$ starts at 0). Also, Fig. 2 for $k = 70, r = 0.4, w = 150$ reports a true alarm with $vote \geq 2$ for $TP = 1$ at $nbrProcInst = 1350$, where $lowerVouchBound = 1200$ and $upperVouchBound = 1500$ (i.e. $wIter = 9$). However, no case in Fig.3 satisfies the condition of a true flag alarm ONLY generated between $lowerVouchBound$ and $upperVouchBound$ with $vote \geq voteF; voteF = 2$ for $TP = 1$.

Similarly, Fig.4, and Fig. 5 show the number of TP votes generated by RandSubOut using AnyOut base learners over window size $w = 100$, and $w = 200$ respectively with varying values of parameters $oScoreAggr$ and $vouchF$. It is clear that no case in both figures satisfies the condition of a true flag alarm ONLY generated between $lowerVouchBound$ and $upperVouchBound$

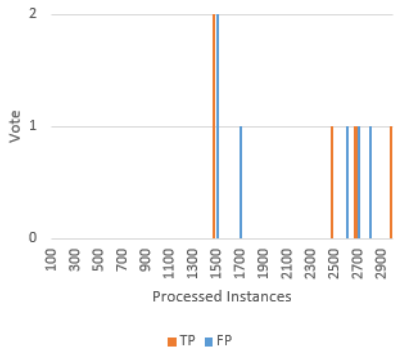

Fig. 6 TP votes vs. FP votes generated by RandSubOut with MCOD over $w = 100$.

Fig. 7 TP votes vs. FP votes generated by RandSubOut with MCOD over $w = 150$.

with $vote \geq voteF; voteF = 2$. Moreover, the results revealed flag alarms with a great number of votes which do not map to the malicious insider threat scenario. For instance, the worst case of $vote = 14$ implies that 14 feature subspaces voted for this flag alarm as true, so approximately 14 or less random features reported a malicious insider threat which is not the case. Mapping to the malicious insider threat scenario considered, only few features such as *freq. of connecting a device* and *freq. of copying files to removable media device* would only report a malicious insider threat. The reason behind those unstatisfying results would correlate to that an outlier score of an instance x_i is computed once a new instance x_{i+1} arrives. This means that the processing is interrupted and the confidence of the outlier score computed would be very low.

Based on the above experiments, we select two cases of RandSubOut with MCOD base learners. The first case for $k = 60, r = 0.5, w = 100$ is reported in Fig. 6 showing the TP votes vs. FP votes with respect to processed instances. The second case for $k = 70, r = 0.4, w = 150$ is reported in Fig. 7 showing the TP votes vs. FP votes with respect to processed instances. Fig. 6 flags a true alarm with $vote = 2$ for $TP = 1$ and a false alarm with $vote = 2$ for $FP = 1$ at $nbrProcInst = 1500$. However, Fig. 7 only flags a true alarm with $vote = 2$ for $TP = 1$ at $nbrProcInst = 1350$, but it didn't flag any false alarm with $vote \geq 2$. Thus, in this case $FP = 0$ where no false alarms are generated. Hence, the vote goes to RandSubOut with MCOD base learners for $k = 70, r = 0.4, w = 150, vouchF = 2, voteF = 2$.

5 Conclusion

This paper addresses the problem of high number of false alarms in the existing insider threat detection approaches from the perspective of stream mining data where a concept drift is an indication of an insider threat. We present RandSubOut approach for insider threat detection over real-time data streaming. RandSubOut allows to detect insider threats represented as localised outliers over random feature subspaces, which may not be detected over the whole feature space. It also introduces two parameters: a vouch factor to confirm whether a flag alarm should be generated over a feature subspace, and a vote factor to confirm whether a flag alarm should be generated as a total vote of the ensemble. We evaluated RandSubOut as an ensemble of MCOD and AnyOut outlier detection methods, according to evaluation measures including TP and FP. The experiments revealed that RandSubOut with MCOD base learners outperforms RandSubOut with AnyOut base learners. Moreover, RandSubOut with MCOD base learners for $k = 70, r = 0.4, w = 150 vouchF = 2, voteF = 2$ provided the best results, where it votes for true alarms with $vote \geq 2$ for $TP = 1$ at real-time or near real-time. In this case, the votes for FP are ≤ 2, so no false alarms were generated.

Future work will be to address malicious insider threat scenarios that map to the different patterns of concept drift other than outliers. It will also tackle how to distinguish between a normal drift and a concept drift over real-time data streaming.

References

[1] Angiulli F, Fassetti F (2010) Distance-based outlier queries in data streams: the novel task and algorithms. Data Mining and Knowledge Discovery 20(2):290–324

[2] Assent I, Kranen P, Baldauf C, Seidl T (2012) Anyout: Anytime outlier detection on streaming data. In: International Conference on Database Systems for Advanced Applications, Springer, pp 228–242

[3] Axelrad ET, Sticha PJ, Brdiczka O, Shen J (2013) A bayesian network model for predicting insider threats. In: Security and Privacy Workshops (SPW), 2013 IEEE, IEEE, pp 82–89

[4] Azaria A, Richardson A, Kraus S, Subrahmanian V (2014) Behavioral analysis of insider threat: a survey and bootstrapped prediction in imbalanced data. IEEE Transactions on Computational Social Systems 1(2):135–155

[5] Barrios RM (2013) A multi-leveled approach to intrusion detection and the insider threat. Journal of Information Security 4(01):54

[6] Bin Ahmad M, Akram A, Asif M, Ur-Rehman S (2014) Using genetic algorithm to minimize false alarms in insider threats detection of information misuse in windows environment. Mathematical Problems in Engineering 2014

[7] Chen Y, Malin B (2011) Detection of anomalous insiders in collaborative environments via relational analysis of access logs. In: Proceedings of the first ACM conference on Data and application security and privacy, ACM, pp 63–74

[8] Chen Y, Nyemba S, Malin B (2012) Detecting anomalous insiders in collaborative information systems. IEEE transactions on dependable and secure computing 9(3):332–344

[9] Gama J, Žliobait I, Bifet A, Pechenizkiy M, Bouchachia A (2014) A survey on concept drift adaptation. ACM Computing Surveys (CSUR) 46(4):44

[10] Georgiadis D, Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, Manolopoulos Y (2013) Continuous outlier detection in data streams: an extensible framework and state-of-the-art algorithms. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data, ACM, pp 1061–1064

[11] Glasser J, Lindauer B (2013) Bridging the gap: A pragmatic approach to generating insider threat data. In: Security and Privacy Workshops (SPW), 2013 IEEE, IEEE, pp 98–104

[12] Huang L, Stamp M (2011) Masquerade detection using profile hidden markov models. computers & security 30(8):732–747

[13] Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, Manolopoulos Y (2011) Continuous monitoring of distance-based outliers over data streams. In: 2011 IEEE 27th International Conference on Data Engineering, IEEE, pp 135–146

[14] Legg PA, Moffat N, Nurse JR, Happa J, Agrafiotis I, Goldsmith M, Creese S (2013) Towards a conceptual model and reasoning structure for insider threat detection. JoWUA 4(4):20–37

[15] Nurse JR, Legg PA, Buckley O, Agrafiotis I, Wright G, Whitty M, Upton D, Goldsmith M, Creese S (2014) A critical reflection on the threat from human insiders–its nature, industry perceptions, and detection approaches. In: International Conference on Human Aspects of Information Security, Privacy, and Trust, Springer, pp 270–281

[16] Parveen P, Weger ZR, Thuraisingham B, Hamlen K, Khan L (2011) Supervised learning for insider threat detection using stream mining. In: 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence, IEEE, pp 1032–1039

[17] Parveen P, Mcdaniel N, Weger Z, Evans J, Thuraisingham B, Hamlen K, Khan L (2013) Evolving insider threat detection stream mining perspective. International Journal on Artificial Intelligence Tools 22(05):1360,013

[18] Punithavathani DS, Sujatha K, Jain JM (2015) Surveillance of anomaly and misuse in critical networks to counter insider threats using computational intelligence. Cluster Computing 18(1):435–451

[19] Sen S (2014) Using instance-weighted naive bayes for adapting concept drift in masquerade detection. International Journal of Information Security 13(6):583–590

[20] Tang K, Zhou MT, Wang WY (2009) Insider cyber threat situational awareness framwork using dynamic bayesian networks. In: Computer Science & Education, 2009. ICCSE'09. 4th International Conference on, IEEE, pp 1146–1150

[21] Verble J (2014) The nsa and edward snowden: surveillance in the 21st century. ACM SIGCAS Computers and Society 44(3):14–20

[22] Yang D, Rundensteiner EA, Ward MO (2009) Neighbor-based pattern detection for windows over streaming data. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, ACM, pp 529–540

[23] Young WT, Goldberg HG, Memory A, Sartain JF, Senator TE (2013) Use of domain knowledge to detect insider threats in computer activities. In: Security and Privacy Workshops (SPW), 2013 IEEE, IEEE, pp 60–67

[24] Yu Y, Graham JH (2006) Anomaly instruction detection of masqueraders and threat evaluation using fuzzy logic. In: 2006 IEEE International Conference on Systems, Man and Cybernetics, IEEE, vol 3, pp 2309–2314

[25] Zhang R, Chen X, Shi J, Xu F, Pu Y (2014) Detecting insider threat based on document access behavior analysis. In: Asia-Pacific Web Conference, Springer, pp 376–387