Feature Engineering in Big Data for Detection of Information Systems Misuse

Eduardo Lopez*
McMaster University
Hamilton, Ontario
lopeze1@mcmaster.ca

Kamran Sartipi
East Carolina University
Greenville, NC
sartipik16@ecu.edu

ABSTRACT

The increasing availability of very large volumes of digital data (i.e. Big Data) enables many interesting research streams on a wide variety of phenomena. However, there has been a paucity of Big Data sets in the area of cybersecurity in information systems, as organizations are reluctant to share data that may provide too much unrestricted visibility into their operations. In this study, we explore the use of a real-life, anonymized, very large dataset containing user behavior – as captured in log files – including both regular usage as well as misuse, typifying the dynamics found in a situation with compromised user credentials. Through the experiment, we validate that the existence of a large user behavior dataset in itself does not necessarily guarantee that abnormal behaviors can be found. It is essential that researchers apply deep domain knowledge, critical thinking and practical focus to ensure the data can produce the knowledge required for the ultimate objective of detecting an insider's threat. In this paper we develop, formulate and calculate the features that best represent user behavior in the underlying information systems, maintaining a parsimonious balance between complexity, resource demands and detection effectiveness. We test the use of a classification model that proves the usefulness and aplicability of the features extracted.

CCS CONCEPTS

• **Security and privacy** → *Formal methods and theory of security*; *Domain-specific security and privacy architectures*; Human and societal aspects of security and privacy;

KEYWORDS

feature engineering; anomaly detection; big data security; insider's threat; predicting misuse

ACM Reference Format:
Eduardo Lopez and Kamran Sartipi. 2018. Feature Engineering in Big Data for Detection of Information Systems Misuse. In *Proceedings of 28th Annual International Conference on Computer Science and Software Engineering (CASCON'18)*, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 12 pages. https://doi.org/10.475/123_4

*Eduardo Lopez, DeGroote School of Business, McMaster University. Correspondence concerning this article should be addressed to: Eduardo Lopez 1280 Main St W, Hamilton, ON L8S 4L8. E-mail: lopeze1@mcmaster.ca

CASCON'18, October 2018, Markham, Ontario, Canada

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

1 INTRODUCTION

Information Technology has dramatically changed society in many ways. The workplace is a radically different environment since Information Systems (IS) became one of the strongest enablers for its processes. Almost every organization uses – and depends – on IS for the delivery of value. But IS can be misused. Security issues in IS are becoming more prevalent, and are being more widely reported than ever before. News pertaining to the stealing of IS users' data such as the Yahoo incident [19] and the stolen credit data of non-users from the company Equifax [16] have significantly increased public awareness on the importance of cybersecurity. Furthermore, the very public coverage of the data released by Edward Snowden [24]- which we can consider an 'insider's threat' becoming a reality - highlight the growing importance of security and privacy in the fabric of today's society in general, and IS in particular. In addition to the significant damage suffered by the millions of direct victims of these acts, the stakeholders (organizations and individuals) deemed accountable (regardless of who was responsible) for these events have faced severe consequences in cost, reputation and even in their lives and careers [22].

The insider's threat, in particular, is a growing issue in organizations. It can be defined as a threat originating from users (or somebody impersonating them) who have been given access rights to an IS and misuse their privileges, impacting the confidentiality, availability or integrity of the information deliberately or because of non-compliance [27].

Insider's misuse of IS is a significant challenge for organizations, as exemplified in the leak of diplomatic papers by Chelsea Manning [23]. Insiders accounted for as much as 39% of data breaches in 2015, through accidental or deliberate misuse of data [21], but this rate is poised to increase with the growing number of users and interconnected systems in an ever-more technified society. The Insider's threat in IS as a phenomena provides fascinating elements: the use of IS is regularly and systematically stored electronically through the creation of *logs*. These are files that record the events and interactions that users have with an IS. A thorough and timely review of the logs from the various IS used may help the detection of abnormal behaviors that can signal a misuse of users' credentials. However, this task is complex, resource-intensive and may not suffice for the identification of the threat. The logs usually store information in a non-structured manner – very large text files in some cases –, and capture myriad different events that may be non-related to the behavior of the user in the IS. Even the task

of searching for the abnormal behavior can become impractical unless there is a way to separate normal from abnormal observations – most of the time without previous knowledge or a varying conception of what normality looks like.

The analysis of user behavior patterns requires a projection of the data into a space in which computational models can be utilized. At the most basic level, we are interested in the *features* that best represent user behaviors in the information system. A feature can be defined as a variable that describes aspects of the objects in scope [9]. A feature shall define, characterize or identify the underlying phenomena in a manner that can be used by downstream processes. The process of *feature engineering* may involve mathematical transformation of the raw data, feature extraction and/or generation, feature selection and feature evaluation. The processes may include (dis)aggregation, cleansing, coding and/or imputation of variables [12].

In this experiment we use a very large, anonymized dataset - almost 1.6 billion records - that has been collected from production systems running for a period of two months. The data includes labeled misuse of the IS, providing a unique opportunity to perform and assess the strength of the feature engineering process.

Our contribution to researchers and practitioners include:

Selection of effective user behavior features. we suggest the most relevant features enabling analysis of user behavior towards detecting IS insider's misuse. These features can be used as the departing point for the creation of insider's threat detection systems in academia and industry.

Efficient feature extraction and transformation. we share our learnings extracting and transforming variables on a large scale environment that typifies real conditions, and how the efficiency needs are paramount for a timely outcome.

Insider's threat detection system architecture. we articulate a system architecture that meets the needs of our current scope, and can be implemented at scale for very large distributed systems.

The remainder of this paper is organized as follows. Section 2 provides the required background information useful in understanding the experiment performed. In Section 3 we describe the real-world, large dataset we use in this research. Section 4 discusses the proposed approach including the system architecture. In Section 5 we document our experiment results, and we conclude the paper with a discussion in Section 6.

2 BACKGROUND AND RELATED WORK

In this section we provide key elements that are useful in the interpretation of the results of this study.

2.1 Machine Learning

Within the scope of Artificial Intelligence (AI), machine learning has taken a central stage. It is concerned with a specific application of AI, in that it seeks to design artifacts that are able to learn based on data. This is in contrast with the typical computing paradigm of programming software declaratively to react to specific actions or events. In the machine learning realm, there are few literal instructions given to the system, but rather it is trained based on examples (i.e. historical data).

The output of machine learning constructs can be divided broadly in two kinds: a parametric one in which there is an assumption on the functional relationship between the variables (such as linear or logistic regression), and a non-parametric one in which no parameters (such as the β in regression) are estimated and that typically would require large amounts of data for its proper functioning.

Machine learning tasks can be broadly categorized as *supervised* or *unsupervised*. Supervised learning happens when the AI artifact is trained using data that has been previously labeled, or classified. In the detection of IS' misuse by insiders, the AI would be given examples (labeled data) on both normal and abnormal behaviors, ultimately creating a model to use in the classification of observations. In unsupervised learning, the machine learning artifact is given the historical data without labels. In other words, the machine learning algorithm needs to 'decide' first what the normality baseline is, in order to identify potential abnormal behaviors.

Very broadly we can abstract the actions that are possible in machine learning in four types: classification, prediction, clustering and pattern mining.

In *classification* activities, an AI identifies an object or observation as belonging to a specific class. This is typically a supervised task since the labels have to be known when the actual classification is performed. In the insider's threat detection, we have a typical two-class (or binary) classification model: whether an observation is (or not) insider's misuse.

In *prediction*, an unknown value is forecasted. E.g. the actual number of computers accessed by a user. Some times this activity is called regression since this process typifies the prediction of a value.

In *clustering*, the labels or classes, are not known in advance – rather the AI groups objects together based on predefined measures of similarity. E.g. grouping users by computer usage volume.

In *pattern mining* the AI finds reoccurring regularities that appear in the data. E.g. what user behaviors are most associated with one another. These are the four basic activities that can be executed

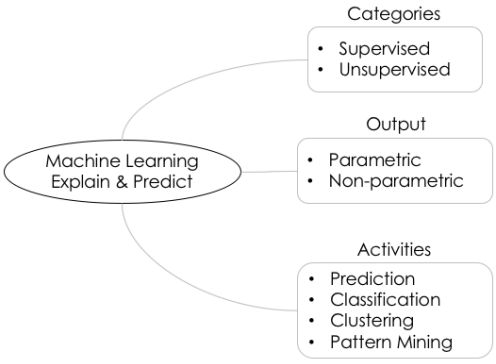
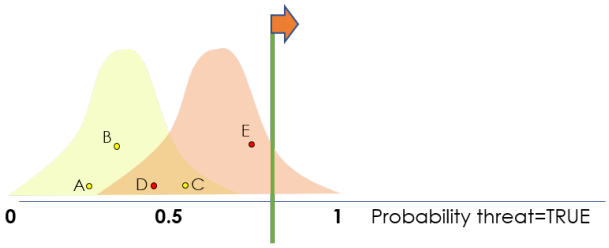
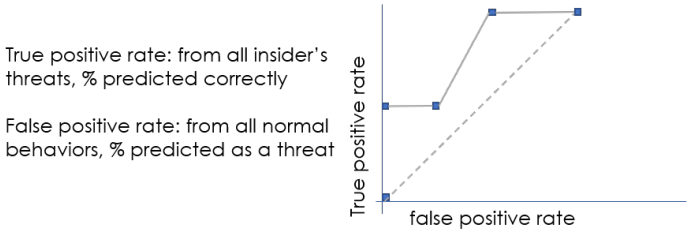


Figure 1: Mind map of machine learning concepts.



(a) User behaviors



(b) ROC curve

Actual or "ground truth" Estimated by the classifier model

Observation	Actual Class	Probability threat=TRUE	Prediction @ threshold<0.25	Prediction @ threshold=0.3	Prediction @ threshold=0.5	Prediction @ threshold=0.65	Prediction @ threshold>0.75
A	Threat=FALSE	0.25	TRUE	FALSE	FALSE	FALSE	FALSE
B	Threat=FALSE	0.35	TRUE	TRUE	FALSE	FALSE	FALSE
D	Threat=TRUE	0.45	TRUE	TRUE	FALSE	FALSE	FALSE
C	Threat=FALSE	0.55	TRUE	TRUE	TRUE	FALSE	FALSE
E	Threat=TRUE	0.75	TRUE	TRUE	TRUE	TRUE	FALSE
	Sensitivity (i.e. true positives)		100%	100%	50%	50%	0%
	1-Specificity (i.e. false positives)		100%	66%	33%	0%	0%

(c) Classifier

Figure 2: Classification model performance: example with four observations.

with machine learning. Other activities that are more specialized can be derived as ensembles of these four foundational tasks, or may include specific implementations such as density estimation or dimensionality reduction. A summary of the machine learning concepts is depicted as a mind map in Figure 1.

2.2 Success Metrics in Binary Classification Models

The detection of insider's misuse of IS can be formulated as a binary classification problem. In other words, the model receives as input an observation (i.e. user behavior) and produces a probability (between 0 and 1) of that observation being an insider's threat. For illustration purposes, we assume that the number of actual threats and of normal behaviors follow a normal distribution, as depicted in Figure 2(a). The ground truth – known and labeled data – is that A, B and C are normal behaviors, whereas D and E are insider threats taking place. If the probability of the observation being a threat is higher than a specified threshold, the system classifies the observation as an insider's threat. In the table on Figure 2(c) we display the resulting probabilities for each observation under five different thresholds: <0.25, 0.3, 0.5, 0.65 and >0.75. Choosing one of these thresholds would lead to different accuracy rates. These results can be quantified using a confusion matrix. Please refer to Figure 3.

X represents the actual number of insider's threats that were predicted correctly. W is the number of normal events that were correctly predicted as normal. Y and Z represent the number of observations that were predicted as one class but are in reality a

		Predicted	
		YES – it's a threat	NO – it's normal
Actual	YES	X	Y
	NO	Z	W

Figure 3: Confusion matrix for a two-class or binary classification.

member of the other class. We define sensitivity as the ability of the model for correctly predicting an insider's threat ('true positives'), and specificity of the classifier as its ability to predict normal behavior ('true negatives').

$$sensitivity = true\_positive\_rate = \frac{X}{X+Y}$$

$$specificity = true\_negative\_rate = \frac{W}{Z+W}$$

Sensitivity is also referred to as the *true-positive rate* or *probability of detection*. Conversely, specificity of the model is defined as the ability to rule out observations as non-anomalous correctly. We can also call it *true-negative rate*. Whereas sensitivity quantifies the avoiding of false negatives, specificity quantifies the avoiding of false positives. A perfect classifier has a sensitivity of 100% and a

specificity of 100%. In reality, classification models exhibit a trade-off between the two values. If we depict the confusion matrices for the thresholds of 0.3 and 0.5, they would show the values in Figure 4.

Thres.= 0.3		Predicted: YES – it's a threat	Predicted: NO – it's normal
Actual	YES	2 (D, E)	0
	NO	2 (B,C)	1 (A)

Thres.= 0.5		Predicted: YES – it's a threat	Predicted: NO – it's normal
Actual	YES	1 (E)	1 (D)
	NO	1 (C)	2 (A,B)

Figure 4: Confusion matrices for thresholds at 0.3 and 0.5 for the classification model and observations A, B, C, D and E.

The true positive and false positive rates will change depending on the threshold we select. If we plot these two rates for all threshold values and all available observations, we obtail a Receiver Operating Characteristic or ROC curve, depicted on Figure 2(b).

A very good classifier that separates correctly the two classes would tend to the upper left point [0,1]. A very bad classifier would be equivalent to a random draw, with a curve close to the diagonal. The Area Under the Curve (AUC) is usually used to represent the quality of a given classifier as captured in a ROC curve [7].

ROC curves are very popular in medical and clinical applications, since it is a suitable tools to run experiments and compare different models. In the case of the insider's threat detection, we use ROC curves, and more specifically the AUC, for the purposes of validating that the features we selected can enable a good classification model with AUC larger than 0.5.

2.3 Feature Engineering

Machine learning models have as a defining characteristic the need for large amounts of data. However, the existence of large datasets is in itself not sufficient for obtaining results through computational and mathematical models [8]. It is possible that the data does not *represent* well the characteristics that are the objective of the analysis. The activities undertaken to ensure the dataset suitably represents the knowledge we are seeking through inductive analysis is called feature engineering.

It is possible to abstract the feature engineering as multiple iterations of three tasks in sequence: feature construction, feature selection and feature evaluation [9]. The construction of the features may involve plethora of activities oriented towards the meaningful representation of the data. (Dis)aggregation, constraining, (de)normalization, categorization, binning or other transformations may be required for the conversion of raw data into a task-representative feature space. Effective feature construction usually requires solid domain knowledge and a pragmatical goal-orientation. The same raw dataset may provide different features depending on the ultimate objective for the computational model. The output of feature construction is a rich feature set that enables computational models' use.

In many situations, the constructed set is large, evidencing the 'course of dimensionality' in which a significant number of dimensions are represented in the features. Some of the features may not bring new information, or may not represent well the knowledge we are looking for. Thus, the process of feature selection ensures that the machine learning model has the most uniquely relevant features that are fit-for-purpose.

Once the feature set has been determined, it can be tested against the data. If the data is labeled, multiple accuracy metrics can be utilized for assessing how well the feature set represents the knowledge of interest. Feature evaluation is also an important process to perform when there is feature and/or concept drift. Feature evolution is present when the features used to represent the data change, e.g. when new tags are used to represent existing images in a system. The concept drift occurs when the same features represent a different class. Concept evolution happens when new classes appear in the phenomena.

Research in feature engineering is extensive and diverse. In some cases, the nature of the data drives the feature engineering activities to be conducted. This is typified by the multiple applications related to Natural Language Processing (NPL). From the initial work for the use of appropriate terms in document indexing [17], the field has evolved into developing features that go beyond the classical bag of words and into syntactic, stylistic [11] and semantic feature extraction [14]. For visual data, object detection and speech recognition, there has been an evolution from feature extraction tasks designed by the researcher using specific domain knowledge to the automated discovery of latent features, most recently using deep architectures in neural networks [15]. The use of deep learning for the analysis of system logs shows promising results [10], potentially leading the way to its application to the insider threat phenomena.

In terms of feature engineering as applied to the insider's threat, it is essential to recognize the particular characteristics that the phenomena provides. The raw data for this experiment comes from multiple system logs. This file captures multiple events – some time unrelated to each other – that happened on a specific time frame in a particular resource. Having a time stamp for each event means that the data structure has some characteristics typical of a time-series domain. Meaningful features may be extracted through characterization methods using autocorrelation, distribution, stationarity, entropy and non-linear time-series analysis [9]. The transformation of time-based data into a feature space may allow understanding at a different level of abstraction. However, it is important to recognize that time-series feature extraction alone may be unsuitable to the understanding of user behavior. A user may perform tasks at varying speeds, potentially identifying the same sequence as different patterns. A complementary approach may involve the use of timeless features (sometimes referred to as spatial) as well as temporal features. In detecting the insider's misuse, the total number of authentications, or the count of computers accessed over the whole analysis timespan may provide relevant information. Using the spatial features when constrained by timing values would provide the additional insight that may indicate changes in behaviors that are due to information systems misuse.

A second critical nature of the insider's threat data pertains to its condition as a data stream. A system log may be viewed as a pipeline of observations that capture events that are happening to the entities in the space. Data streams mean not only new observation, but may also change the context in relevant ways for

the machine learning artifact. We can have one or several of the following:

Feature evolution. In this case, there is a varying number of features in the data stream at different points in time. To address feature evolution, it is possible to perform feature construction through linear projection functions. One of the most used is an application of Principal Component Analysis (PCA), which calculates a new set of features based on a linear combination of the existing variables so it is possible to account with most of the variance in the data [18]. In this case PCA addresses not only the course of dimensionality but also ensuring that the principal components are the transformed features to use in the identification of the threat. When the distribution of the data is not linear, an extension called Kernel PCA can be used [13].

Concept drift. This is present whenever the thresholds for class determination change [25]. In the insider's threat, concept drift is expected, thus it is important that the model be re-represented so the new dynamics linking feature to classes are suitably updated.

Concept evolution. New observations may imply that there are new classes not seem previously [25]. Since we have interpreted the insider threat as a binary classification model, we expect the same two classes (normal, insider threat) to be constant.

A third characteristic that is relevant to the insider's threat detection pertains to the sequence patterns in the data. A system log contains information about the specific sequence of activities a user is following. User behaviors interpreted as a sequence of actions or events can be modeled through graphs. Feature construction from sequence data or graphs involve the mining of patterns: this may include association mining, frequent sequences identification. Given the very large number of patterns that can be extracted from system logs, it is important to employ strategies to reduce the search space.

Feature engineering for user behaviors with the goal of detecting insider misuse of information systems may draw concepts from multiple domains in its quest to represent the underlying information of interest.

Since the challenge we face in the insider's IS misuse detection pertain to the behavior of the user within the system, we require markers that clearly delineate the user's actions in the system. Two common types of behavioral features found in the literature [26] are defined as follows:

2.4 Learning Approaches for Insider Threat Detection

Broadly speaking, a classification model is used when detecting the insider's misuse of an information system. The classifier may deem a user behavior in the IS as normal or anomalous. If there is labeled data that the classifier can use – a 'signature' – the detection of IS misuse will rely on comparing each observation to the baseline. This is akin to the classical approach that can be found in many of the anti-virus software packages available. A signature file is used as the yardstick in detecting malicious software that may have been downloaded to a computer. This approach is resource-efficient since the computing task performed is a relatively trivial direct comparison. The false positives (normal behaviors identified as abnormal) are usually few since the control is explicit in its appraisal and the outcome unequivocal. The main drawback from this method is that the threat agent can avoid the control by using different, new abnormal behaviors for which no signature yet exists.

In contrast, an anomaly detection procedure concentrates on defining what a normal behavior looks like first – in as close to real-time as possible – and compares it against the observation being assessed. Anomaly detection is a widely researched topic in scholarly literature. It can be defined identification of observations or cases that do not follow the established normal patterns. Anomaly detection is inherently an unsupervised learning task, as It assumes no pre-labeled data exists describing what normal or anomaly is. Anomaly detection as a method can be found in many domains, as the identification of anomalies is applicable across many activities. From the vantage point of cybersecurity and the insider's misuse of IS, the data available to the system administrator – or the AI artifacts – is limited to the logs capturing the activity and events on any given system. This data has not been classified until that moment. The task at hand is, therefore, to evaluate a set of historical observations stored in the log file and use them to classify a new observation as normal or abnormal. Anomaly detection focuses first on creating a profile that represents normality [**?**] and then compares the created baseline against the new observations. To some degree, anomaly detection can be described as a process of creating the signatures dynamically, and then comparing the observations against them. The drawbacks of the anomaly detection approach include that it is resource-intensive, and may have a large number of true negatives (or behaviors considered anomalous when they are normal). These two aspects are becoming less of an issue as the computing power available grows and the sophistication of the algorithms available increases.

From the vantage point of cybersecurity, anomaly detection can be found in multiple works across many areas, as a established technique for the identification of anomalous behavior [20].

3 DATASET

Having the opportunity of using real-life datasets for the purposes of security research is rather uncommon as organizations are reluctant to share information that may provide too much visibility into their operations and the potential security issues they have encountered. There are some databases available to researchers but they are usually fully synthetic, outdated or not very large. The paucity of data makes the process of articulating or testing security-centric theories difficult. Fortunately, a dataset has been made available from the Los Alamos National Laboratory in the United States [2]. This anonymized dataset is the result of 58 days of continuous monitoring on multiple systems in their computer network, for more than 12,000+ users and 17,000+ computers [6]. The data includes labeled observations identifying known compromised users and events, creating a remarkable opportunity to analyze and test the effectiveness and efficiency of IS misuse detection techniques when applied to the insider's threat identification. The processing of this large dataset characterizes well the challenges associated with Big Data. It contains large volumes of unstructured data, representing high-velocity creation of logs from multiple computer systems (routers, desktop and server machines) and their users. The dataset

also includes labeled data identifying known compromising events. The dataset is 90 GB+ with approximately 1.6 billion records.

The following is a description of the data entities present in the dataset.

- *Authentications.* This dataset contains 1.051 billion authentication events on individual computers – servers and workstations. Each of these records is an authentication event, i.e. a user or a computer loging into a system. It includes nine variables for each observation. It collects the time (measured in seconds since the beginning of the timeline), the source and destination user, the source and destination computers, the authentication type, the logon type, the authentication orientation, and whether it was a successful attempt or not. Figure 5 shows a representative list of records.

```
1,C625$@DOM1,U147@DOM1,C625,C625,Negotiate,Batch,LogOn,Success
1,C653$@DOM1,SYSTEM@C653,C653,C653,Negotiate,Service,LogOn,Success
1,C660$@DOM1,SYSTEM@C660,C660,C660,Negotiate,Service,LogOn,Success
```

Figure 5: Sample authentication records of the dataset.

- *Domain Name Service.* This dataset contains 40,821,591 records. The events recorded in this dataset pertain to the translation of network names (such as 'www.google.com') to their numerical equivalent in the underlying network (also called an IP address).
- *Network Flows.* This dataset contains 129,977,412 records. The events recorded in this table capture the routing actions performed by the network computing devices.
- *Processes* . This table contains 426,045,096 records. It lists all the processes (which we will equate to applications or programs) that the user ran on the computer system directly or as a result of an application being used in a system. It contains the time in which it took place (in seconds), the computer or user@domain that ran the process, the computer in which it was executed, the process identifier and whether it is start/end event.
- *Compromised Users.* This dataset contains 749 records. Each record belongs to a user that is a known information system misuse event taking place. It contains 4 fields: the time (in seconds) in which the event was captured, the user@domain, the source computer and the destination computer. This is the table that will be used for the purposes of testing the prediction models to be developed. There are no missing or invalid values.

It is important to note that from the 12,000+ users in the dataset, there were 98 users identified as compromised, or less than 0.9%. From the 1.058 billion authentication events, only 749 events are compromised.

4 APPROACH

Given the context in which we design and run the experiment, we now describe the architecture selected, as well as the feature engineering process posited.

4.1 Architecture

To use machine learning for the purposes of this experiment, we abstract the activities into three main processes: data transformation into a feature set, Model learning based on training data and decision making using the current observation, which we set at t_0. Please refer to Figure 6 for a task-based architecture of the experimental system. The abstractions enable us to assess each component and the context – spatial and temporal – in which it operates.

The learning stage uses as input the historical data as it creates the baseline profile. The processing demands in this stage are usually higher than those on the decide stage, as it depends directly on the characteristics of the data, the specific AI activity taking place and the parameters selected – especially how long in the past we set the model to analyze. The transformation of the dataset into a

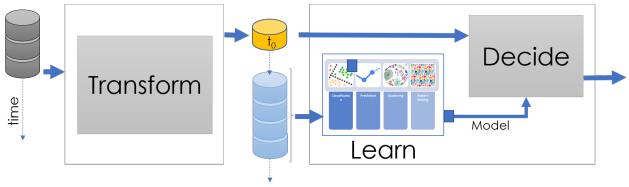


Figure 6: Task-level system architecture.

feature set needs to comply with both qualitative and quantitative principles. The features produced must be fit-for-purpose: they need to contain information that a mathematical model can use to make decisions. They need alignment with the hypothesis space in which the machine learning model is used. If the historical data is not rich enough, it may be difficult to assess if there is anything abnormal about it. In terms of the quantitative requirements, the information produced by the transformation shall be sufficient for the learning data (i.e. training) an effective representation of the phenomenon.

In the architecture designed, raw data observations are transformed into features through the *transform* component for a period T). The domain in which the machine learning will operate – cybersecurity – requires all the processes to be performed rapidly, so the actual outcome (normal or abnormal behavior) is relevant by the time a decision is made by the system.

Once the features are available for analysis, sufficient historical information shall be collected and provided to the learner component. This means that a time window needs to be created (of length W) for the receiving of the features. The learning component will take L time intervals in the training of the model, and the decision is made in an interval D. If the time window for the historical data is too far in the past (large T), current normal behaviors that have not been seen before may be wrongly identified as anomalies. A time window that is too long (large W) will impact the timeliness of the outcome, as well as the normality signature being produced.

In the context of a timely analysis (such as the insider's misuse of an IS), it is critical that the learning stage takes place continuously and efficiently. This may drive a need for significant computational power or a very efficient algorithm that can effectively analyze large datasets in an acceptable time span. If the log files are very large, the learning of the normality baseline may take a long time, rendering

the analysis potentially irrelevant (e.g. an insider's misuse identified too late to take any action). A final requirement that is essential to the smooth flow of the anomaly detection scheme pertains to the continuous functioning of the detection artifact. Given the immediateness requirement placed on the detection procedure, the deciding stage needs to happen in real-time. This also places a burden on the architecture of the system, since it needs to be online the vast majority of the time.

In order to meet the requirements for the insider's misuse detection, we select two technology stacks that provide the versatility and power needed, and leverage the availability of a High Performance Computer environment.

Apache Spark [1] is a general engine for large scale data processing, and contains multiple Application Programming Interfaces (API) in popular languages such as Python, Java and R. Apache Spark takes advantage of computing abstractions and parallelism for the processing of very large datasets. It contains multiple modules that are applicable to large dataset manipulation: SparkSQL for dataset analysis, ML for machine learning, GraphFrames for network graphs manipulation and Spark streaming for the processing of streamed data – which is critical for continuous processing.

The second tool is Knime [5] which is an Integrated Developing Environment (IDE) based on the Java programming language, with multiple nodes available for data mining and machine learning. Knime has been considered one of the leaders in the popular Gartner data science quadrant [4]. Knime is open source and freely available for multiple operating systems, including Microsoft Windows, Apple macOS and various Linux distributions. The large availability of machine learning models enable the researchers rapid prototyping and testing of approaches.

In terms of hardware, the High-Performance Computing (HPC) environment available to researchers at Compute Canada is used. Compute Canada [3] is a consortium of 18 Canadian universities that joined forces to offer researchers in all disciplines large scale computing power, through large computer clusters running Linux distributions. Researchers can use the resources in batch or interactive jobs, depending on the characteristics of the tasks. Please refer to Figure 7 for a technology stack architectural view of the experimental system.

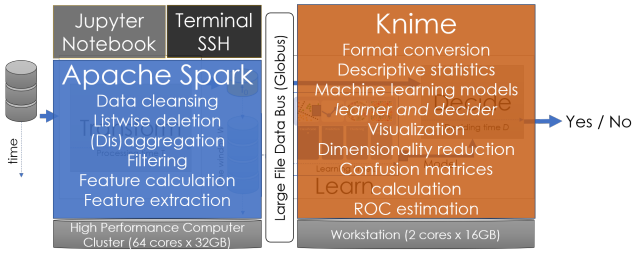


Figure 7: Technology stack experimental architecture

There are four potential data sources in the very large dataset studied: authentications, Domain Name Service (DNS) calls, network flows and processes or programs used. These four datasets provide rich information that can enable the creation an effective feature set. Based on domain knowledge, and with the intention of a parsimonious model, the authentication log is deemed sufficient for the detection of the insider's IS misuse. The rationale is that a compromised user will try to authenticate to different systems, or in an uncharacteristic way. By analyzing the variance in the authentication, a machine learning model shall enable detection of potential IS misuse.

The authentication data use for this research exists as events in sequence on a per-second basis. Understanding the context of using IS for the delivery of work in organizations, we make the determination that the user behavior data may be analyzed at the daily level. Regular behavior of IS users present a daily pattern that can be used for the purposes of finding what 'normality' means. Thus, we convert the seconds to their equivalent of a date (we select the first second to start on January 1, 2018 for simplicity).

With the daily data, we determine the weekdays and weekends. This information is not originally labeled in the dataset, and it is essential to the analysis of the variance. We remove the data pertaining to the weekends as it may create noise when compared with regular work days with higher authentication volume.

We proceed to the determination of missing and invalid data. Some of the authentication data is incomplete, represented by the '?' sign. Upon inspection of the data and reviewing the sources, we assess that the data follows a Missing Completely At Random (MCAR) pattern. These records are deemed to be an ignorable missingness situation [18], and therefore a *listwise* deletion method is employed, with the complete row being deleted if any of the variables had missing data. The resulting dataset has 583,350,822 records. We further filter the dataset to include only values for user authentications, ignoring the computer authentications, as our focus is on characterizing user behaviors. This means selecting in the user field values starting with the letter 'U'. Thus, we transform 1.051 billion records into a significantly more manageable 325,771 observations, each corresponding to a user authentication event taking place.

The data is now suitable for the feature construction.

4.2 Feature engineering

The ultimate objective of the experiment – identification of insider's threats – as well as the structure and nature of the data guide the feature engineering activities.

The data has been pre-processed to ensure we have valid observations that contain the knowledge we seek. We have data points that are directly comparable to one another since we defined the time unit of comparison to be a workday. The features we conceptualize need to exist in the hypothesis space we face. Thus, we need to find features that represent the user behaviors when abstracted to the time unit chosen.

The question we need to pose now is how to characterize what a user does in a day. We can further specify a user's behavior by interpreting the data as a sequence of events. Analyzing the authentications data, we can see that the information provides multiple value-pairs. One authentication event contains the source and destination computers, source and destination domains and source and destination users. This implies that the information is a flow record between two nodes. We observe that for the same user, any given node may appear as source or destination at different times.

In fact, when aggregating the multiple records for a specific user, it is possible to identify sequences of flows as the user authenticates from a node to a second one, and then to a third or more. This is the equivalent of a graph.

In its most basic form, graph analysis – sometimes referred to as network analysis – is an extension in the interpreting of relationships between entities. This is better conceptualized with an example: social networks. One of the most powerful applications of graph analysis is typified by the search for influencers in groups of individuals. In social networks such as Facebook or Twitter, each user is connected in one or multiple ways to several other users or entities. Colloquially, being a 'friend' on Facebook means that you are somewhat connected to your friends' friends. This dynamics create a complex network that has valuable information about users and their IS behaviors. Figure 8 illustrates the sequences that any given user may perform in a day. A user may log into computer 3 in the morning, and then authenticate to computers 1, 2 and 4 afterwards. Since many different users may use the same computers – such as authenticating to their email server – we need a way to differentiate the different sequences (i.e. graphs) for each user. We can use the concept of attributes in the graph to maintain a virtual separation between the graphs of every user. We define a vertice in the graph as a node with three attributes: the user, the computer and the date. In terms of the edges, we can also articulate attributes that are relevant to the authentication event, such as its success or failure.

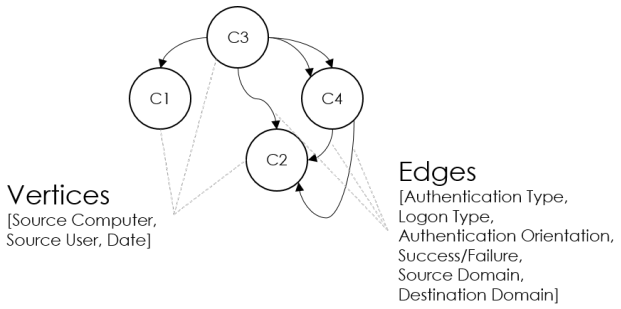


Figure 8: Simple feature graph.

The depiction shows a typical path that a user may take when using IS in the organization. She may turn on the computer, log into her first system of the day (computer three), and as the day progresses she will access other computers in the network. Sometimes she will log into the other computers directly, but most probably she will authenticate in a consistent sequence. At the end of the day the user will have virtually created an 'authentication path' of her IS activity throughout the day. Although any user's graph may differ from day to day, it is human nature to perform the same tasks in the same sequence consistently. It is also logical to assume that different users will follow different graphs given their job duties, preferences or efficiency in completing tasks. Given our focus on user behaviors and our domain knowledge, authentication graphs constrained by day can explain well the behavior of any given user. As explained previously, only weekdays will be used given the high probability of significant differences between user's activities on a weekday vs. a weekend. Both edges and vertex have attributes that can further enrich the behavior definitions. Each edge may be a successful or a failed authentication event, or be performed in the morning or afternoon. Conversely, a vertice can contain the attributes of certain computers, such as being a server or a workstation.

We proceed to the formulation and estimation of the first set of features, as it is depicted in Figure 9. The data is aggregated at the daily level, enabling the analysis of normative events in a given date.

Feature 1: NmbDestinationUsers. On a relatively complex information landscape, there will be multiple inter-connected systems that may or may not share a common set of authentication credentials. It is quite normal that the architecture requires users to log into different systems with different usernames or passwords. Thus, we define this indicator variable as capturing how many different user names any specific user has logged in as, on any system, in a day. A typical user will likely use only one user name (its own), with multiple users potentially signaling an abnormal pattern. However, it is important to recognize that some users – specially power users or system administrators – may have several user names. The comparison shall be performed to a set of peers, or with the user himself.

Feature 2: NmbSourceComputers. This variable captures how many different computers the user has logged from. A typical user will use only her own computer.

Feature 3: NmbDestinationComputers. This variable captures how many different computers the user has logged into. This is especially applicable to programs that run out of different servers (like web systems). Every time a user logs into the website, it is logged into the computer servers that host the application. A potential insider's threat may be marked by an unusual number of destination computers as the threat agent tries to access or misuse other systems.

Feature 4: NmbAuthenticationTypes. This variable represents the number of different authentication types that are being used by the user. This number shall be consistently small as the type of authentication shall not vary much during the regular use of an IS.

Feature 5: NmbSuccessOrFailure. With this feature we articulate whether the user has had only success events in its daily login activities (which can be considered normal) or if she has had failed events in any given day, signaling a potential threat.

Feature 6: NmbSourceUserDomain: A *domain* can be described as a defining set of objects that any given user interacts with or has access to. It is common that a user is in a certain domain as the source, but authenticates to a system that resides in a different domain. This variable captures the number of originating domains that the user is attached to when performing the login action.

Feature 7: NmbDestinationDomain. This variable captures the destination domain for the system that the user is authenticating into.

The above seven features correspond to the baseline markers of user activities that may aid in the determination of abnormal

Second	Source User @ Domain	Destination User @ Domain	Source Computer	Destination Computer	Authentication Type	Logon Type	Authentication Orientation	Success / Failure
1	U101 @ Dom1	U101 @ Dom1	C101	C102	Negotiate	User	Logon	Success

Date	User	Nmb Destination Users	Nmb Source Computers	Nmb Destination Computers	Nmb Authentication Types	Success / Failures	Nmb Source Domains	Nmb Destination Domains
3-Jan-2018	U101	2	4	2	1	5	1	1

Figure 9: Processing of the data into basic features: original record in the dataset, and its transformation into basic features for user U101

behavior. In addition to this, it is important to perform feature engineering to conceptualize variables that are a result of a chain of events.

These networks can be immensely complex, given the fluid and varying nature of relationships between entities. Furthermore, the computing power needed to manage and query graphs is significant, driving the need to use Big Data tools for its analysis. The process depiction is in Figure 10.

Leveraging graph analysis concepts, we construct the following features for the analysis of user's behavior:

Feature 8: cnt_comp_degrees. In a graph network, the degrees for a specific vertice (a computer in this research) captures the number of edges (authentication events) that depart or end in it. With these feature, we capture how many computers have participated in authentication events (i.e. have edges attached to it) in any given day by the user.

Feature 9: cnt_comp_indegrees. In graph analysis, in-degrees pertain to the number of destination edges attached to a vertex. With this feature, we capture how many computers any given user, on any weekday has with edges terminating in it. In other words, it is the number of destination computers that follow the authentication path for the user.

Feature 10: cnt_comp_outdegrees. In graph analysis, the number of outdegrees per vertice corresponds to the number of edges that depart for any given computer in an authentication path. In the language we have used in this study, we can conceptualize this feature as the number of source computers used by any given user on any given day.

Feature 11: min_degrees. This feature captures the minimum number of edges that any given vertex has in the authentication graph. In other words, the minimum number of authentication events in any given computer used by the user in the day.

Feature 12: max_degrees. This variable captures the maximum number of authentication events that any computer used by the user had on that day.

Feature 13: avg_degrees. This feature captures the average number of authentication events the computers used by the user had in the weekday.

Feature 14: sum_degrees. This feature aggregates all the authentication events that took place when the user was in the IS on a given day.

Feature 15: min_indegrees. This value captures the minimum number of events pertaining to a computer being the destination of the authentication.

Feature 16: max_indegrees. This is the maximum number of events for any given computer being the source node for the authentication.

We complete the feature set with calculations on the average, sum, min and max for the indegrees and outdegrees of the authentication graph.

We have now a set of features that can be used for the detection of the insider's misuse. Although the focus of this study is not the detection process, we describe at a high level how the transformed feature dataset is used.

If the architecture of the detection system uses supervised learning, each of the records (with composite primary key being day and user) shall be labeled as a normal or anomalous record. In the particular case of this experiment, we do have access to the users that were compromised, as well as the days in which the misuse event took place. Using this information, all the records in the feature dataset are labeled appropriately. This labeled feature set can then be split in training and test partitions, and used in a computational model such as logistic regression or neural networks. The model learns from the training set, and with the test set we can assess how good the classification exercise performed. We perform this process in our system since our focus is on the feature engineering, so our intent is to demonstrate that the feature set does allow a computational model to predict better than a random draw.

For most applied cases, it is rather improbable that the historical data is labeled. If this is the case, unsupervised learning is required. Multiple computational methods can be used such as multivariate

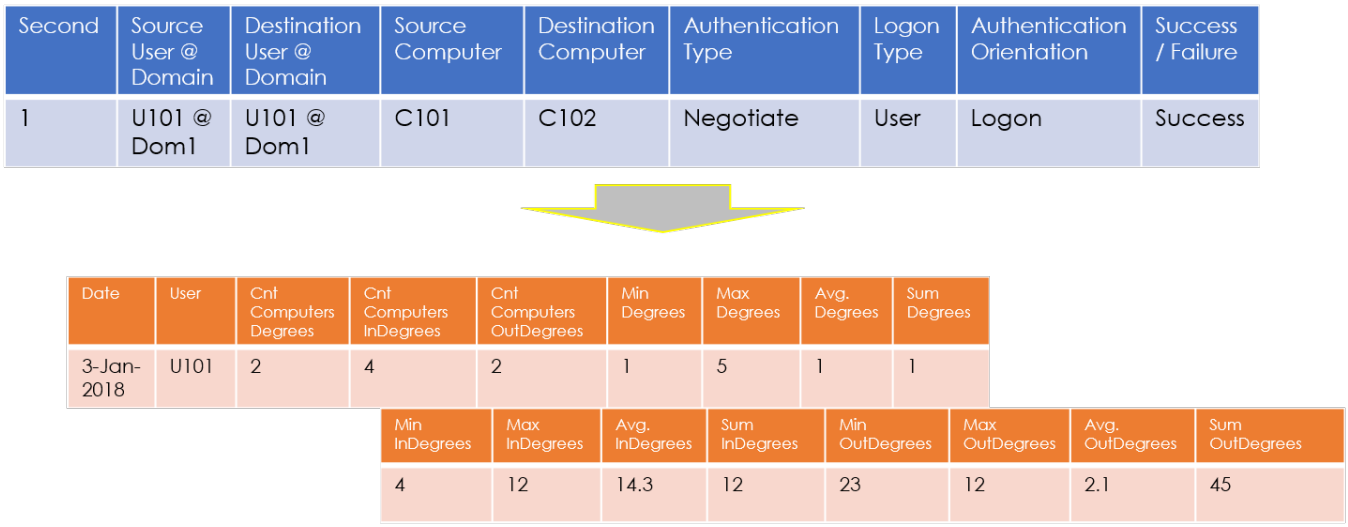


Figure 10: Processing of the data into graph features.

regressions, outliers analysis, pattern mining and Principal Component Analysis (PCA). These methods are out of scope of this experiment.

5 EXPERIMENTATION

We now proceed to use the feature set for the detection of the threat in a supervised computational model. Our research question pertains to the ability of detecting a user as compromised given the set of features extracted. Each record in the feature set represents a user's behavior on any given day. More specifically, we have spatial features for every user in each day. We preform one last transformation to obtain one record per user and capturing all the markers of the behavior. Since we have 42 weekdays in the data, we can have one variable per weekday per spatial feature. The process transposes the feature dataset from the original matrix of 325,771 events x 22 spatial features to a 11,255 users x 924 spatio-temporal features. Please refer to Figure 11.

We add one last field to the feature set: "whether the user was compromised or not". This information is available in the source dataset. We proceed to split the feature set in a training and a test partition. We train a logistic regression model with the training data and test the results with the test data.

As it was explained in the background section, the relationship between specificity and sensitivity may be displayed using a Receiver Operating Characteristic (ROC) curve. The ROC plots the sensitivity vs. the (1-specificity) of a classifier. The Area Under the Curve (AUC) in a ROC curve ranges from 0 (completely deficient) to 1 (perfect classifier) [7].

The classification exercise is inherently a supervised activity. This means that the data shall have the class for each observation, so the machine learning model can learn what are the characteristics that an anomalous observation possesses. From this perspective, it is not a true anomaly detection mechanism, but rather a way to test that the feature set used is valid and represents the behaviors exhibited by any given user. In a typical classification exercise, the historical data is divided in a training and a testing dataset. The training set is used to find the parameters of the model and the test set is used to evaluate how well the model performs the classification. From a conceptual perspective, the classification is based on a known signature of what constitutes an abnormal behavior. A signature-based model has theoretically less false positives, but may have significant challenges in identifying insider's misuse of IS when the attacker uses a new, unexpected behavior for which there is no signature. For the purposes of a supervised classification exercise, the data is divided in training (80%) and testing (20%) using stratified sampling to ensure the few compromised user observations are properly distributed between the two sets.

Logistic Regression. the classic parametric classifier uses logistic regression as its theoretical underpinning. It is related with the process of linear regression, but instead of predicting a value, the dependent variable is a function of it named the logit [7].

We use logistic regression to process the features captured for each of the users. Logistic regression is especially sensitive to colinearity, so a correlation filter is applied prior to the training of the model. Using the model we obtain the ROC curve depicted in Figure 12. The AUC is 82.79%, with this classifier performing significantly better than the random draw.

6 DISCUSSION

The increasing availability of large datasets have led to an explosion of machine learning applications in a multitude of domains. The phenomena of the insider's threat in information systems is one of the areas in which these technologies can create significant value, since the user behaviors are captured in large log files. However, before this data can be effectively used for this purpose, the feature engineering process is critical for the articulation of relevant variables that can enable a successful analysis.

In this study we presented a set of features that capture the user behavior, and that can be analyzed for the purposes of detecting information systems' misuse. We leverage the existence of a very

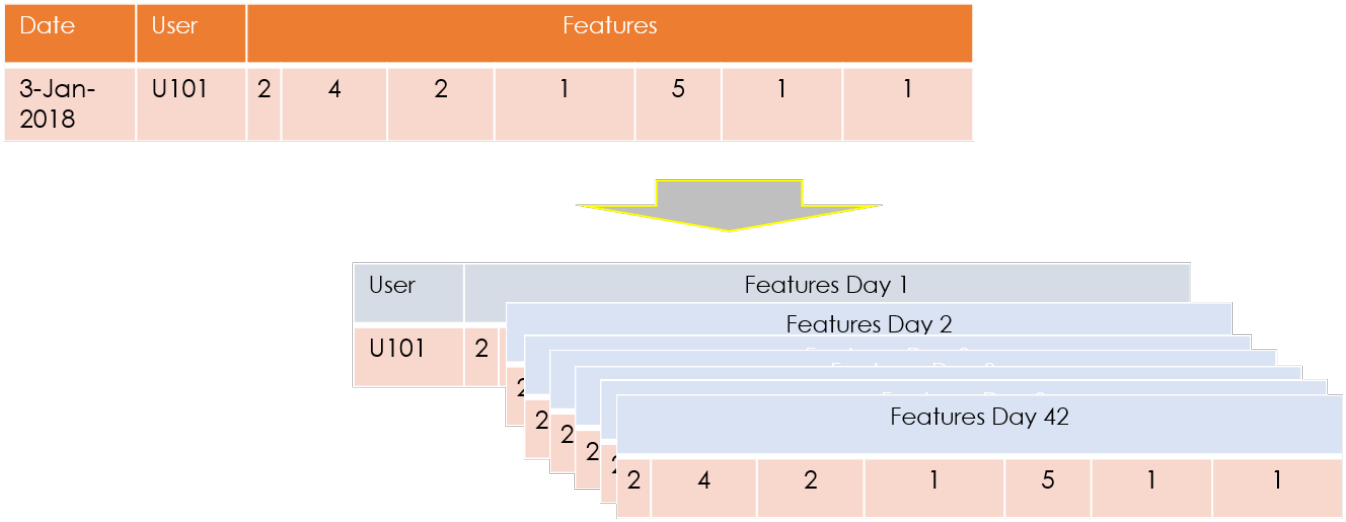


Figure 11: Final transformation of spatial features into spatio-temporal features.

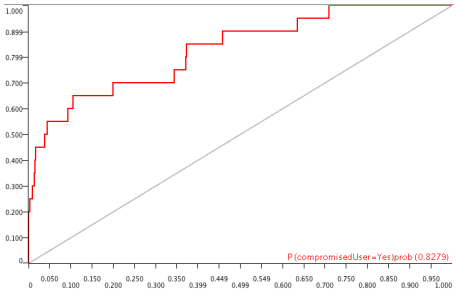


Figure 12: ROC curve for the logistic regression on the large dataset from Los Alamos laboratory.

large, anonymized dataset to experiment with the approach we posit. We cleanse and aggregate the authentication events data, and from the resulting dataset we extract the features we believe are meaningful markers of user behavior in an IS.

The final stage of our experiment is to prove that the features selected do contain the information that would enable the detection of insider's threat. We do this through the use of a logistic regression classifier since the large dataset contains ground truth – labeled data that can be used to train the classifier.

Through the ROC curve displayed on Figure 12 we confirm that the features permit the detection of IS misuse: the AUC is 82% when compared with a random draw of 50%, significantly improving the results when compared to a random selection when looking for anomalous behavior.

It is important to note that the logistic classifier is used in this experiment since the focus is on the feature engineering aspects and not on the detection process. The logistic classifier is simple to implement and provides unambiguous results. However, in real world applications it is unlikely that such a system would be used since the existence of labeled data representing all potential user behaviors is rather improbable. An insider's threat detection system will likely need an anomaly detection approach in which the anomalous signature is continuously updated to be able to react to new threats.

A follow-up experiment to the one presented may consist of using multiple detection methods departing from the features extracted. The focus would be on the detection process in contrast with this experiment that concentrates on the feature engineering aspects. In addition to this, the use of deep learning on the features extracted may be a good complementary work to explore since the identification of patterns in the data is a strength of this technology.

The importance of domain knowledge in the conceptualization and calculation of features cannot be overstated. Features that can be efficiently and effectively used for the insider's IS misuse detection require strong familiarity with the area of application, and a rigorous methodology that produces information – and ultimately knowledge – from data.

REFERENCES

[1] [n. d.]. Apache Spark™ - Unified Analytics Engine for Big Data. https://spark.apache.org/
[2] [n. d.]. Los Alamos National Lab: National Security Science. https://www.lanl.gov/
[3] 2018. Compute Canada | Calcul Canada. https://www.computecanada.ca/
[4] 2018. Gartner Recognizes KNIME as a Leader in Data Science and Machine Learning Platforms | KNIME. https://www.knime.com/about/news/gartner-recognizes-knime-as-leader-in-data-science-and-machine-learning-platforms
[5] 2018. KNIME - Open for Innovation. https://www.knime.com/
[6] Alexander D. Kent. 2015. Comprehensive, Multi-Source Cyber-Security Events. https://doi.org/10.17021/1179829
[7] Peter C Bruce, Galit Shmueli, and Nitin Patel. 2014. *Data Mining for Business Analytics*. Vol. 9781461476. 1–166 pages. https://doi.org/10.1007/978-1-4614-7669-6
[8] Pedro Domingos. 2012. A few useful things to know about machine learning. *Commun. ACM* 55, 10 (2012), 78. https://doi.org/10.1145/2347736.2347755 arXiv:cs/9605103
[9] Guozhu Dong and Huan Liu. 2018. Feature engineering for machine learning and data analytics. c (2018).
[10] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17* (2017), 1285–1298. https://doi.org/10.1145/3133956.3134015

[11] Song Feng, Ritwik Banerjee, and Yejin Choi. 2012. *Syntactic Stylometry for Deception Detection.* Technical Report. 8–14 pages. www.tripadviser.com,
[12] Isabelle. Guyon. 2006. *Feature Extraction.* Vol. 207. 778 pages. https://doi.org/10.1007/978-3-540-35488-8
[13] Heiko Hoffmann. [n. d.]. *Kernel PCA for Novelty Detection.* Technical Report. http://heikohoffmann.de/documents/hoffmann{_}kpca{_}preprint.pdf
[14] Thomas Hofmann. 1999. *Probabilistic Latent Semantic Indexing.* Technical Report. http://cis.csuohio.edu/{~}sschung/CIS660/PLSIHoffman.pdf
[15] Yann Lecun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (may 2015), 436–444. https://doi.org/10.1038/nature14539 arXiv:arXiv:1312.6184v5
[16] Ron Lieber. 2017. How to Protect Yourself After the Equifax Breach. , 10 pages. https://www.nytimes.com/interactive/2017/your-money/equifax-data-breach-credit.html
[17] H P Luhn. [n. d.]. *The Automatic Creation of Literature Abstracts*.* Technical Report. http://www.di.ubi.pt/{~}jjpaulo/competence/general/{%}281958{%}29Luhn.pdf
[18] Lawrence S. Meyers, Glenn Gamst, and A. J. Guarino. [n. d.]. *Applied multivariate research : design and interpretation.* 978 pages. https://us.sagepub.com/en-us/nam/applied-multivariate-research/book246895
[19] Nicole Pelroth. 2017. All 3 Billion Yahoo Accounts Were Affected by 2013 Attack - The New York Times. https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html?{_}r=0
[20] Animesh Patcha and Jung Min Park. 2007. *An overview of anomaly detection techniques: Existing solutions and latest technological trends.* Technical Report 12. MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING. 3448–3470 pages. https://doi.org/10.1016/j.comnet.2007.02.001
[21] Why Read and This Report. 2016. Hunting Insider Threats. (2016).
[22] Ron Lieber and Stacy Cowley. 2017. Trying to Stem Fallout From Breach, Equifax Replaces C.E.O. - The New York Times. https://www.nytimes.com/2017/09/26/business/equifax-ceo.html
[23] Scott Shane and Andrew Lehren. 2010. Leaked Cables Offer Raw Look at U.S. Diplomacy. https://www.nytimes.com/2010/11/29/world/29cables.html
[24] Steven Erlanger. 2016. Edward Snowden Says Disclosures Bolstered Individual Privacy - The New York Times. https://www.nytimes.com/2016/09/17/world/europe/edward-snowden-defending-his-patriotism-says-disclosures-helped-privacy.html
[25] Bhavani Thuraisingham. 2018. *Big Data Analytics with Applications in Insider Threat Detection.*
[26] Bimal Viswanath, M Ahmad Bashir, Max Planck, Software Systems, Mark Crovella, Saikat Guha, Krishna P Gummadi, and Alan Mislove. 2014. Towards Detecting Anomalous User Behavior in Online Social Networks. In *the 23rd USENIX Security Symposium.* 223–238. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/viswanath
[27] Merrill Warkentin and Robert Willison. 2009. Behavioral and policy issues in information systems security : the insider threat. *Information Journal of Information Systems* 18, 2 (2009), 101–105. https://doi.org/10.1057/ejis.2009.12