**Security Product to Identify High Risk User Activity When Interacting With Managed Hosts, Business Applications And Data**

**InsDetek**

Noah Asing, Diru Jia, Wenqi Kou, Tianxiao Gaoqu

# Executive Summary

Detecting malicious insiders is a critical cybersecurity issue. Unfortunately, insider threat detection remains elusive due to high reliance on manual inspection; algorithms often yield enormous amounts of false positives, overwhelming human investigators. An ideal detection system would be tolerant to low investigation budgets to maximize the security team’s effectiveness. The principal interest of this research is to apply supervised learning models on unlabeled datasets by combining unsupervised learning with manual tagging to produce labeled data. In doing so, our insider threat management procedure identifies suspicious users and prioritizes investigation resources to minimize reliance on human teams while maintaining high detection rates.

Our system bisects input data and ranks the first segment of users by suspicion using an unsupervised anomaly detection algorithm. The most anomalous user activity is then passed to a human team for investigation and manual labeling. First segment users in lower suspicion percentiles default to a benign label, and this newly labeled data then trains a supervised classifier., which in turn predicts labels for the second segment of users.

Our process enables high-accuracy supervised classification models to be applied on unlabeled data, and prioritizes the investigation budget to avoid wasted time and energy. Utilizing the r4.2 release of the CMU-CERT dataset, we used an Autoencoder for unsupervised anomaly detection and compared Random Forest, XGBoost and LSTM methods for our supervised classification. Implementing sliding window data augmentation and greater-than-unity weights for the loss function improved recall rates from 27% to 37%. Increasing model complexity and depth did not affect performance.

Future research should include this system’s implementation on real-world, non-synthetic datasets, comparing performance across granularity levels, determining the effects of individualized models separated by employee departments, and increasing classification categories to identify specific malicious scenarios.

# INTRODUCTION

## 1. Why are insider threats critical?

Insider threats, or cyber security weaknesses involving personnel within the organization, are critical risks across all industries. These weaknesses are a major issue, and contribute roughly 50% of breaches logged in the VERIS database, a cybersecurity breach reporting database managed by Verizon (Bailey et. al. 2020, 4). Furthermore, intimate systems knowledge and internal privileges leads to highly damaging breaches. In 2019, a data breach at Capital One compromised social security numbers, credit card application information, or bank account information for over 100 million people (Flitter and Weise 2019). Three years previously, an executive at Waymo, Google’s self-driving research team, downloaded 14,000 company files, including numerous trade secrets and proprietary designs, before leaving the company and joining Uber’s self-driving division (Isaac 2019). The ensuing Waymo-Uber civil suit settled for roughly $245 million in Uber stock (Sage, Levine and Somerville 2018). Thus, the cybersecurity community and potential client organizations are highly interested in Insider Threat Detection (ITD).

Monitoring and managing insider threats is difficult, as insiders are already within the system. Furthermore, it is difficult to algorithmically model the unpredictability of human behavior, so programs either have high false positive rates (FPR) or fail to detect anything at all (Moore et. al. 2015, 4-17). When flooded with false alarms, the human security teams are overwhelmed and true positives are missed; plus, persistent investigation leads to distrust between employees and the organization (Moore et. al. 2015, 4-17).

## 2. Solution proposal

Effective ITD must balance between a human security team and the detection algorithm, so that maximal data can be analyzed without increasing the required human investigation budget (IB). An ideal detection algorithm would also learn from the human security team, so it may better analyze human behaviors. This proposed model will strive for high accuracy while allowing a lean IB.

Our team presents a semi-supervised ML model with three primary components. First, an unsupervised learning model ranks the suspiciousness of a segment of the security logs, based on anomaly detection (AD) principles. The most suspicious activity (cutoff determined by IB) is validated and labeled by a human team. Any data beyond the IB coverage receives a default benign activity label. This labeled data then is used to train a supervised learning model that classifies the remaining logs. The semi-supervised system enables analysis for large amounts of data while maintaining minimal IB, and introduces human screening to avoid a high-FPR algorithm.

# EXISTING SOLUTIONS

## 1. Related work

AD is one of the most popular topics in cybersecurity. Initial ML research for AD used the Isolation Forest (Liu et al. 2008) algorithm. It differs from classic methods like Principal Components Analysis in AD (Chalapathy et. al. 2017) by identifying easily separated points, instead of striving for normalization and is an efficient unsupervised model for continuous numerical data. Therefore, an isolation forest algorithm was used for the unsupervised learning baseline. Widely used supervised models include Random Forest (Guha and Sudipto 2016) and Support Vector Machines, which identify clusters and establish boundaries to differentiate normal data. As a simple and robust model, Random Forest was used for the supervised learning baseline.

Recent works use deep learning to accommodate increasing amounts of data, including generative methods like AutoEncoder (AE) (Zhou, Chong and Randy 2017) and Generative Adversarial Network (Zenati et al., 2018). Unlike classic methods, these do not require explicit feature constructions. Initial experiments showed that AE outperformed other unsupervised models, so AE will be a primary component of the unsupervised portion. Improved versions of these models include the Variational AE and AE-based Deep Neural Network models for dimensionality reduction. Regarding deep neural networks with time series data, Recurrent Networks like GRU and LSTM (Malhotra et al., 2016) yield great performance. In the technical research phase, LSTM-based convolutional neural networks and simple prediction LSTM models will be tested.

## 2. Limitations of current methods

Most companies utilize rules-based user-behavior monitoring software for ITD. There are four typical approaches: Prevention and monitoring, Event detection, Investigation and HR/business-unit action (Bailey et. al. 2020, 4). These approaches depend on human labor, meaning large IB for individual, manual screening. Moreover, subjectivity typically elevates the FPR.

In academia, ITD works fall under Machine Learning (ML) systems and AD approaches (Le & Zincir-Heywood, 2020). Early ML systems for ITD included Bayesian networks (Caputo et al. 2009) and graph-based algorithms (Eberle et al. 2010), but the graphed approaches suffered from irregular structures in big data. As the field of deep learning matured, more ML approaches were introduced. Since the action log is non-stationary, where user activities are time-variant, recurrent neural networks (RNN) and long short-term memory (LSTM) can be applied to the sequential data and generate anomaly scores by users (Yuan and Wu 2021). However, the generalization of ML for ITD is constrained by the available training sets.

## 3. Opportunities

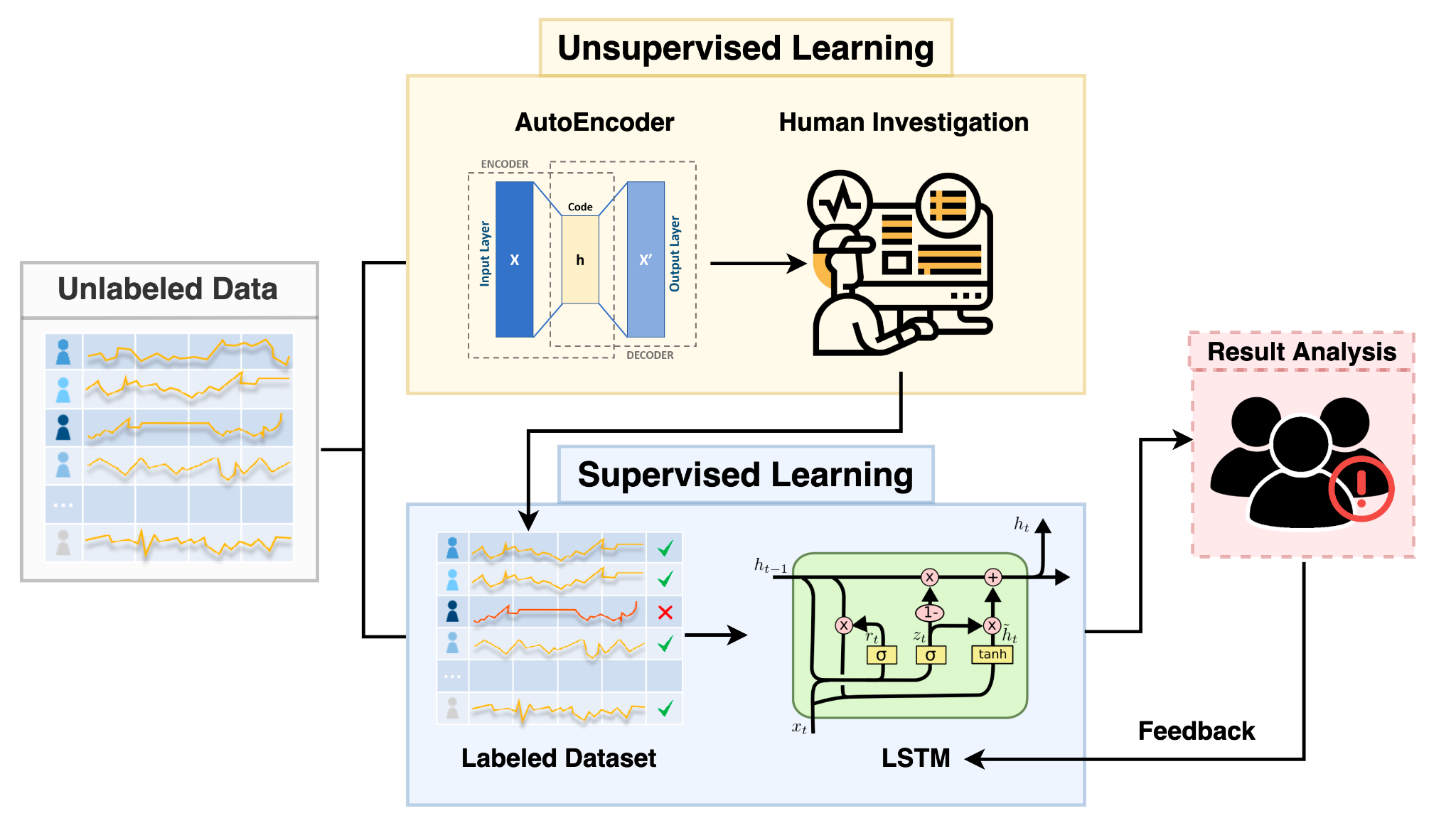
Inability to prioritize IB means current ITD methods waste IB on false positives, or have poor detection rates and data explosions. One opportunity for improving ITD lies in applying the AD methods, to isolate cases with higher probability of being malicious. In this case, the IB can be saved by prioritizing the behaviors with higher risk.

Current methods also suffer from data explosions, since datasets often consist of session-based action logs. Thus, ML methods should be used to generalize the detection mechanism, and reduce reliance on IB. Unfortunately, one setback is the limited availability to label cybersecurity datasets for model training. One potential solution is to annotate unlabeled datasets manually and optimize this process with AD. After creating the labeled dataset, traditional ML and deep learning methods can be applied to improve the detection rates and generalize the model. Combining unsupervised AD methods and supervised algorithms establishes an end-to-end process for ITD.

# METHODS AND PROPOSAL

## 1. System overview

The principal interest of this research is to apply supervised learning models on unlabeled datasets by combining unsupervised learning with manual tagging to produce labeled data. This labeled data trains a supervised learning model, which can achieve a higher detection rate. Lastly, the task of manual tagging must be prioritized in order to minimize demand on human labor.



**Figure 1**. Top-level overview of proposed ITD procedure with an LSTM classifier.

The proposed system first divides the dataset into two segments as shown in Figure 1. Since malicious activity contains atypical behavior compared to benign activity, the first data segment is processed by an unsupervised model performing AD to rank user activity by suspiciousness. For the unsupervised learning models, the system employs an AutoEncoder. Next, based on the IB, the upper percentile of the ranked data is manually labeled and the remaining ranked data defaults as benign activity. Thus, this first part of the data is now labeled. Supervised learning models are trained on the labeled data segment and applied to the remaining half of the dataset. The supervised learning algorithms employed and compared include Random Forest, XGBoost and LSTM. Merging results from manual screening and the supervised learning predictions yields scores for the whole dataset.

## 2. Data collection and preparation

### 2.1 Dataset Overview

In the field of ITD, there are few comprehensive real-world datasets with labels, because the user activity logs often contain corporate intellectual property or other protected information. Currently, many ITD studies use a dataset released in a joint effort by Carnegie Mellon University and the US Computer Emergency Readiness Team (CMU-CERT dataset) to test the performance of their proposed models. Due to its widespread usage as a standard, we also plan to use the CMU-CERT dataset in order to verify the feasibility and effectiveness of our proposed system.

The CMU-CERT dataset is a synthetic dataset that is maintained by the Software Engineering Institute of Carnegie Mellon University (Glasser and Lindauer 2013). This dataset is generated from algorithms trained with over 1000 real insider threat cases (Glasser and Lindauer 2013). There are currently 8 versions of data in this dataset, depending on the version of the generator (Glasser and Lindauer 2013). Each iteration contains log data for simulated users across simulated time ranging from January 2010 to June 2011 (Glasser and Lindauer 2013).

For each version of the data, there are five log files that record user interactions with the computer. In detail, the **logon.csv** records each user's login and logout patterns, **email.csv** records email address endpoints, content and attachments, **http.csv** contains web browsing information, **file.csv** contains the interaction with files (open, read, write), and **device.csv** records the device (PC, mobile) used in a certain action. In addition, there is user profile information and enterprise structure data, including the time a user joined the organization, their department, or supervisor's name.

The enormous amount of data in the CERT dataset (more than 10 million data entries in r4.2 release alone) and the scattered data tables present significant difficulties to the work of data processing and feature engineering. The following section describes the feature extraction process.

### 2.2 Feature Extraction

#### 2.2.1 Feature Aggregation

Our team constructs features grouped by user behavior and the context in which the behavior occurs. The constructed features contain three main categories of information: (1) the device used, (2) the time, and (3) the numerical and statistical characteristics of the specific behavior. By combining these three types of information, we obtained a total of more than 600 features that can describe the user's behavior.

For the first type of information, the device used, our system obtains information about the specified user's supervisor based on the enterprise architecture data. In addition, we define PCs that are used by more than 5 people as shared PCs. Also, we filter out users whose job title is IT to prevent the impact of such outliers on model training, as these users utilize large numbers of devices regularly. For the second type of information, our system classifies the time when the action occurred into three categories: weekday office hours, weekday off hours, and weekends.

The third type of information is extracted from the five log files, which can be divided into two categories: frequency features and statistical features according to the calculation method. Frequency features count the number of times a certain user behavior occurs within a certain time frame, such as the number of files read, or the number of websites viewed. Statistical features summarize the mean, variance or median of a user's behavior; these features include numbers such as the average file size, or the average number of words in the file. In addition, for features extracted from file and http logs, we classify activities by website content and file attributes. Websites are categorized into groups such as social media, cloud, and job seeking, while files are grouped by filetype, or extension.

#### 2.2.2 Temporal Information

The feature extraction above is based on the snapshot of user's behavior within a time window, and does not consider the variation of user's behavior over a period of time. However, the abnormal behavior of users is often manifested as outliers in a time series. Thus, we consider adding time series information to the features.

Insiders usually behave normally for a long period of time before an abnormal behavior occurs. So we try to find the turning point of the abnormal behavior of insiders. Our approach compares the degree of bias between the user's current behavior data and the behavior over a recent period of time. Specifically, we have tried three methods for calculating deviations, namely calculating the mean, median and variance of the statistical features of a user's behavior within a period of time, and calculating the difference between the statistical value of the user's current behavior and the above three values. The length of the time window is also an adjustable parameter. We chose one month, that is, thirty days, as the time window in the following experiments.

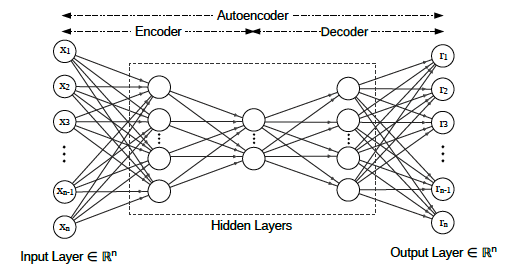
#### 2.2.3 Data Granularity

In order to calculate the frequency and statistical features, we need to determine a time duration. We choose three different lengths of time windows: week, day and session, where session represents the time interval between a user logging in and logging out. We aggregated the values of the frequency and statistical features for these three data granularities. By combining the above three types of information, we have constructed features that contain user activity information with context.

## 3. Description of ML algorithms

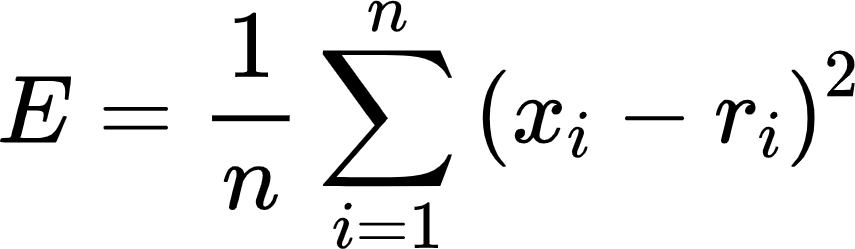
### 3.1 AutoEncoder

An AutoEncoder is an unsupervised learning model consisting of two main elements, an encoder and decoder (Hinton and Salakhutdinov 2006). Figure 2 shows an example of an AE model with 3 hidden layers. During the encoder phase, the model discovers a compressed representation of the given data, while the decoder aims to reconstruct the original input (Hinton and Salakhutdinov 2006). During training, the decoder learns to select the most informative features and store them in the compressed hidden layers (Hinton and Salakhutdinov 2006). AE takes the data itself as the training target and restores the compressed data to the original data to the maximum extent (Hinton and Salakhutdinov 2006).

**

**Figure 2**. Network architecture of a general AE model (Le et. al. 2020).

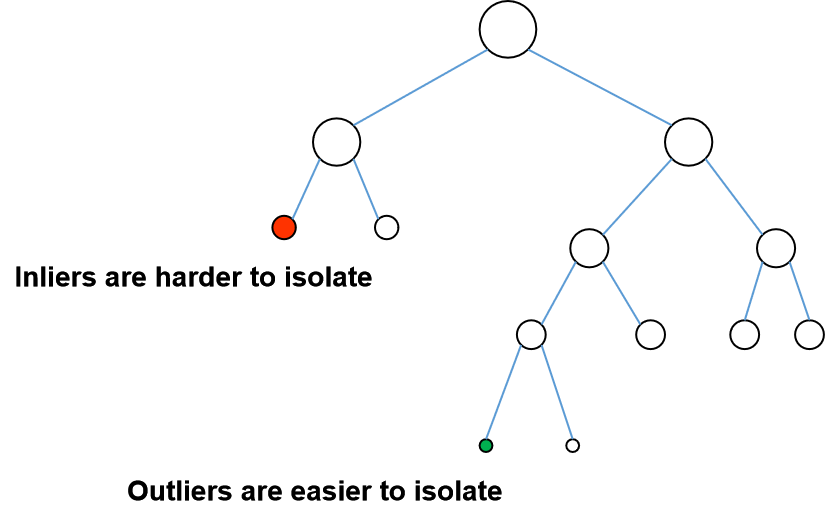
In the unsupervised AD case, there is no outlier sample to learn from, so the algorithm assumes that outliers and normal samples follow different distributions. For normal data, an AE is able to restore data well, but for anomalous data, due to atypical distributions, the AE cannot restore these data points well. Those outliers may lead to large reconstruction errors. Using Mean Squared Error to measure the reconstruction performance, based on the model and parameters shown in Figure 2, error can be represented by:



When the reconstruction error is greater than a certain threshold, the corresponding data point will be marked as an outlier and the corresponding data point is ranked as anomalous.

### 3.2 Isolation Forest

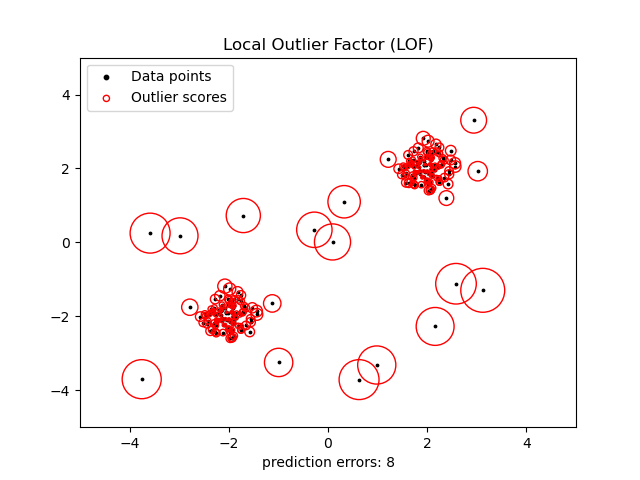
An important part of anomaly detection is to describe the "difference" of abnormal data. The most direct way is to use a variety of statistical, distance and density quantitative indicators to describe the degree of separation between data samples and other samples. However, Isolation Forest (Liu et al. 2011) has a more ingenious idea, which attempts to directly depict the degree of "alienation" of data without resorting to other quantitative indicators. Isolation Forest has a good reputation both in academia and industry for its simplicity and efficiency.



**Figure 3**. An example isolation tree.

### 3.3 Local Outlier Factor

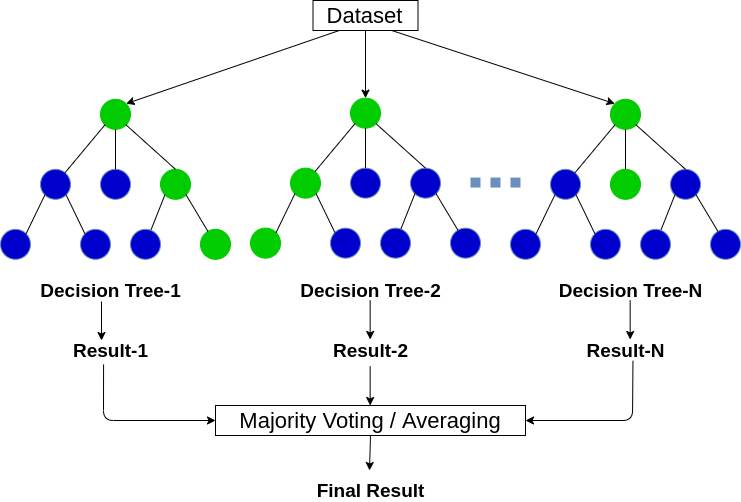
Local Outlier Factor (LOF) is a classical algorithm based on density (Breuning et al. 2000). Anomaly detection algorithms prior to LOF were mostly based on statistical methods or borrowed clustering algorithms for outlier identification (e.g., DBSCAN, OPTICS). Statistical anomaly detection algorithms usually assume that the data obeys a specific probability distribution, which may be invalid. On the other hand, clustering methods can only give a binary judgment (i.e. whether it is an outlier or not), and cannot quantify the degree of anomaly of each data point. In comparison, the density-based LOF algorithm is simpler and more intuitive. It doesn't require much distribution, and quantifies abnormality of each data point.



**Figure 4**. Visualized results of LOF.

### 3.4 Random Forest

Random Forest is a bagging (Bootstrap Aggregation) approach, which is an ensemble learning algorithm based on decision trees (Breiman 2001). Decision trees are a simple explanative algorithm and conforms to human intuitive thinking. In other words, a decision tree is a supervised learning algorithm based on if-then-else rules. Random Forest is composed of many decision trees, and there is no correlation between different decision trees. The randomness is mainly reflected in the randomization of training samples and feature selections.



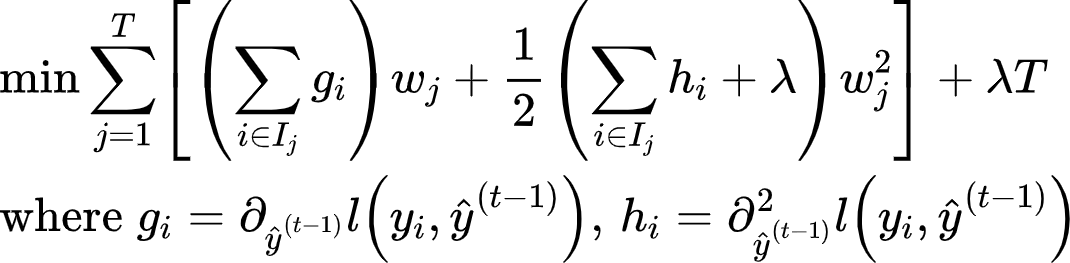
**Figure 5**. Procedural structure of a Random Forest algorithm with *n* trees (Goyal 2021).

The structure of a random forest algorithm composed of *n* decision trees is depicted in Figure 5. First, *m* samples were randomly placed back from the original training set using the Bootstrapping method for *n* times, then *n* training sets were generated. For *n* training sets, *n* decision trees are trained respectively. These decision trees form the Random Forest model. For classification problems, the final result is determined by a voting method. Random Forest is easy to implement and computationally inexpensive, but it shows amazing performance in classification problems.

### 3.5 XGBoost

XGBoost is short for "Extreme Gradient Boosting", of which the "Gradient Boosting" method is derived from Friedman’s (2001) greedy function approximation algorithm. XGBoost is an open source machine learning project developed by Chen and Guestrin (2016), which efficiently implements a gradient boosted decision tree and makes several algorithmic and engineering improvements.

Compared to Random Forest, XGBoost is an ensemble learning model based on the Boosting method. Boosting is an iterative algorithm, with a constant training set. Each iteration weights samples according to prediction results from the previous iteration, so that error decreases progressively. XGBoost is composed of a set of decision trees and each individual tree is trained using the CART algorithm (Chen and Guestrin 2016). The training of each tree depends on the residuals of the previous tree, and the residuals are used as targets for the next decision tree. Lastly, the final result is the sum of each tree’s results instead of an average.

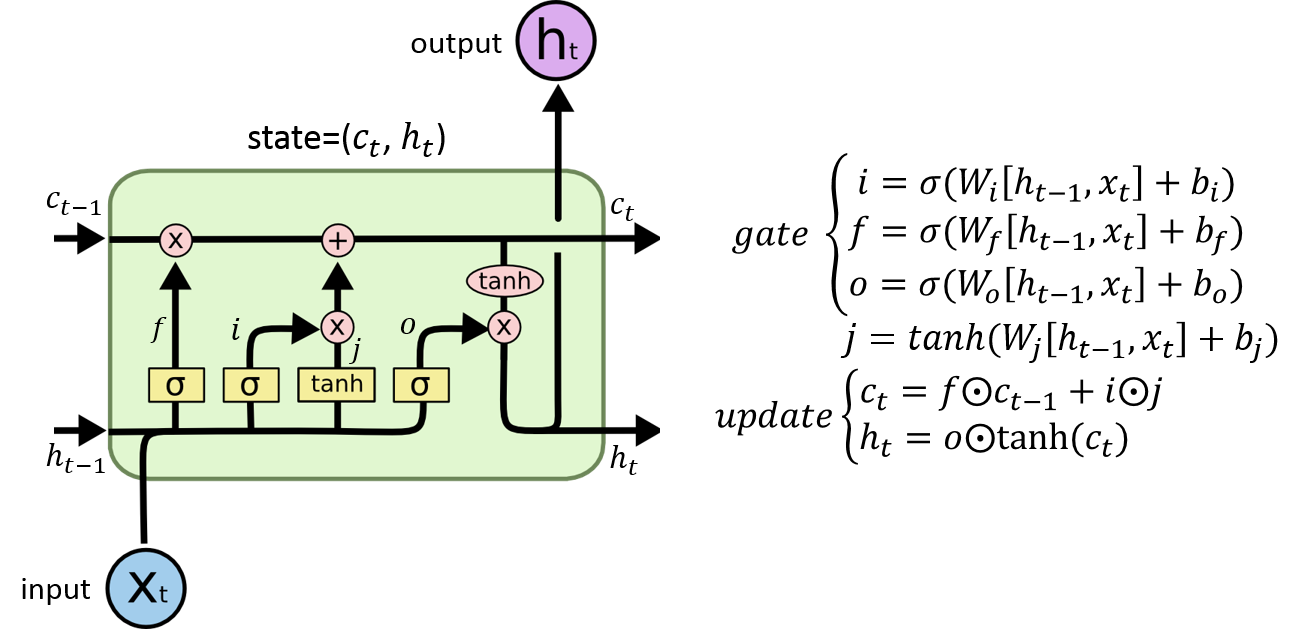


Further, the loss function of XGBoost summed with second-order gradient and regularization terms, which makes the objective function converge faster.

### 3.6 LSTM

Long short-term memory (LSTM) is a supervised deep learning model and it is a specialized Recurrent Neural Network (RNN), which is mainly used to solve vanishing gradient and gradient explosion problems in long sequence training. In simpler terms, LSTM are more tolerant to longer sequences than standard RNNs.

LSTMs, like other RNNs, are self-recurrent structures, continuously connected by multiple identical units after time-axis expansion. In general terms, the structure associates earlier inputs to the current input, allowing the model to grow and infer from its earlier context. In contrast to RNN, LSTM unit cells consist of three gates, which determine the information to forget or pass.



**Figure 6.** An algorithm diagram from LSTM models (Olah 2015).

As shown in Figure 6, the input gate *i* determines the portion of the current network input data to save to the unit state. The forget gate *f* determines the portion of the previous unit state to retain and incorporate to the current state. Lastly, the output gate *o* controls how much of the current cell state is fed to the current output value. LSTM controls the memory state of the previous information, input information and output information through these three gates, thus ensuring that the network can better handle the long-distance dependency relationship.

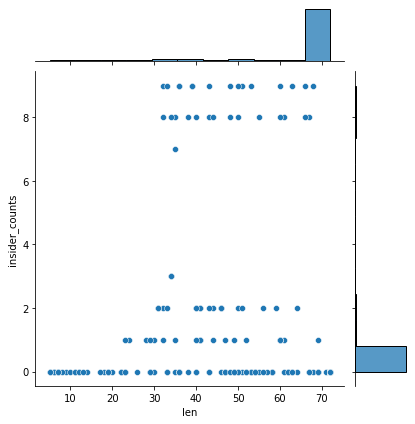
# EXPERIMENTAL EVALUATIONS

## 1. Data Configuration

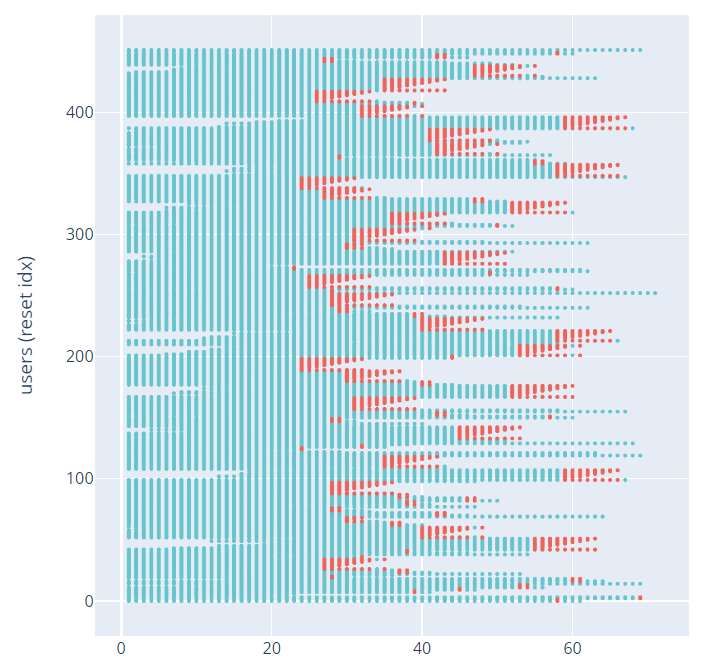
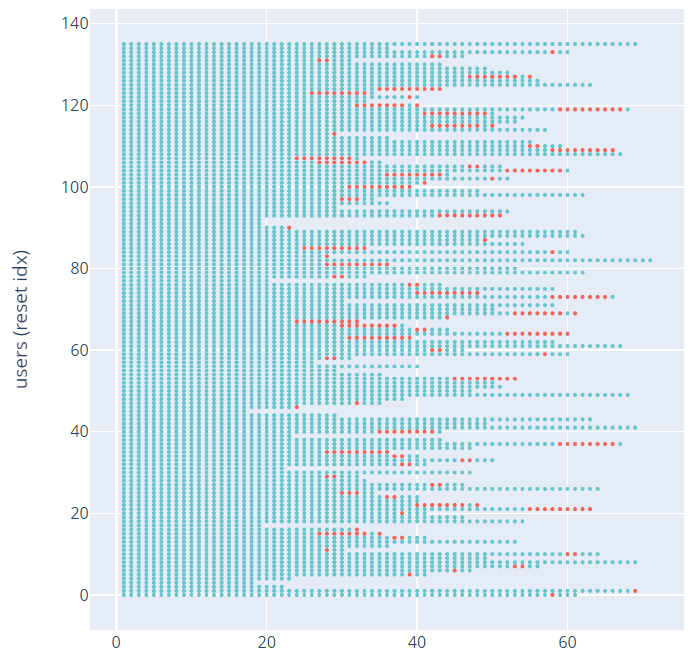
In this work, we used the CMU-CERT version r4.2 dataset, which contains activity data for 1000 users, of which a total of 70 users represent high-risk behavior. Though there are four types of insider threat scenarios, labeled separately, in the dataset, we simplified our problem into a binary classification problem to mitigate the negative impact of imbalanced data on model training. Since the dataset contains five log files with more than 10 million entries, we have deployed a MySQL database on Google Cloud to store our experimental data, in order to increase the efficiency of data extraction and handle version management.

In order to mitigate the effects of the imbalanced dataset, we performed data augmentation and normalization. In the raw data, 7% of the users had malicious activities and 0.47% of the activities were malicious. While inspecting the dataset in preparation for the data augmentation, we discovered several associations between user activity sequence length and the target label. For example, all users with activity sequence length less than 15 or exactly 72 were benign, as seen in Figure 7.

After removing all users with sequence length less than 15 or exactly 72, the dataset was reduced from 1000 users to only 136 users. Since the users with malicious activities were unaffected, 70 of these remaining 136 users were insiders.

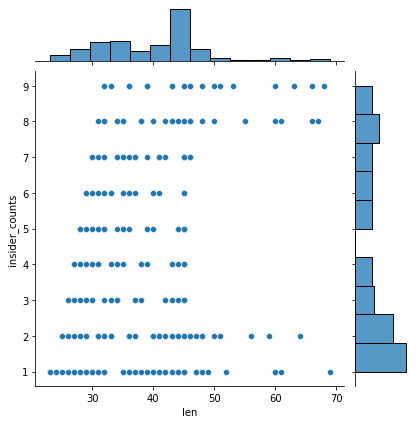


**Figure 7**. Original distribution of users’ insider activities and sequence lengths. There is a clear skew of activities with no insider activities, and significant skew in activity sequence lengths towards 72.



**Figure 8**. Data augmentation results by sliding window. The x-axis is a proxy for time, week wise. The blue species represent benign activities and the red denotes malicious behavior.

In order to increase the amount of data available and help further mitigate the imbalanced data, sequences were sliced with a target length 70% of the original distributions. Then, sliding window data augmentation was performed on the sequences containing malicious activity. This process slices sequences containing malicious activity to include subsequences with varying amounts of malicious activities along a sliding view. The length of the new sequences were sampled from a normal distribution that is similar to the original distribution of the sequences with malicious activities. Figure 8 depicts the effects of the data augmentation. In this less imbalanced, but smaller dataset, 85.40% of the users had malicious activities, but 13.67% of the activities were malicious. The new distributions of insider activities and sequence lengths can be seen in Figure 9.



**Figure 9**. Distribution of user insider activities and sequence lengths after filtering by length and performing sliding window data augmentation.

## 2. Experimental Settings

### 2.1 Model configurations and hyperparameter

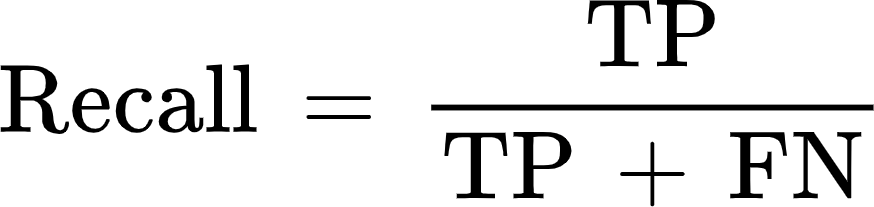
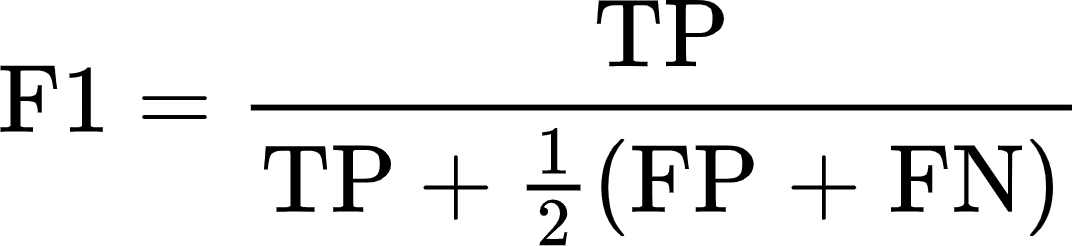
In the experiment, we used the Python 3.7 environment for data preprocessing and model training on Google Colab.

For our unsupervised learning model, we first tried to use the *pytorch* package to build a fully connected network, composed of three encoding layers and three decoding layers. Unfortunately, this implementation yielded poor model performance on detection. Our next implementation of AE employed the *MLPRegressor* from the *scikit-learn* package, and set up two encoding and decoding layers each with Tanh as the activation function. The training effect was significantly improved. In addition, we also tried the isolation forest model with 200 n-estimators and 256 max-samples.

For our supervised learning models, we applied Random Search Approach to find the best parameters for Random Forest (RF) and XGBoost. In detail, we tuned the number of estimators in RF from 100 to 500 and maximum depth of each decision tree from 4 to 10. We also tuned the number of estimators in XGBoost from 50 to 250, maximum depth of each tree from 5 to 25 and learning rate from 0.005 to 0.3. We also implemented early stopping in the XGBoost tuning to prevent overfitting.

### 2.2 Performance metrics

The Recall Rate is the primary consideration for model evaluation. Relative to benign logs, insider threats are infrequent and data is severely imbalanced. The model will occasionally return no positive predictions, so the Precision Rate is a misleading metric. Recall measures the frequencies that a class is properly detected, so it is more robust for cybersecurity contexts, when very few positive targets occur. F1 scores are the harmonic means between recall and precision, making them a well-rounded metric for general model comparisons.

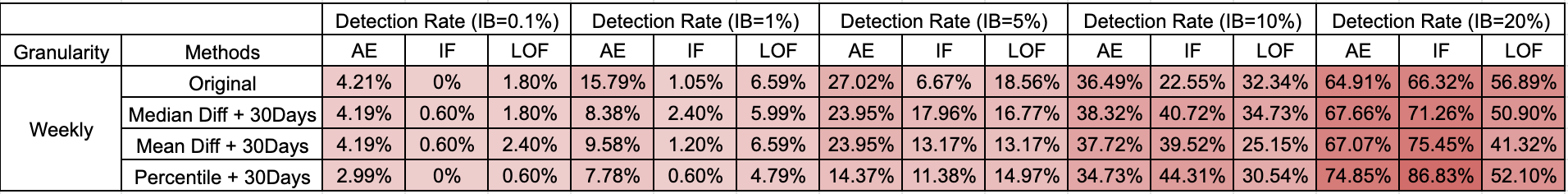
Furthermore, runtime is another important metric. Since the anomaly detection is a real time-series case, the system must finish predictions within at most one day. Lastly, the model cannot yield too many false positive results (malicious activities) or the human team will not be able to investigate fast enough. Organizations must determine a set percentage of the company population as a reasonable proportion for a security team to inspect and manually label in a given day.

## 3. Results and design conclusions

### 3.1 Unsupervised model comparisons

At the beginning, we conducted unsupervised learning approaches across the entire dataset as our baseline for subsequent model comparisons. We plotted the detection rate of AE, IF and LOF models on the regular dataset and three different time-correlated datasets.

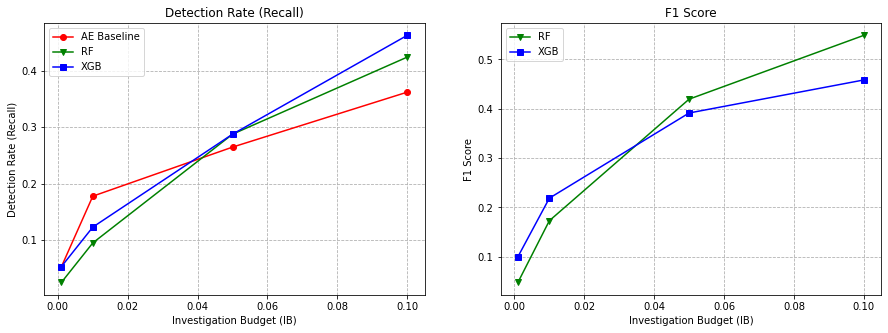
**Table 1**. Unsupervised Learning Models Performance with Different IB



According to the results shown in Table 1, there are two major takeaways. The first is that when the Investigation Budget Rate (IB) becomes 20%, the detection rates of the three models are quite high. However, the 20% IB puts too much pressure on the human investigation team. In addition, there will be more false positives yielded, which does not meet the requirements of our system performance metrics. The second takeaway is that among these three unsupervised models, AE and IF have better performances. Furthermore, the performance of AE is more reliable than IF over the appropriate IB range (under 10%). Therefore, the team decided to use AE as the principal unsupervised learning model to produce labeled data for subsequent supervised learning model training.

### 3.2 Supervised model

The supervised learning algorithms employed and compared include Random Forest, XGBoost and LSTM-based CNN. Merging results from manual screening and the supervised learning predictions yields scores for the whole dataset. According to Figure 6, overall our system with supervised learning models performs better than the purely unsupervised learning baseline model. In addition, with the same increasement in IB, our model has higher growth than the baseline. Specifically, at the 10% IB, our system has a 10.1% increase in detection rate compared to the baseline model.



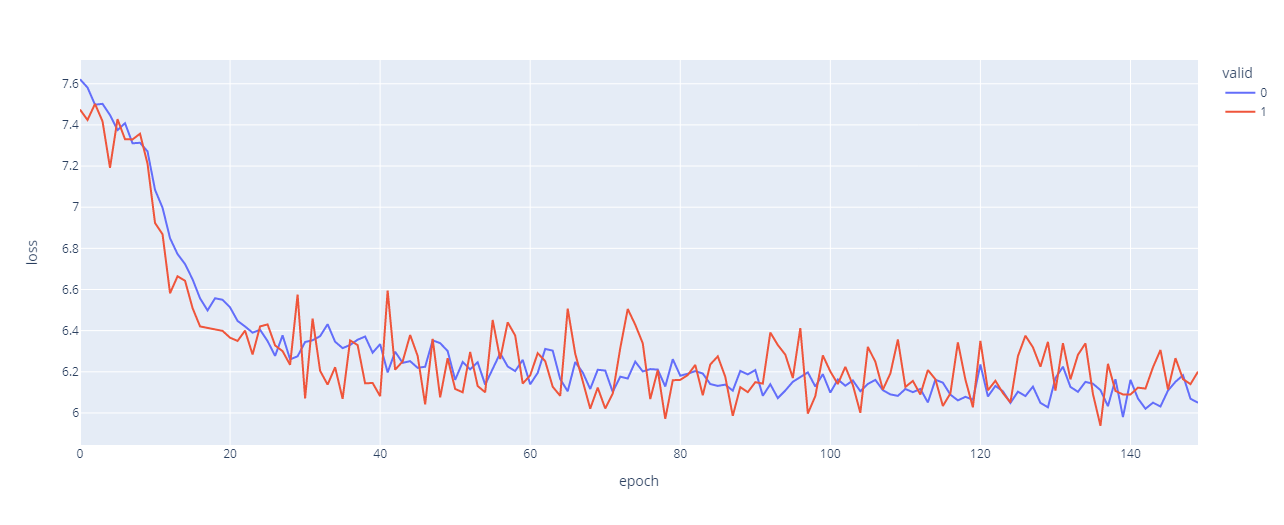
**Figure 10**. System performance in both recall and F1 score for varying IBs.

For our deep learning models, several experiments were performed to increase the recall. The most effective methods for improving performance were optimizing normalization, data augmentation and adding weight to the loss function.

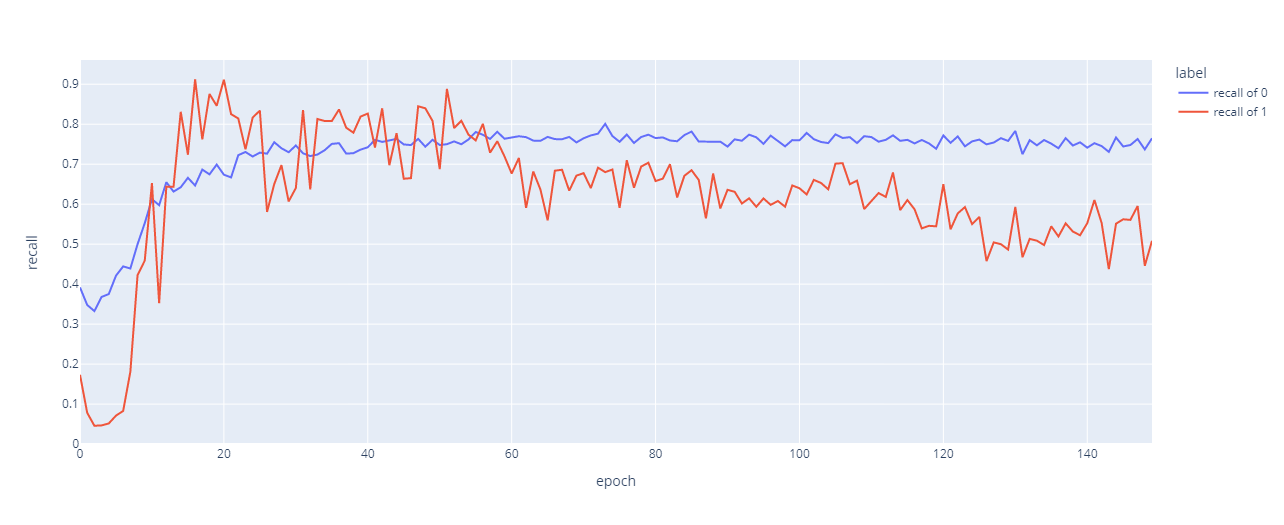
Our deep learning classifier employed two encoding layers, followed by an LSTM layer, two linear hidden layers and an output layer. Output logits are passed through a softmax and logarithm to obtain prediction log-probabilities.

Experimentation was performed with the convolutional layers, but due to the small amount of malicious data, increasing network complexity did not increase performance. Similarly, increasing the hidden size in LSTM and Linear layers did not improve the recall rate. In addition to adding depth and complexity to the network, we tried improving the optimizer and loss function. Given the skewed distribution of our data, our loss function yielded small (and thus unsteady) values, so parameters were not updating between epochs and loss did not hold a downward trend. We employed weights greater than unity to boost the loss value and force parameter updates. During the experiments, we also noticed that the last batch in each epoch had a very unsteady loss value. Due to the skewed distribution, subsets of the data frequently contained only one class and induced bad performance. After dropping the last batch in the data loader, the recall on malicious data has been raised slightly.

After the data processing and fine tuning, we were able to raise the recall rate on malicious data while maintaining the same high recall rate on clean data. The latest model has raised the recall rate on malicious data from 32% to 88.8% , with a downward trend in the loss (see Figure 11).

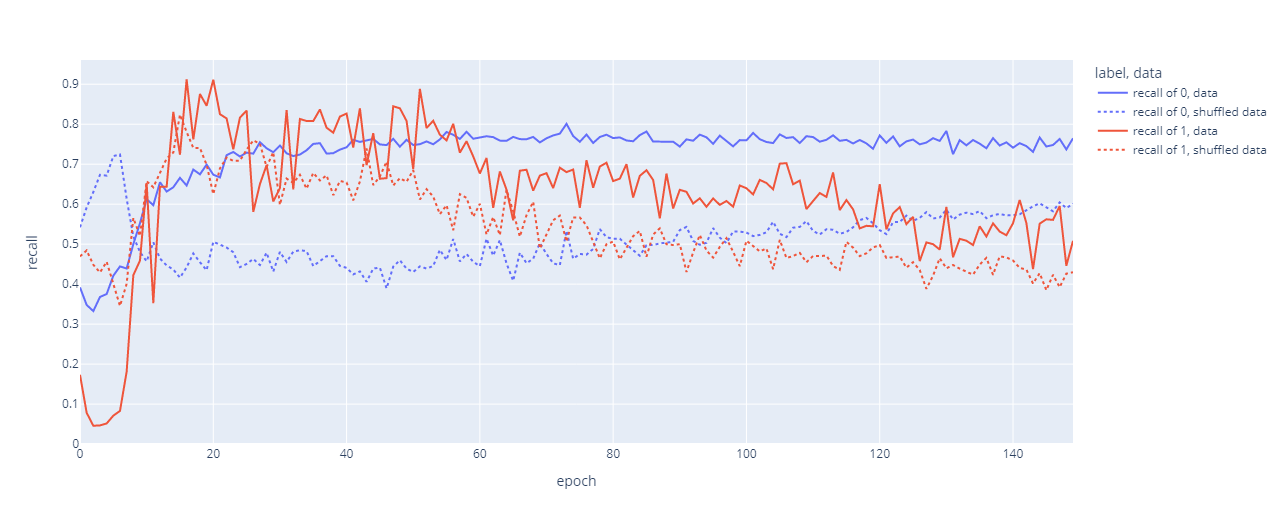


**Figure 11**. Loss by epoch, for validation data (red) and training data (blue).



**Figure 12**. Model performance, measured in recall, by epoch, of malicious data (red) and clean data (blue).

Additionally, we assumed that the activities should be time dependent before using the LSTM-based model. In order to prove the time-dependence and validity of RNN, we shuffled the data and used the same model to compare. We kept the original users, but shuffled the activity sequences of each user. From the figure below we can see that if we use the shuffled data, the recall on malicious data can reach the level of 80%, but at a really low recall on clean data. The trend of recall on both groups is towards 50% and 50%, which is not as satisfying as the performance on time ordered data. Therefore, it’s reasonable to use recurrent neural networks like LSTM on the CMU-CERT data.



**Figure 13**. Model performance on original data and shuffled data (dot line), measured in recall, by epoch, of malicious data (red) and clean data (blue).

# CONCLUSION

## Overview

Insider threat detection is a difficult problem because imbalanced datasets lead to algorithms with many false positives, which overwhelm human investigators. Our ITD procedure introduces tolerance to low IB by blending unsupervised and supervised models. An unsupervised anomaly detection system identifies suspicious behavior in a fragment of the dataset, and prompts human investigation. Through a combination of manual investigation and default labeling from the unsupervised model, the dataset fragment is labeled and used to train a supervised classifier. The supervised classifier is much higher accuracy than the unsupervised model, and predicts the remaining segments of the dataset. This method aids in allocating investigation resources to reduce reliance on human efforts, while employing the high detection rates associated with supervised models. Our system is tested on the r4.2 release of the synthetic CMU-CERT dataset. Limitations in the CMU-CERT dataset were found that would cause length-based classification; filtering these users and reducing imbalance with data augmentation resulted in a small dataset that limited model performance. Model performance was unaffected by increased complexity or depth due to the small dataset. AutoEncoder reconstruction error is used for unsupervised anomaly detection and an LSTM network is used for supervised classification. Performing data augmentation with a sliding window process and weighting the loss function at greater-than-unity resulted in a more effective training process and improved recall rates from 27% to 62.98%.

## Future work

This work raises several points for future development, with three main considerations. Firstly, this model is untested on real-world data; our system is developed for a single generation of the CMU-CERT synthetic dataset, and tailoring it and testing with actual user activity logs may illuminate additional strengths and weaknesses. Additionally, real-world data would not have the size and sequence length limitations found in the CMU-CERT dataset, and may benefit from higher complexity classifiers. Secondly, our varied data granularities were aggregated in order to construct contextual features, but our overall process may perform differently when comparing data granularities across time, size (data stream) or other features. Similarly, under larger datasets, we may see benefits when training separate models for different employee departments; for example, anomalous web browsing activity is likely to differ between an engineering team and a human resources team. Varying the groupings and granularity of our data is likely to yield interesting results. Additionally, the dataset contains four different types of malicious scenarios, but we simplified the classification problem into binary categories—malicious or benign—in order to help mitigate the imbalanced data. A future direction for this research would be increasing the classification categories to accommodate the different scenarios and determine if performance can be maintained.

# REFERENCES

Bailey, Tucker, Brian Kolo, Karthik Rajagopalan, and David Ware. 2020. “Insider Threat: The Human Element of Cyberrisk.” McKinsey & Company. Accessed November 11, 2021. <https://www.mckinsey.com/business-functions/risk-and-resilience/our-insights/insider-threat-the-human-element-of-cyberrisk>.

Breiman, Leo. 2001. “Random Forests.” *Machine Learning* 45, no. 1 (2001): 5–32. https://doi.org/10.1023/a:1010933404324.

Breunig, Markus M., Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. “LOF.” *ACM SIGMOD Record* 29, no. 2 (June 2000): 93–104. https://doi.org/10.1145/335191.335388.

Caputo, D., Maloof, M., & Stephens, G. (2009). Detecting Insider Theft of Trade Secrets. *IEEE Security & Privacy Magazine*, *7*(6), 14–21. https://doi.org/10.1109/msp.2009.110

Chalapathy, Raghavendra, Aditya Krishna Menon, and Sanjay Chawla. 2017. “Robust, Deep and Inductive Anomaly Detection.” Machine Learning and Knowledge Discovery in Databases, 36–51. https://doi.org/10.1007/978-3-319-71249-9\_3.

Goyal, Chirag. 2021.“Random Forest Interview Questions: Random Forest Questions.” Analytics Vidhya, June 8, 2021. https://www.analyticsvidhya.com/blog/2021/05/bagging-25-questions-to-test-your-skills-on-random-forest-algorithm/.

Chen, Tianqi, and Carlos Guestrin. 2016. “XGBoost: A Scalable Tree Boosting System.” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 13, 2016, 785–94. https://doi.org/10.1145/2939672.2939785.

Eberle, W., Graves, J., & Holder, L. 2010. “Insider Threat Detection Using a Graph-Based Approach.” Journal of Applied Security Research, 6(1), 32–81. https://doi.org/10.1080/19361610.2011.529413

Flitter, Emily, and Karen Weise. 2019. “Capital One Data Breach Compromises Data of Over 100 Million.” *New York Times*, July 29, 2019. https://www.nytimes.com/2019/07/29/business/capital-one-data-breach-hacked.html

Friedman, Jerome H. 2001.“Greedy Function Approximation: A Gradient Boosting Machine.” *The Annals of Statistics* 29, no. 5 (2001). https://doi.org/10.1214/aos/1013203451.

Hinton, G. E., and R. R. Salakhutdinov. 2006. “Reducing the Dimensionality of Data with Neural Networks.” *Science* 313, no. 5786 (July 28, 2006): 504–7. https://doi.org/10.1126/science.1127647.

Isaac, Mike. 2019. “Former Star Google and Uber Engineer Charged With Theft of Trade Secrets.” *New York Times*, August 27, 2019. https://www.nytimes.com/2019/08/27/technology/google-trade-secrets-levandowski.html

Glasser, Joshua and Brian Lindauer. 2013. "Bridging the Gap: A Pragmatic Approach to Generating Insider Threat Data," In *2013 IEEE Security and Privacy Workshops*, pp. 98-104. IEEE, 2013. https://doi.org/10.1109/SPW.2013.37.

Guha, Sudipto, Nina Mishra, Gourav Roy, and Okke Schrijvers. “*Robust random cut forest based anomaly detection on streams.*” In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 2712-2721. Proceedings of Machine Learning Research, 2016.

Le, Duc C., Nur Zincir-Heywood, and Malcolm I. Heywood. 2020. “Analyzing Data Granularity Levels for Insider Threat Detection Using Machine Learning.” IEEE Transactions on Network and Service Management 17 (1): 30–44. https://doi.org/10.1109/tnsm.2020.2967721

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "[*Isolation forest.*](https://cs.nju.edu.cn/zhouzh/zhouzh.files/publication/icdm08b.pdf)" In *2008 eighth IEEE international conference on data mining*, pp. 413-422. IEEE, 2008.

Malhotra, Pankaj, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. “Long Short Term Memory Networks for Anomaly Detection in Time Series.” <https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2015-56.pdf>.

Moore, Andrew P, William E Novak, Matthew L Collins, Randall F Trzeciak, and Michael C Theis. 2015. “Effective Insider Threat Programs: Understanding and Avoiding Potential Pitfalls.” CMU SEI Publications. Software Engineering Institute, Carnegie Mellon University (October). <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=446367>.

​​Olah, Christopher. “Understanding LSTM Networks.” Understanding LSTM Networks, 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Sage, Alexandria, Dan Levine, and Heather Somerville. 2018. “Waymo accepts $245 million and Uber's 'regret' to settle self-driving car dispute.” *Reuters*, February 9, 2018. https://www.reuters.com/article/us-alphabet-uber-trial/waymo-accepts-245-million-and-ubers-regret-to-settle-self-driving-car-dispute-idUSKBN1FT2BA

Yuan, S., & Wu, X. (2021). Deep learning for insider threat detection: Review, challenges and opportunities. Computers & Security, 104, 102221. https://doi.org/10.1016/j.cose.2021.102221

Zhou, Chong, and Randy C. Paffenroth. "Anomaly detection with robust deep autoencoders." In Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 665-674. 2017.

Zenati, Houssam, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. "Efficient GAN-based anomaly detection." arXiv preprint arXiv:1802.06222 (2018).

Pledge of Academic Integrity

We affirm that we are the sole authors of this report and we give due credit (i.e., use correct citations) to all used sources.

|  |  | April 13, 2022 |
| --- | --- | --- |
| Noah Asing |  |  |
|  |  | April 13, 2022 |
| Wenqi Kou |  |  |
|  |  | April 13, 2022 |
| Tianxiao Gaoqu |  |  |
|  |  | April 13, 2022 |
| Diru Jia |  |  |