A Novel Method for Cybersecurity Malicious Activity Detection Using Semi-Supervised Machine Learning Based on Known Bad Records

Hongwei Will Li

Department of Electrical Engineering,

Villanova University, Villanova, PA 19085 USA

hli8@villanova.edu

**Disclaimer**: The author is also currently employed by a large financial firm as a cyber-security professional. Due to intellectual property concerns, (my current employment contract has broad intellectual property scope, including work I performed outside of employment settings), the ideas presented in this report have been pared-down in terms of the proposed solution complexity and degree of novelty. It has also been scrubbed clean of any Personal Identification Information (PII) not related to the author, except the information already in public domain.

Further, the author reserves all intellectual property rights permitted by Villanova University policy.

# Abstract:

To combat the growing threat of cyber based threats, cyber security professionals are always on the look out for new ways to detect and protect against malicious software and actors with intent to harm. Artificial Intelligence and Machine Learning have shown much potential to improve the capabilities of the defenders. In this paper, we establish theoretical basis on why we need to explore semi-supervised learning, provide in-depth review of published semi-supervised machine learning methods, and propose a novel Tri-training technique to apply semi-supervised machine learning to suit the specific needs of cyber-security analysis. The proposed Tri-training method will utilize Random Forest algorithm, Support Vector Machine (SVM) algorithm, and the Shared Nearest Neighbor algorithm, based on the often class-imbalanced nature of the cyber security data set, and the lack of sufficient training data. Through utilization of appropriate machine learning performance metrics, and comparing the efficacy of the proposed method over published benchmarks for supervised and unsupervised machine learning techniques, we conclude that by utilizing the large amount of unlabeled data currently available, in addition to the limited training data from “Known Bad Records”, cyber security professional in deed can leverage this new method to gain new capabilities in the fight against malicious cyber threats.

# Introduction

Russian attacks of America’s electric grid have recently been reported by New York Times report (Drago, 2018) and the Wall Street Journal report (Smith & Barry, 2019). These reports not only described the details of some of these attack attempts, but also highlighted the importance of strengthening cyber-security defense around the electric grid.

Cyber-security professionals have always relied on rules and signature-based tools to help with the defensive efforts (Davis & Clark, 2011). However, as demonstrated by these recent news reports, the cyber-attack tactics, techniques, and procedures (TTP) have increased in magnitude and sophistication to the point where zero-days, nation state actors, and advanced persistent threats are now part of the everyday news-lingo. This is where the traditional defensive tool sets are falling behind of adversaries.

Artificial intelligence (AI) and Machine learning (ML) techniques have been hailed as able to make a profound impact to data-driven business (Kronz, Gartner, 2019). This has for sure caught the attention of many cyber-security professionals, including the author as the new hope to defend against constantly advancing adversarial TTP’s.

The most well-known of the cyber security public data set is the KDD Cup 1999 data ("KDD Cup 1999 Data," 1999), and it had helped spawn a large body of literature on machine learning studies (Patcha & Park, 2007), (Davis & Clark, 2011), (Aggarwal & Sharma, 2015), (Moustafa & Slay, 2015), (Moustafa & Slay, 2016), and (Aburomman & Reaz, 2017).

The KDD99 data set, for example, consists of around 4.9 million rows of data for the training set, with about 80% of malicious data points. The test data has about 300,000 rows, and around 75% of the test records are malicious. Even though these characteristics are drastically different from real life network traffic, the importance of these studies was that they have blazed the trails, have proven that machine learning methodology can be used for cyber security studies, and have provided the necessary playgrounds for new generations of cyber security professionals to learn and practice using machine learning as a new tool.

# The Need for Semi-Supervised Learning

In real life cybersecurity applications, most often we are faced with a large amount of unlabeled data, and a small set of labeled data obtained from either white-lists or more often black-lists, as the author calls it: Known Bad Records (KBR’s). This is unlike most of the well-formed public data sets (e.g. KDD Cup 1999), where there exist a large pre-labeled training data set.

This problem of training data shortage of high-quality cyber-security cannot be easily solved like popular image classification problems, where training data can be obtained almost for free by launching a popular photo contest (Silverstein, 2019). In contrast, it is a very expensive proposition to assign experienced cyber-security analysts to spend months just to help data scientists to label millions of seemingly normal traffic logs with a small percentage of malicious traffic.

Therefore, the author proposes a novel **Semi-Supervised learning (SSL)** of generating predictions or classifications to take advantage of the easily available and large quantity of unlabeled data.

But first, we want to start with a quick review of the three basic types of machine learning, according to definitions provided by (Chapelle, Schölkopf, & Zien, 2010): unsupervised, supervised, and semi-supervised learning.

**Unsupervised learning** is where if a set of data points X= (x1, …, xn) is given, and one is asked to find interesting structure in the data X. Most often, this involves estimating a density distribution of the data set X, but the unsupervised task can also be quantile estimation, clustering, outlier detection, or dimensionality reduction.

For **supervised learning**, the goal is to learn a mapping from X to Y, with the training set made of pairs (xi, yi). The data points yi are often called vector of labels for examples of xi. With this richer information of label vectors provided, the mapping between X to Y can be predicted by many families of supervised learning methods, some are generative in nature, where the algorithm tries to model the class-conditional density p(x|y) and infer a predictive density according to Bayes theorem:

while others are discriminative in nature, where they do not try to estimate how the xi have been generated, but instead try to directly estimate p(y|x).

**Semi-supervised learning (SSL)** is halfway between supervised learning and unsupervised learning. Even though some labeled data is given, the majority of the data points are unlabeled.

The data set X = (xi) is divided into two parts:

1. The points Xl := (x1, …, xl), for which labels Yl := (y1, …, yl) are provided
2. The points Xu := (xl+1,…,xl+u), and u>>l, where the labels are unknown.

# In-Depth Review of Semi-Supervised Learning Methods

Although it is often taken for granted, but supervised learning requires the **smoothness assumption** for the learnings to be generalized from the given training data to unseen test data: If two points x1, x2 are close, then the output labels y1, y2 should also be close.

Similarly, there is also the **semi-supervised smoothness assumption** as stated in (Chapelle, 2010): “If two points x1, x2 in a high-density region are close, then so should be the corresponding outputs y1, y2.”; “If on the other hand, they are separated by a low-density region, then their outputs need not be close.”

A corollary of the **smoothness assumption** is the **cluster assumption**: If points x1, x2 are in the same cluster, they are likely to have the same class label.

A further corollary of the above assumptions is the **low-density separation assumption** as elegantly stated by (Chapelle, 2010): “The decision boundary should lie in a low-density region.”

Armed with these basic assumptions, one can see why the study of Xu := (xl+1,…,xl+u), and gaining knowledge about p(x), should provide us with useful information about p(y|x), in addition to what learning we can achieve from the labeled data Xl := (x1, …, xl), and their labels Yl := (y1, …, yl). Therefore, the employment of semi-supervised learning is possibly productive over supervised learning, given the relatively small size of l and large size of u.

Many algorithms were given as examples (Chapelle, 2010) that can be employed to perform semi-supervised learning, such as:

1. Generative models including Gaussian Mixture Model (GMM), Naïve Bayes, or Hidden Morkov Models (HMM)
2. Transductive SVM (TSVM), or Semi-supervised SVM (S3VM)
3. Graph-based models

**Self-Training**: Despite the differences in the specific algorithms or models, the main method used is Self-Training. It relies on its own predictions on unlabeled data to add to the training data set:

1. A model m is initially trained on the labeled data set Xl.
2. At each iteration, predictions from the unlabeled data set is added back to the labeled set Xl, given the predictions pass a pre-determined threshold τ.
3. These new labels are called pseudo-labels or proxy-labels.
4. The iteration steps can be repeated as necessary, or convergence when the prediction confidence is maximized.

Self-Training Algorithm:

1: **repeat**

2: m 🡨 train\_model (Xl)

3: **for** x Xu **do**

4: **if** max m(x) > τ **then**

5: Xl 🡨 Xl {(x, p(x))}

6: **until** prediction confidence is maximized

As pointed out by (Ruder, Plank, 2018), the self-training method has shown some successful use cases, but small errors in the beginning tend to be amplified with the iteration steps.

**Co-training** was later introduced by Blum and Mitchell (Blum & Mitchell, 1998) to bring in two different view points to minimize the amplification of intrinsic inductive biases.

1. This can be accomplished either by splitting the features into two different sets or by using two different algorithms.
2. At each iteration, only inputs that are higher than a threshold τ according to exactly one of the two models are moved to the training set of the other model.
3. One model thus provides the labels to the training set on which the other model is uncertain.

Co-Training Algorithm:

1: **repeat**

2: m1 🡨 train\_model (Xl1)

3: m2 🡨 train\_model (Xl2)

4: **For** x Xu **do**

5: **if** max m1(x) > τ **and** max m2(x) < τ **then**

6: Xl2🡨 Xl2 {(x, p1(x))}

7: **if** max m2(x) > τ **and** max m1(x) < τ **then**

8: Xl1🡨 Xl1 {(x, p2(x))}

9: **until** prediction confidence is maximized on *one* classifier

Tri-training was proposed by Zhou & Li (Zhou & Li, 2005) to include democratic-style learning with three models in semi-supervised learning mode.

1. This method will use three different models to further diversify on the inductive biases.
2. At each iteration, if two of the three models agree on a label for a previously unlabeled data point, it is added to the third model as training data.
3. The final prediction can be made with a majority mode scheme.

Tri-Training Algorithm:

1: **for** i {1,..,3} **do**

2: Si 🡨 bootstrap\_sample (Xl)

3: mi 🡨 train\_model (Si)

4: **repeat**

5: **For** i {1,..,3} **do**

6: **For** x Xu **do**

7: **if** pj(x) = pk(x) (j, k i) **then**

8: Xli🡨 Xli {(x, pj(x))}

9: mi 🡨 train\_model (Xli)

10: **until** prediction confidence is maximized on *all* classifiers

11: **apply** majority vote over mi

Related work and algorithms employed to tri-train include:

1. J48 Decision Trees, BP Neural Networks, and Naïve Bayes algorithms were used in (Zhou & Li, 2005)
2. Many variations of Support Vector Machines (SVM) in (Li, Zhang, & Li, 2010)
3. Naïve Bayes, Random Forest, and Decision Tree in (Søgaard & Rishøj, 2010)
4. Support Vector Machine (SVM), Multinomial Logistic Regression (MLR), Extreme Learning Machine (ELM) and k-Nearest Neighbor (KNN) in (Tan, Zhu, Du, Wu, & Du, 2016)

# Additional Challenges for Cyber-Security Analysis

Compounding the problem of not having sufficient training data, another unique challenge for real life cyber-security studies is the class-imbalanced nature of the data.

Using binary classification problem as an example, traditional machine learning algorithms have the intrinsic assumption that the two classes have roughly the same number of data points. In most cases, the majority of the cyber-security data will be normal data. Only a minority percentage of the total network traffic, files, system calls, etc., actually belong to malware or were generated by malicious intents. (The exception might be Distributed Denial of Service (DDoS), but that exception usually only holds true for a relative short burst when DDoS attack is underway). This is one reason many practitioners also refer to the task of identifying malicious traffic as anomaly detection.

As Krawczyk (Krawczyk, 2016) and Guo (Guo, et. al. 2017) summarized, most learning algorithms will be biased toward the majority class. But, it is the minority class that is of interest to the cyber-security data scientists!

Take K-Nearest Neighbor (KNN) algorithm as an example, let’s assume 99% of the training example is normal or “0”, and only 1% is malicious or “1”, the generic version of KNN will have a hard time identifying minority class data points (rare and infrequent events).

As an extreme example, the easiest way to guarantee 99% prediction accuracy is to always label every data point normal or “0”, regardless of what we can learn from the features. While this fictitious data scientist maybe happy singing success with a 99% metric, the adversaries could be even happier roaming around the target network with the slightest worry of being detected.

There are three main approaches to solve the challenges posed by class-imbalanced data sets: data-level methods, algorithm-level methods, or hybrid methods that combine the previous two approaches.

At data-level, one may choose to either artificially reduce the size of the majority class (Down-Sampling), or increase the size of the minority class (Up-Sampling). The concept sounds straightforward, but it is the execution that warrants many scientific studies, if one is not satisfied with a random execution method. For example, instead of simply repeat the same minority class data points, Synthetic Minority Over Sampling Technique (SMOTE) was proposed to increase the size of the minority class. (Chawla, Bowyer, Hall, & Kegelmeyer, 2002).

At algorithm-level, one may choose algorithms that are less prone to majority-class bias, or adopt cost-sensitive learning by assigning a higher cost to boost the importance of the minority class data points. As pointed out by Guo (Guo, et. al. 2017), it is difficult to set the cost matrix, since the mis-classification costs can be hard to quantify beforehand. This was probably the main reason cost-sensitive learning remains an under-represented topic in scientific literature.

# Proposed Methods

In this paper, addressing the twin challenges of the lack of training data, and class-imbalanced nature of real-world cyber-security problems, the author proposes a novel Tri-training technique to apply semi-supervised machine learning technique to the world of cyber-security analysis.

Random Forest and Support Vector Machine (SVM) algorithms from Scikit Learn were selected to provide two different view points for the tri-training analysis. These two algorithms have significantly different inductive biases, and have been proven to have good performance in past studies by the author on cyber-security data sets, as shown in Figure 2.

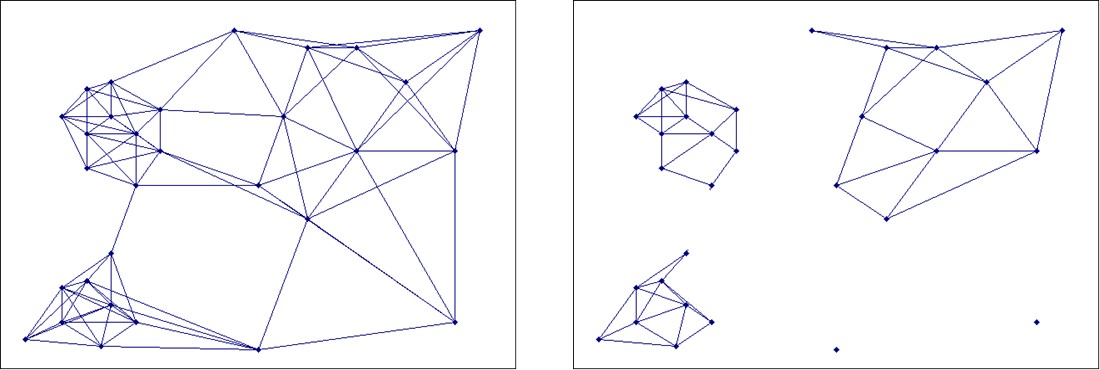
Taking cues from some recent fraud investigation (most often class-imbalanced) techniques ("Link Analysis and Crime - An examination," 2015), Shared Nearest Neighbor (SNN), a Link-based Object Classification (LOC) algorithm was selected as last of the three algorithms for the tri-training method. The Shared Nearest Neighbor Algorithm for the classification analysis was first proposed by L. Ertöz (Ertöz, Steinbach, & Kumar, 2003) and reviewed by K. Wilson in (Samatova, Hendrix, Jenkins, Padmanabhan, & Chakraborty, 2013, p. 135-166) on the Graph-based Proximity Measures to perform Link-based Object Classification (LOC).

The main steps of Shared Nearest Neighbor (SNN) can be described below as a two-tier algorithm:

1. Each of the input data point will be a Node in a Graph.
2. Define **Connectedness** as having at least one feature (or dimension of the vector) as being “**equal**”.
   1. Initial definition of a feature being “**equal**” can be **identical string comparison**, such as identical Name string field. In subsequent iterations of this algorithm, the “equal-ness” can be redefined and relaxed based on specific domain knowledge, such as IP addresses in the same IANA assigned subnet, or from the same geographic area, ISP, or autonomous system number (AS-number). the “equal-ness” can also be collapsing all zip codes within the same county to a single value.
   2. A pair of **nodes are connected** with an Edge if the two nodes meet the **Connectedness** condition.
3. Pick an integer as the seed value for hyperparameter k for K shared nearest neighbors.
4. Run **Shared Nearest Neighbor Clustering algorithm** (defined below) to find clusters.
5. Label any data points that are in the same cluster as one or more training data points as **Malicious**.
6. For training data set only. Optimize the hyperparameter k (from step 3) to achieve desired level of performance metric with k-opt.
   1. The neighborhood list size, k, is the most important parameter as it determines the granularity of the clusters. If k is too small, even a uniform cluster will be broken up into pieces due to local variations in the similarity, and the algorithm will tend to find many small, but tight, clusters. On the other hand, if k is too large, then the algorithm will tend to find only a few large, well-separated clusters, and small local variations in similarity will not have an impact. The parameter k, adjusts the focus of the clusters. Once the neighborhood size is fixed, the nature of the clusters that will be produced is also fixed.

The **Shared Nearest Neighbor Clustering algorithm** (with the Graph as input from the **Machine Learning Routine**, and the value of k) (Ertöz et al, 2003):

1. Compute the **similarity matrix**. This corresponds to a similarity graph with data points for nodes and edges whose weights are the similarities between data points. **Similarity is defined as two nodes share the same neighbor, weights equal to the number of shared neighbors**.
2. Sparsify the similarity matrix by keeping only the k most similar neighbors. This corresponds to only keeping the k strongest links of the similarity graph. (As shown in the following two diagrams, the right diagram is a sparsified version of the left diagram by removing edges that have weights less than k)



(Figure 1. reproduced from (Ertöz et al, 2003), permission request TBD)

1. Construct the shared nearest neighbor graph from the sparsified similarity matrix. At this point, we could apply a similarity threshold and find the connected components to obtain the clusters. (Jarvis, Patrick, 1973)
2. Find the SNN density of each Point. Using a user specified parameter, Eps, find the number of points that have an **SNN similarity** of Eps or greater to each point. This is the **SNN density** of the point.
3. Find the core points. Using a user specified parameter, MinPts, find the core points, i.e., all points that have an SNN density greater than MinPts.
   1. MinPts should be a fraction of the neighborhood size, k.
4. Form clusters from the core points. If two core points are within a radius, Eps, of each other they are place in the same cluster.
5. Discard all noise points. All non-core points that are not within a radius of Eps of a core point are discarded.
6. Assign all non-noise, non-core points to clusters. This can be done by assigning such points to the nearest core point.

# 5. Performance Metric Selection

The metrics most often used by data scientists to measure the efficacy of a particular study include (Guo, et. al., 2017):

1. =
2. =
3. Receiver Operating Characteristic (ROC) Curve, which plots True Positive Rate (TPR) against False Positive Rate (FPR)

When applied in cyber-security field, due to the class-imbalanced nature, the number of negative samples usually overwhelm the positive samples. This makes the Accuracy and FPR rather insensitive to the actual performance of the model.

Because Precision focuses more on the predicted positives rather the negatives, it is affected less by the large number of true negative samples. It is a good metric to use when the cost of false positive detection is high, such as in email spam detection if a “ham” is labeled as “spam”.

Recall, on the other hand, focuses on the actual positive samples. It is the metric of choice if the cost of false negative detections is very high, such as prediction of contagious virus.

F1 Score provides the balance between Precision and Recall by calculating the harmonic mean of Precision and Recall.

Therefore, the rest of the study will adopt **F1 Score** as the measure of study performance to take into account of the effect of both False Positive and False Negative predictions. However, in specific applications, model tuning should be adopted with the most appropriate metric.

# 6. Proposed Experiment

The author attempts to apply machine learning techniques to study more recent public data sets on cyber security domain, which provides more modern normal data points, as well as more current attack traffic. The public data set used is published by Moustafa and Slay. (Moustafa & Slay, 2015), and (Moustafa & Slay, 2016) and available at <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>

These data sets were generated using normal activities and synthetic attack traffic from an IXIA PerfertStorm tool network simulation tool. The resulting network traffic was then analyzed by Argus tool and Bro-IDS tool to provide the 40+ features that can be reasonably made available to a modern-day cyber analyst. Detailed malicious attack types include: Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, Shellcode, and Worms.

Further, in order to simulate real life cyber security data, the author down selected the malicious data points to be about only 1% of the size of the normal data.

The results of this tri-training study will be compared to benchmarks for popular supervised and unsupervised machine learning techniques to demonstrate the efficacy of the proposed semi-supervised methods.

The results will also be compared to popular tri-training methods published for balanced data sets to highlight the improvements of the proposed method when facing class-imbalanced data.

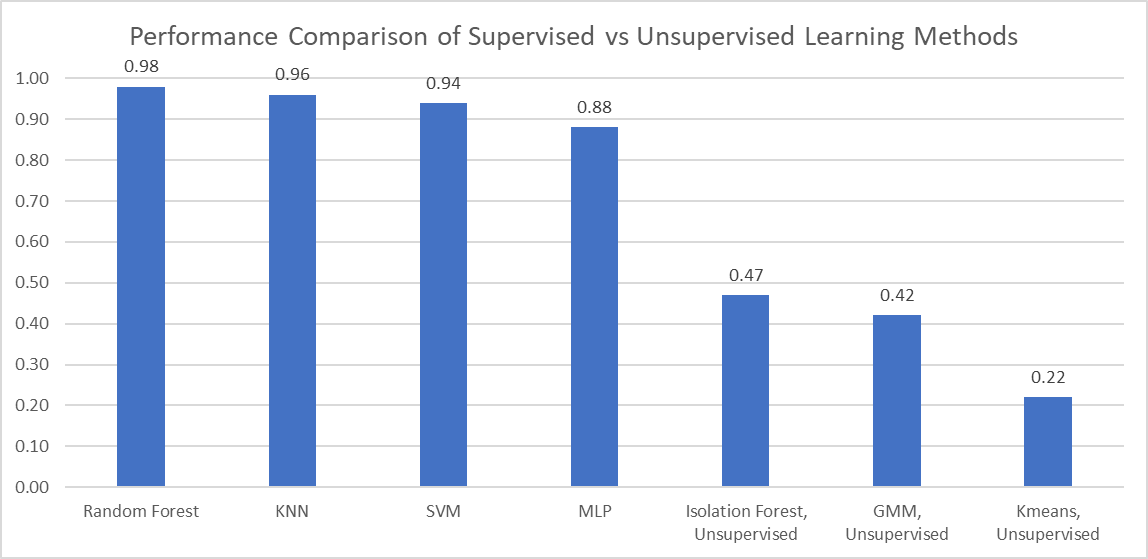


Figure 2. Sample machine learning efficacy comparison chart

# 7. Conclusions and Future Work

In this paper, the author pointed out the key challenges of conducting real life cyber security studies when compared to past studies using synthetic data sets.

First, quality labeled training data is very expensive to obtain. Instead of hiring multiple dedicated cyber security analysts to label millions of data points accurately, the author proposed to utilize the limited amount of data from white-lists or black lists (Known Bad Records), as well as the large unlabeled data for machine learning.

Second, in the world of cyber security, often the amount of normal traffic will be the majority of traffic records, while malicious traffic or records are the minority, or considered anomaly condition. The author proposed to take into consideration this class-imbalanced nature of cyber security data and apply appropriate methodology to mitigate the majority class bias.

By providing theoretical proof, the author described a novel tri-training method for semi-supervised machine learning study on cyber-security data. The goal is to better utilize the large amount of unlabeled data currently available, in addition to the limited training data from “Known Bad Records”.

Armed with this new tool, the author aims to help cyber-security professionals to turn the tide in the battle against adversaries often equipped with new tactics, techniques, and procedures.

For future work, the author described experiments necessary to provide evidence for the theoretical proofs for this new method, as well as the appropriate metrics to measure the success.

For longer term research, deep learning techniques have shown great potential in general Artificial Intelligence applications and could be adapted for cyber-security applications as well.

References:

Aburomman, A. A., & Reaz, M. B. (2017). A survey of intrusion detection systems based on ensemble and hybrid classifiers. *Computers & Security*, 65, 135-152. doi:10.1016/j.cose.2016.11.004

Aggarwal, P., & Sharma, S. K. (2015). Analysis of KDD Dataset Attributes - Class wise for Intrusion Detection. *Procedia Computer Science*, 57, 842-851. doi:10.1016/j.procs.2015.07.490

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the eleventh annual conference on Computational learning theory - COLT' 98*. doi:10.1145/279943.279962

Chapelle, O., Schölkopf, B., & Zien, A. (2010). *Semi-supervised Learning*. Cambridge, MA: Mit Press.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321-357. doi:10.1613/jair.953

Davis, J. J., & Clark, A. J. (2011). Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, 30(6-7), 353-375. doi:10.1016/j.cose.2011.05.008

Drago, A. (2018, July 27). Russian Hackers Appear to Shift Focus to U.S. Power Grid. *New York Times*. Retrieved from <https://www.nytimes.com/2018/07/27/us/politics/russian-hackers-electric-grid-elections-.html>

Ertöz, L., Steinbach, M., & Kumar, V. (2003). Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data. Proceedings of the 2003 SIAM International Conference on Data Mining, 47-59.

Guo, H., Li, Y., Shang, J., Gu, M., Huang, Y., Gong, B. (2017) Learning from Class-imbalanced Data: Review of Methods and Applications. *Expert Systems with Applications*, 220-239.

Jarvis, R. A., Patrick, E. A., (1973) Clustering Using a Similarity Measure Based on Shared Near Neighbors. *IEEE Transactions on Computers*, Vol. C-22, No. 11, 1025-1034.

KDD Cup 1999 Data. (1999, October 28). Retrieved from http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4), 221-232. doi:10.1007/s13748-016-0094-0

Kronz, A. (n.d.). Hype vs. Reality of AI, Data Science and Machine Learning. Retrieved from <https://www.gartner.com/webinar/3894964?ref=mrktg-srch>. Retrieved on 2/4/2019.

Li, J., Zhang, W., & Li, K. (2010). A Novel Semi-supervised SVM Based on Tri-training for Intrusition Detection. *Journal of Computers*, 5(4). doi:10.4304/jcp.5.4.638-645

Link Analysis and Crime - An examination. (2015, December 17). Retrieved from https://crimetechweekly.com/2015/12/17/link-analysis-and-fraud/

Moustafa, N., & Slay, J. (2015). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). 2015 *Military Communications and Information Systems Conference (MilCIS)*. doi:10.1109/milcis.2015.7348942

Moustafa, N., & Slay, J. (2016). The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 25(1-3), 18-31. doi:10.1080/19393555.2015.1125974

Patcha, A., & Park, J. (2007). Network anomaly detection with incomplete audit data. *Computer Networks*, 51(13), 3935-3955. doi:10.1016/j.comnet.2007.04.017

Ruder, S., & Plank, B. (2018). Strong Baselines for Neural Semi-supervised Learning under Domain Shift. (n.d.). Retrieved from https://arxiv.org/abs/1804.09530.

Samatova, N. F., Hendrix, W., Jenkins, J., Padmanabhan, K., & Chakraborty, A. (2013). Practical Graph Mining with R. Boca Raton, FL: CRC Press, 135-166. <https://www.csc2.ncsu.edu/faculty/nfsamato/practical-graph-mining-with-R/slides/pdf/Graph_Cluster_Analysis.pdf>

Silverstein, J. (2019, January 16). Is the "10 Year Challenge" on Facebook a privacy scheme disguised as a meme? Retrieved from https://www.cbsnews.com/news/facebook-10-year-challenge-meme-could-it-mine-your-data-facial-recognition/

Søgaard, A., & Rishøj, C. (2010). Semi-supervised dependency parsing using generalized tri-training. *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, 1065-1073. Downloaded from https://www.aclweb.org/anthology/C10-1120

Smith, R., & Barry, R. (2019, January 10). America’s Electric Grid Has a Vulnerable Back Door—and Russia Walked Through It. *The Wall Street Journal.* Retrieved from <https://www.wsj.com/articles/americas-electric-grid-has-a-vulnerable-back-doorand-russia-walked-through-it-11547137112>

Tan, K., Zhu, J., Du, Q., Wu, L., & Du, P. (2016). A Novel Tri-Training Technique for Semi-Supervised Classification of Hyperspectral Images Based on Diversity Measurement. *Remote Sensing*, 8(9), 749. doi:10.3390/rs8090749. Downloaded from https://www.mdpi.com/2072-4292/8/9/749.

Zhou, Z., & Li, M. (2005). Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11), 1529-1541. doi:10.1109/tkde.2005.186