# Implications of Combining Domain Knowledge in Explainable Artificial Intelligence

Sheikh Rabiul Islam[a], William Eberle[b]

[a]University of Hartford, 200 Bloomfield Ave, West Hartford, CT 06117
[b]Tennessee Technological University, 1 William L Jones Dr, Cookeville, TN 38505

**Abstract**

Although the infusion of domain knowledge in explainable artificial intelligence techniques is a viable approach to enhance the explainability of "black box" model's decision, there are some open challenges. Some of the challenges include quantification of explainability, compromise in performances, and information sacrifices. In our prior work, we demonstrated that the infusion of domain knowledge in network intrusion detection provides better explainability of decisions, better generalization to work well with unknown attacks, and a faster decision or response. In this paper, we extend our prior work to quantify the level of information sacrifices from the introduces generalization, and quantify the level of explainability in an explainable artificial intelligence technique applied to a network intrusion detection problem. Our experimental results suggest that, as a result of domain knowledge infusion, the level of information sacrificed is very negligible, and the explainability score, using a recently proposed proxy method, is better than the case of not using domain knowledge.

**Keywords**

explainable artificial intelligence, domain knowledge infusion, explainability quantification

## 1. Introduction

Artificial Intelligence (AI) based "black box" models lack explainability even though they show superior performance in many applications. Furthermore, the lack of explainability leads to a lack of trust in the model and prediction, which can involve ethical and legal issues in critical domains due to the potential implications on human interests, rights, and lives (e.g., credit approval in finance, killer robots in defense, etc.). To mitigate the unethical use of AI as well as to promote the responsible use of AI systems, various governments have started taking different precautionary initiatives. Recently, the European Union implemented the rule of "right of explanation", where a user can ask for an explanation of the algorithmic decision [1]. In addition, more recently the U.S. government introduced a new bill, the "Algorithmic Accountability Act", which would require companies to assess their machine learning systems for bias and discrimination, with a need to take corrective measures [2]. The U.S. Department of Defense (DoD)

*In A. Martin, K. Hinkelmann, H.-G. Fill, A. Gerber, D. Lenat, R. Stolle, F. van Harmelen (Eds.), Proceedings of the AAAI 2021 Spring Symposium on Combining Machine Learning and Knowledge Engineering (AAAI-MAKE 2021) - Stanford University, Palo Alto, California, USA, March 22-24, 2021.*

shislam@hartford.edu (S.R. Islam); weberle@tntech.edu (W. Eberle)

has identified explainability as a key stumbling block in the adoption of AI-based solutions in many of their projects [3, 4].

Network intrusions are a common cyber-crime activity, estimated to cost around $6 trillion annually in damages by 2021 [5]. To combat these attacks, an Intrusion Detection System (IDS) is a security system to monitor network and computer systems [6]. Research in AI-based IDSs has shown promising results [6], [7], [8], [9], [10], and has become an integral part of security solutions due to its capability of learning complex, nonlinear functions and analyzing large data streams from numerous connected devices. A recent survey by [11] suggests that deep learning-based methods are accurate and robust to a wide range of attacks and sample sizes. However, there are concerns regarding the sustainability of current approaches (e.g., intrusion detection/prevention systems) when faced with the demands of modern networks and the increasing level of human interaction [7]. In the age of IoT and Big Data, an increasing number of connected devices and associated streams of network traffic have exacerbated the problem. In addition, the delay in detection/response increases the chance of *zero day exploitation*, whereby a previously unknown vulnerability is just discovered by the attacker, and the attacker immediately initiates an attack. However, improved explainability of an AI model could quicken interpretation, making it more feasible to accelerate the *response*. In prior work [12], we introduce a novel approach for an AI-based explainable intrusion detection and response system, and demonstrate its effectiveness by infusing a popular network security principle (CIA principle) into the model for better explainability and interpretability of the decision.

However, domain knowledge incorporation involves feature engineering, and the amount of information that is being sacrificed is still not clear. In addition, there is a need to realize the impact of lost information due to feature engineering which is key to success in applied Machine Learning. Also, the performance of predictive modeling depends on the chosen algorithm/model, available data, and the features used. To mitigate the quantification of and the impact of lost information, we introduce the use of Principal Component Analysis (PCA), which is a feature extraction technique that creates a new smaller set of features that still captures most of the information of the data. In summary, PCA finds the directions of maximum variance (i.e., the direction of maximum data dispersion using the Eigenvector and the importance of the direction using the Eigenvalue) in high-dimensional data and project the data onto a smaller subspace (i.e., smaller set of new features) while retaining most of the information in the raw data.

Quantification of explainability is another open challenge, and a known limitation in prior work [12]. In this paper, we extend that work to quantify the level of explainability for intrusion detection and response, adapting from the work [13] for credit default prediction.

In summary, our main contributions in this work are as follows: (1) we extend our prior work on explainable AI for intrusion detection to quantify and analyze the level of information loss from the introduced generalization, and (2) we quantify the level of explainability in the previously proposed XAI system for intrusion detection and response.

We start with a background of related work (Section 2) followed by a description of the proposed approach, an intuitive description of algorithms, and an overview of the dataset (Section 3) used in this work. In Section 4, we describe our experiments, followed by Section 5 which contains a discussion on results from the experiments. We conclude with limitations and future work in Section 6.

## 2. Background

Research in Explainable Artificial Intelligence (XAI) is a re-emerging field following some earlier work of [14], [15], and [16]. Previous work focused on primarily explaining the decision process of knowledge-based systems and expert systems. The main reason for the renewed interest in XAI research has stemmed from recent advancements in AI and ML and their application to a wide range of areas, as well as concerns over unethical use and undesired biases in the models. In addition, recent concerns and laws by different governments are necessitating more research in XAI.

AI-based IDSs have continued to show promising performance [6],[7],[8],[9],[10]. However, there are still the problems of long training times and a reliance on a human operator [7]. Incorporating domain knowledge for explainability has gotten little attention. Previously, we introduced the concept of infusing domain knowledge [12] for network intrusion detection.

There are two primary directions of research towards the evaluation of the explainability of an AI/ML model: (1) human study-based, and (1) model complexity-based. Human study-based explainability evaluation is an open challenge. [17] argue that interpretability is not an absolute concept; rather, it is relative to the target model, and may or may not be relative to the human. Their finding suggests that a model is readily interpretable to a human when it uses no more than seven pieces of information [18]. However, this might vary from task to task and person to person. For instance, a domain expert might consume a lot more detailed information depending on their experience. Much attention is still needed for quantifying the explainability with human subjects.

In the literature, model complexity and (lack of) model interpretability are often treated as the same [19]. For instance, in [20] and [21], model size is often used as a measure of interpretability (e.g., number of decision rules, depth of the tree, number of non-zero coefficients). More recently, [19] attempts to quantify the complexity of the arbitrary machine learning model with a model agnostic measure. In that work, the author demonstrates that when the feature interaction (i.e., the correlation among features) increases, the quality of representations of explainability tools degrades. In fact, from our study of different explainability tools (e.g., LIME, SHAP, PDP), we have found that the correlation among features is a key stumbling block to representing feature contribution in a model agnostic way. Keeping the issue of feature interactions in mind, [19] proposes a technique that uses three measures: number of features, interaction strength among features, and the main effect (excluding the interaction part) of features to measure the complexity of a post-hoc model for interpretation. While their work focuses primarily on post-hoc models, we use their approach as a precursor to formulate our explainability quantification. In addition, our approach is model agnostic and targets any notion (e.g., pre-modeling, post-hoc).

## 3. Methodology

### 3.1. Proposed Approach

In previous work [12], the proposed approach for combining domain knowledge consists of two components: a *feature generalizer*, which gives a generalized feature set with the help of

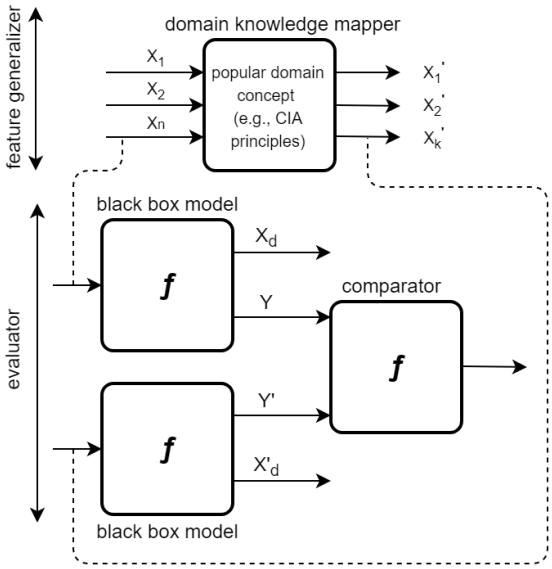


**Figure 1:** Proposed Technique

domain knowledge in two different ways; and an *evaluator* that produces and compares the results from the "black box" model for multiple configurations of features: domain knowledge infused features, newly constructed features from domain knowledge infused features, selected features, and all features.

## 3.2. Feature Generalizer

The *feature generalizer* (Figure 1, top portion), takes original features of the dataset ($X_1$, $X_2$, .... $X_n \in X$ where $X$ is the set of all features) and infuses domain knowledge to produce/reconstruct a concise and better interpretable feature set ($X_1$', $X_2$', ..... $X_k$' $\in X$' where $X$' is the universal set of original/transformed/constructed features, but here $k$ is much smaller than $n$) in two different ways:

### 3.2.1. Feature Mapping

We use CIA principles as domain knowledge, which stands for *confidentiality*, *integrity*, and *availability*. We analyze all types of attacks for associated compromises in each component of CIA principles (see Table 2 in Appendix). For example, a *Heartbleed* vulnerability is related to a compromise in *confidentiality* as an attacker could gain access to the memory of the systems protected by the vulnerable version of the OpenSSL; a *Web attack* (e.g., Sql injection) is related to a compromise in *confidentiality* and *integrity* (e.g., read/write data using injected query); and flooding a database server with injected complex queries (e.g., a cross join), would constitute an *availability* compromise. Another example is an *Infiltration* attack, which is related to a compromise in *confidentiality* as it normally exploits a software vulnerability (e.g., Adobe Acrobat Reader) to create a backdoor and reveal information (e.g., IP's). *Port scan* attack

is related to a compromise in *confidentiality* as the attacker sends packets with varying destination ports to learn the services and operating systems from the reply. All *DoS* and *DDoS* attacks are related to a compromise in *availability* as it aims to hamper the availability of service or data. Furthermore, *SSH patator* and *FTP patator* are brute force attacks and are usually responsible for a compromise in *confidentiality*. For instance, a botnet (i.e., robot network—a network of malware-infected computers) could provide a remote shell, file upload/download option, screenshot capture option, and key logging options which has potential for all of the *confidentiality, integrity, and availability* related compromises.

From the feature ranking of the public CICIDS2017 dataset [22], for each type of attack, we take the top three features according to their importance (i.e., feature importance from Random Forest Regressor) and calculate the mapping with related compromises under the CIA principles. For example, the feature *Average Packet Size* is renamed as *Avg Packet Size - A* where -A indicates that it is a key feature for the compromise of *availability*. To get this mapping between features and associated compromises, we first find the mapping between an attack and related compromises (from Table 2 in Appendix), formulated as Equation 7). In other words, Formula 1 gives the name of the associated attack where the feature is in the top three feature to identify that particular attack and Formula 2 gives associated compromises in C, I, or A from the attack name. Thus, with the help of domain knowledge, we keep 22 features out of a total of 78 features. We will refer to these features as the *domain features*.

$$f(feature) \longrightarrow attack \tag{1}$$

$$f(attack) \longrightarrow C, I, orA \tag{2}$$

### 3.2.2. Feature Construction

We also construct three new features, C, I, and A, from the domain features by quantitatively calculating compromises associated with each of the domain features. For that purpose, we calculate the correlation coefficient vector of the dataset to understand whether the increase in the value of a feature has a positive or negative impact on the target variable. We then convert the correlation coefficient (a.k.a., *coeff*) vector V in to a 1 or -1 based on whether the correlation coefficient is positive or negative accordingly. We also group the domain features and corresponding *coeff* tuple into three groups. Using formula 3, 4, and 5, we aggregate each group (from C, I, and A) of domain features into the three new features C, I, and A. We also scale all feature values from 0 to 1 before starting the aggregation process. During the aggregation for a particular group (e.g., C), if the correlation coefficient vector (e.g., $V_i$) for a feature (e.g., $C_i$) of that group has a negative value, then the product of the feature value and the correlation coefficient for that feature is deducted, and vice-versa if positive. In addition, when a feature is liable for more than one compromise, the feature value is split between the associated elements of CIA principles.

$$C = \sum_{i=0}^{n} C_i V_i \tag{3}$$

$$I = \sum_{i=0}^{n} I_i V_i \tag{4}$$

$$A = \sum_{i=0}^{n} A_{\mathrm{i}} V_{\mathrm{i}} \tag{5}$$

### 3.3. Evaluator

The task of the *evaluator* (Figure 1, bottom side) is to execute (supervised models or algorithms) and compare the performance (in detecting malicious and benign records) of four different types of configurations of features: (1) using all features, (2) using selected features (selection is done by feature selection algorithm), (3) using domain knowledge infused features, and (4) using newly constructed features C, I, and A from domain knowledge infused features. In addition, the *evaluator* performs the following two tests:

1. Explainability Test: The purpose of this test is to discover the comparative advantages or disadvantages of incorporating domain knowledge in the experiment; and
2. Generalizability Test: The purpose of this test is to analyze how different approaches perform in unknown or unseen attack detection. All training records for a particular attack are deleted one at a time, and the performance of the model is evaluated on the same test set, which includes records from unknown or unseen attacks. Details of these tests are described in Section 4.

### 3.4. Algorithms

We use six different standard algorithms for predicting malicious records: one of those is a probabilistic classifier based on Naive Bayes theorem, and the remaining five are supervised "black box" models : Artificial Neural Network (ANN), Support Vector Machine (SVM), Random Forest (RF), Extra Trees (ET), and Gradient Boosting (GB). In addition, we use Principal Component Analysis (PCA) for quantification of information sacrifices, and a proxy-based explainability quantification method for quantification of explainability.

#### 3.4.1. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a feature extraction technique that creates a new smaller set of features that still captures most of the information of the data. In other words, PCA finds the directions of maximum variance (i.e., the direction of maximum data dispersion using Eigenvector and importance of the direction using Eigenvalue) in high-dimensional data and project the data onto a smaller subspace (i.e., smaller set of new features) while retaining most of the information in the raw data. So, PCA is a viable candidate to realize the level of information loss from the feature engineering. Therefore, we use PCA to demonstrate the information loss induced from the combination of domain knowledge for better explainability.

#### 3.4.2. Explainability Quantification Method

In our prior work [13], we propose a proxy task-based explainability quantification method for XAI in credit default prediction. In this work, we apply that approach for quantification of explainability in XAI for intrusion detection. In fact, along with the previously demonstrated

field of finance, and in the paper with cyber-security, there is nothing about our proposed explainability quantification method that should prohibit it from being applied to other domains. A proxy task-based explainability quantification method considers different properties of output representation (e.g., depth of decision tree, length of rule list) as a metric for evaluation. Usually, humans can relate and process 7+-2 pieces of information (i.e., cognitive chunks) to understand something [18]. For instance, suppose that, in the most generalized form, the quality of an explanation is dependent upon the number of cognitive chunks that the recipient has to relate to understanding an explanation (i.e., the less, the better). Lets assume, E = explainability score; $N_c$ = number of cognitive chunks; I = interaction; $N_i$ = number of input cognitive chunks; and $N_o$ = number cognitive chunks involved in the explanation representation (i.e., output cognitive chunks).

$$E = \frac{1}{N_c} \tag{6}$$

However, sometimes, these cognitive chunks are correlated and their influences are not mutually exclusive. This interaction among cognitive chunks complicates the explainability. So we penalize Formula 6 for having an interaction among cognitive chunks, resulting in Formula 7.

$$E = \frac{1}{N_c} + (1 - I) \tag{7}$$

Where, the interaction $I$ ranges in between 0 and 1, and the less the interaction, the better the explainability, so we take the complement of that.

Furthermore, both the number of input cognitive chunks in the model and the number of output cognitive chunks involved in the representation of output are important to understand the causal relationship, which is vital for explanation. While the ideal explainability case would be when there is only one input and one output cognitive chunk (no chance of interaction), that is unusual in real-world situations. Following the segregation of input and output cognitive chunks, Formula 7 can be re-written as Formula 8:

$$E = \frac{1}{N_i} + \frac{1}{N_o} + (1 - I) \tag{8}$$

where $N_i$ refers to the number of input cognitive chunks and $N_o$ refers to the number of cognitive chunks involved in the explanation representation (i.e., output cognitive chunks). Usually, the more these cognitive chunks, the more complicated the explanation becomes. So, the ratio of the best possible case (i.e., one cognitive chunk) and the observed case is added towards total explainability.

After the addition of the weight terms for each of three predicates, Formula 8 becomes Formula 9:

$$E = \frac{w_1}{N_i} + \frac{w_2}{N_o} + w_3(1 - I) \tag{9}$$

Formula 9 can then be used to quantify the explainability of the explanation method (i.e., global explainability). We can use Formula 9 to also quantify the explainability of an instance level prediction (i.e., local explainability). In that case, the first predicate of Formula 9 (including the weight term) remains the same (i.e., the same number of input chunks). However, predicate 2 and predicate 3 will be different from instance to instance as a different set of cognitive chunks

with different interaction strengths might be involved in the representation of explanation for a particular instance as explanations are selective.

### 3.5. Data

In this work, we use the recent and comprehensive IDS dataset CICIDS2017, published in 2017, that contains various attacks: DoS, DDoS, Brute Force, XSS, SQL Injection, Infiltration, Portscan, and Botnet. In fact, this dataset was created to eliminate the shortcomings (e.g., lack of traffic diversity and volume, lack of variety of attacks, anonymized packet information, and out of date) of previous well known IDS datasets such as DARPA98, KDD99, ISC2012, ADFA13, DEFCON, CAIDA, LBNL, CDX, Kyoto, Twente, and UMASS since 1998. This is a labeled dataset containing 78 network traffic features (some features are listed in our prior work [13]) extracted and calculated from pcap files using CICFlowMeter software [23] for all benign and intrusive flows [22]. This new IDS dataset includes seven common attacks satisfying real-world criteria, publicly available at https://www.unb.ca/cic/datasets/ids-2017.html .

## 4. Experiments

### 4.1. Experimental Setup

We execute the experiments on a GPU enabled Linux machine with 12GB of RAM and a core i7 processor. All supervised machine learning algorithms are implemented using the Python-based *Scikit-learn* [24] library. In addition, we use *Tensorflow* [25] for the Artificial Neural Network. Due to resource limitations, instead of using the whole dataset, we take a stratified sample of the data which is big enough (i.e., 300K records) for a single GPU enabled commodity machine. We make the sampled dataset available to the research community at [26]. Furthermore, we use 70% of the data for training the models and kept 30% of the data as a holdout set to test the model. We confirm the target class had the same ratio in both sets. To avoid the adverse effect of class imbalance in classification performance, we re-sample the minority class of the training set using SMOTE [27] to balance the dataset. However, we do not re-sample the test set, as real-world data is skewed, and oversampling the test set could exhibit an over-optimistic performance.

We run all supervised machine learning algorithms using five different approaches:

1. With all features: using all 78 features of the dataset without discarding any features.
2. With selected features: using *Random Forest Regressor* (adapting the work of [22]) to select important features of the dataset, giving us 50 important features having a nonzero influence on the target variable;
3. With domain knowledge infused features: using infused domain knowledge features (see Section 3.2.1), we will use the term *domain features* interchangeably to express it in short form;
4. With newly constructed features from domain knowledge infused features: using newly constructed features C, I, and A (see Section 3.2.2) from domain knowledge infused features, we will use the term *domain features-constructed* interchangeably to express it in short form; and

5. With an increasing number of Principal Components: using different combinations of *principal components*, in increasing order, as a feature set.

The following are two types of experiments using each of the first four feature settings. The last feature setting, *principal component* as features, is primarily used to measure the information sacrifices.

### 4.2. Explainability Test

For this test, we run six supervised algorithms RF, ET, SVM, GB, ANN, and NB using the four described feature settings and report the results in Section 5.1. Unlike NB, other classifiers are "black box" in nature. NB is a probabilistic classifier based on Bayes Theorem with a strong conditional independence assumption among features.

### 4.3. Generalizability Test

Domain knowledge infusion is capable of enhancing explainability with the added benefit of enhancing generalizability. For testing the generalizability of the approach, we train the classifier without the representative of a particular attack, and test it with the presence of the representative of that particular attack, in order to classify it as malicious/benign. To be more specific, we delete all records of a particular attack from the training set, train the classifier with the records of the remaining 13 attacks, and test the classifier with all 14 attacks. We report the percentage of deleted attacks that are correctly detected as malicious (see Section 5.2). We repeat this one by one for all 14 attacks. We make the source code available to the research community to replicate the experiments at [28].

## 5. Results

The following sections discuss results from the two categories of experiments that we performed in our prior work [13], and from the new experiments for quantification of (1) information sacrifice, and (2) explainability.

### 5.1. Findings from Explainability Test

Comparing the performance using *all features* vs *selected features*, Table 3 (see Appendix) shows that models using *all features* (denoted with an appended -A, for instance RF-A) tend to show better results in terms of all performance metrics. However, while the difference with the *selected features* setting is negligible (<.0007 for RF) for any performance metric, that might be a result of the elimination of features with little significance. In addition, Random Forest outperforms other algorithms SVM, ET, GB, ANN, and NB under this feature setting (i.e., using all features). So we consider the results using all features as a baseline to compare against our proposed approach.

Before comparing the results from our proposed approach with the baseline (i.e., using all features), we seek the best feature setting among two domain related feature settings: domain knowledge infused features and newly constructed features. In other words, in our attempt

to find the better approach among using domain knowledge infused features vs newly constructed features (C, I, and A) from domain knowledge infused features, we find that, in almost all cases, the model with domain knowledge infused features (denoted with an appended -D1, for instance RF-D1) performs better than the counterpart (see Table 4 in Appendix). Although for RF, the maximum performance gap is .2 in the recall, for ET that gap is .048 with a similar precision. As the domain features (22 features) contain a lot more detail than the newly constructed features C, I, and A (3 features), it loses few details. In terms of individual algorithms, RF is again a clear winner this time using domain features. Although NB and ANN exhibit better recall using constructed features, it comes with compromises in precision. So, overall we consider the domain features setting as the best over the constructed features.

While we know the best feature setting is the *all features*, as shown in the comparison of *all features* vs *selected features* in the Table 3 (see Appendix), we also know that *domain features* is the best feature setting from the comparison of *domain features* vs *constructed features* (see Table 4 in Appendix). So we further compare the performance of models using the two best settings *all features* (i.e., baseline) vs *domain features*. We find that, among all models, RF using all features (denoted with an appended -A, for instance, RF-A) performs better than all other algorithms (see Table 5 (see Appendix) and Figure 5 (see Appendix). Interestingly, RF using domain knowledge infused features (denoted with an appended -D1, for instance, RF-D1) also shows promising performance. The difference between these two in terms of any performance metrics is negligible (<.005). In fact, the result of RF using the domain knowledge infused feature settings is better than what [22] reports using the same dataset. The slight improvement might stem from the experimental settings (e.g., training and test set split, re-sampling techniques). Furthermore, in the domain knowledge infused feature setting we are using only 22 features out of 78 total, where each feature indicates the associated compromises (e.g., confidentiality, integrity, or availability), capable of producing better explainable and interpretable results compared to the counterpart. The prediction for a particular sample can be represented as:

$$P(D) = b + \sum_{g=0}^{G} contribution(g) \tag{10}$$

where b is the model average and g is the generalized domain feature (e.g., ACK Flag Count - C), P(D) is the probability value of the decision. Instead of using contributions from each of the domain features, we can express the output in terms of the contribution from each element of the domain concept. For that, we need to aggregate contributions from all features into three groups (C, I, and A). This will enable an analyst to understand the nature of the attack more quickly (see Figure 4 in Appendix). For instance, when the greater portion of a feature contribution for a sample is from features tagged with -A (i.e., *Availability*) then it might be a DDoS attack, which usually comes with very high compromises in *availability* of data or service. We use the *iml* package from the programming language *R* to generate the breakdown of feature contributions of a particular sample's prediction (see Figure 4 in Appendix).

## 5.2. Findings from Generalizability Test

The purpose of the generalizability test is to test the resiliency against unknown attacks. First, we use Random Forest (RF), the best performing algorithm so far, using all four settings of

features. As shown in Table 6 (see Appendix) and Figure 6(see Appendix), we see that except for the constructed feature settings (denoted by Dom-Cons. in Figure 6, Appendix), the performances of other feature settings (all, selected, and domain) are similar. The constructed features fail to provide comparable performance for RF as it has only three features and loses data details (i.e., too much generalization). Surprisingly, a few unknown attacks are only detectable using the domain knowledge infused features. For instance, *Web Attack Sql Injection* is detected as suspicious only by domain knowledge infused features. Overall, although the *domain knowledge infused* feature setting performs slightly worse than the *all feature* setting, it comes with explainable features set with the added capability of identifying a few unknown attacks.

To reiterate, the constructed features set consists of only three features (C, I, and A) constructed from aggregating domain knowledge infused features. As this feature setting is composed of only three features, it is an extreme generalization of features and loses a lot of data details. However, this time it comes with unique capabilities that are realized after applying a statistical approach (Naive Bayes) on the dataset. We find that (see Table 7 in Appendix), for NB, the newly constructed feature setting is best as NB is also able to detect unknown attacks with similar accuracy compared to other feature settings by RF in Table 6 (see Appendix). The most interesting thing about this capability is that this feature set is composed of only three features (C, I, and A), takes comparatively less time to execute, and comes with the added benefit of very good explainability. Once the prediction is expressed as a percentage of influence from each of C, I, and A, the analyst would be able to perceive the level of compromise more intuitively from the hints about the type of attack (e.g., DDoS will show a high percentage of A—compromise in Availability).

However, from Table 3 (see Appendix), Table 4(see Appendix), and Table 5(see Appendix), we can see that NB's performance comes at a cost of precision and recall (i.e., produces comparatively more false positives and false negatives). In addition, NB is a bad probability estimator of the predicted output [29]. However, NB with a constructed features setting could be recommended as an additional IDS for quick interpretation of huge traffic data given the decision is treated as tentative with the requirement of a further sanity check. We also calculate the average time taken by each algorithm for all four feature settings and found that NB is the fastest algorithm. RF, ET, GB, ANN, and SVM take 2.80, 9.27, 77.06, 15.07, and 444.50 times more execution time compared to NB. Besides, the best algorithm, RF ($1^{st}$ in terms of the performance metric and $2^{nd}$ in terms of execution time), can be executed in parallel using an *Apache Spark* for a far better run-time [30] making it highly scalable to big data problems.

Overall, domain knowledge infusion provides better explainability with negligible compromises in performance. In addition, the generalization provides better execution time and resiliency with unknown attacks.

### 5.3. Information Sacrifices from Explainability

Domain knowledge incorporation involves feature engineering, and the amount of information that we are sacrificing is still not clear.

In our proposed approach, we use different techniques for feature engineering: (a) the domain mapped features approach falls under the **feature selection**, and (b) the newly con-

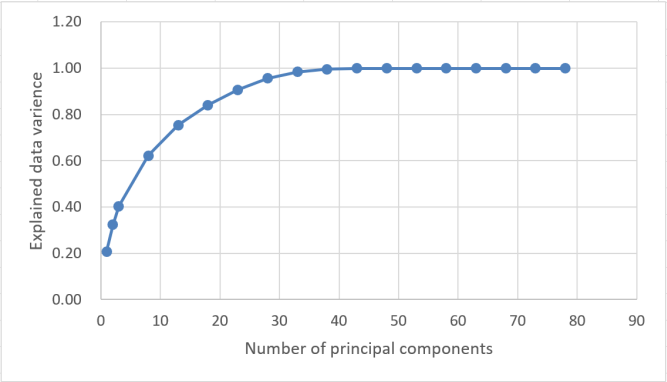


**Figure 2:** Variance ratio and cumulative variance for different principal components.

structed features approach falls under the **feature construction**. Furthermore, we leverage Principal Component Analysis (PCA), which is a feature extraction technique that creates a new smaller set of features that still captures most of the information of the data. The blue bottom line in Figure 2 shows the ratio of variance (ie., percentage of variance) covered by an increasing number of principal components (PC). The blue line shows the variance covered by the number of principal components starting from 1 PC to 78 PCs. Once we reached to 38 PCs, the covered variance was 100%, and following that, with the increase of PCs, there were no changes in the covered variance as 38 PCs were enough to cover 100% variance in the data. Therefore, we need at least 38 PCs to explain the 100% variance in the data.

Figure 3 shows the comparison of performance using different feature engineering techniques used in this paper. For PCA, we are using the first 38 principal components that explain 100% of the data variance (Figure 2) projected by PCA. From Figure 3, we can see that in terms of all metrics, PCA, Original, and Domain features show the same performance (.99) up to 2 decimal digits, and following two decimal digits, PCA minimally outperforms other approaches. On the other hand, the Constructed feature setting performs slightly worse (by 4% for accuracy, 5% for precision, 20% for recall, and 4% for AUC) compared to other features settings. In the constructed features, we are drastically reducing the number of features to 3, and is a reason behind the compromise in performance. Although PCA features lose an explainability of features, it provides an estimation of the present information in the raw data irrespective of the target of the problem.

In the end, our proposed approach (domain mapped feature) provides the competitive accuracy, precision, recall, F-score among all these feature engineering techniques, which is also similar to PCA (up to two decimal digits). So, we can conclude that our approach retains enough valuable information and does not trade valuable information to achieve explainability.

Overall, even though infusing domain knowledge might lead to some compromise in performance, clearly it ensures better explainability and interpretability as the output is made from a concise and familiar set of features from the domain.

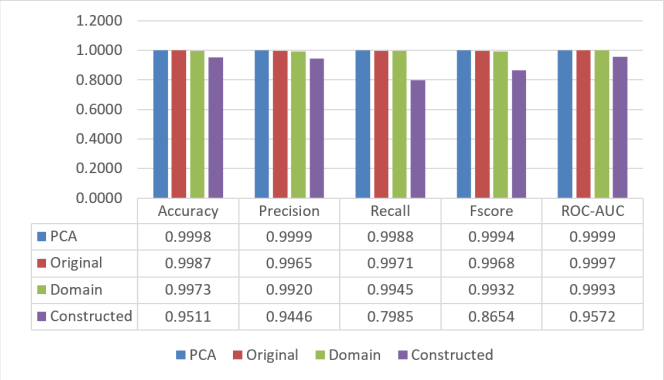


**Figure 3:** Comparison of performance using different features settings.

**Table 1**
Comparison of explainability

| | Original | Domain | Constructed |
|---|---|---|---|
| Input chunks ($N_i$) | 78 | 22 | 3 |
| Output chunks ($N_o$) | 78 | 5 | 3 |
| Int. Strength (I) | 1 | 0.8083 | 0.6803 |
| Explainability (E) | 0.0085 | 0.1450 | 0.3284 |

## 5.4. Explainability Quantification

We represent the predicted output as a composition of individual elements of the domain principle CIA (i.e., *newly constructed features*) (Figure 4 in Appendix).

One will notice that the representation (Figure 4 in Appendix) of the prediction in terms of the *newly constructed features* provides better explainability as the final prediction is segregated into the individual influences of a very concise set of intuitive features (i.e., 3 compared to 78). However, there is no way to quantify the level of perceived explainability. To use our proposed formula in this work (Formula 9) to quantify explainability, we need to calculate the *interaction strength* ($I$) too. We measure the interaction strength among features using R's iml package that uses the partial dependence of an individual feature as the basis for calculating *interaction strength* ($I$).

Applying Formula 9, on metadata (Table 1) of three different feature settings, we see that *newly constructed features* (CIA principal) provide the best explainability score of 0.3284, which is a considerable improvement compared to the 0.0085 that we get using the *original features* (Table 1). In fact, even if we apply the state-of-the-art methods of post-hoc interpretability/explainability like SHAP, the explainability will be still limited to 0.0085 as it does not reduce the number of cognitive chunks to represent output. Besides, using *domain related features*, the explainability score is 0.1450, which is better than using the original features, although worse than using the *newly constructed features*.

## 6. Conclusion and Future Work

We combine domain knowledge in the "black box" system to enhance the explainability of AI-based complex model's decision. We also quantify the level of information sacrifices induced from the domain knowledge infusion. For advancing explainability quantification, which is one of the open challenges in XAI, we applied a proxy task-based explainability quantification method for network intrusion detection. Our experimental results suggest that, as a result of domain knowledge infusion, the level of information sacrifice is very negligible, and the explainability score, using a recently proposed proxy method, is better than the case of not using domain knowledge.

Going forward, as an extension of this work, we would like to address human studies in the explainability quantification, and investigate the effectiveness of domain knowledge infusion among different approaches (e.g., supervised, unsupervised, semi-supervised) for different application areas (e.g., natural language processing, image recognition).

## References


[1] B. Goodman, S. Flaxman, European union regulations on algorithmic decision-making and a "right to explanation", AI Magazine 38 (2017) 50–57.

[2] B. Wyden, Algorithmic accountability, https://www.wyden.senate.gov/imo/media/doc/Algorithmic%20 Accountability%20Act%20of%202019%20Bill%20Text.pdf, 2019.

[3] M. Turek, Explainable ai, https://www.darpa.mil/ program/explainable-artificial-intelligence, 2019.

[4] K. Rankin, The dark secret at the heart of ai, https://www.technologyreview.com/s/604087/the-dark-secret-at-the-heart-of-ai/, 2019.

[5] M. Doyle, Don't be lulled into a false sense of security, https://www.securityroundtable.org/dont-lulled-false-sense-cybersecurity/, 2019.

[6] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, R. Atkinson, Threat analysis of iot networks using artificial neural network intrusion detection system, in: 2016 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, 2016, pp. 1–6.

[7] N. Shone, T. N. Ngoc, V. D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Transactions on Emerging Topics in Computational Intelligence 2 (2018) 41–50.

[8] J. Kim, J. Kim, H. L. T. Thu, H. Kim, Long short term memory recurrent neural network classifier for intrusion detection, in: 2016 International Conference on Platform Technology and Service (PlatCon), IEEE, 2016, pp. 1–5.

[9] A. Javaid, Q. Niyaz, W. Sun, M. Alam, A deep learning approach for network intrusion detection system, in: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), ICST (Institute for Computer Sciences, Social-Informatics and ..., 2016, pp. 21–26.

[10] Z. Li, W. Sun, L. Wang, A neural network based distributed intrusion detection system

on cloud platform, in: 2012 IEEE 2nd international conference on Cloud Computing and Intelligence Systems, volume 1, IEEE, 2012, pp. 75–79.
[11] B. Dong, X. Wang, Comparison deep learning method to traditional methods using for network intrusion detection, in: 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), IEEE, 2016, pp. 581–585.
[12] S. R. Islam, W. Eberle, S. K. Ghafoor, A. Siraj, M. Rogers, Domain knowledge aided explainable artificial intelligence for intrusion detection and response, AAAI-MAKE 2020 Combining Machine Learning and Knowledge Engineering in Practice - Volume I: Spring Symposium (2020).
[13] S. R. Islam, W. Eberle, S. K. Ghafoor, Towards quantification of explainability in explainable artificial intelligence methods, AAAI Publications, The Thirty-Third International Flairs Conference (2020).
[14] B. Chandrasekaran, M. C. Tanner, J. R. Josephson, Explaining control strategies in problem solving, IEEE Intelligent Systems (1989) 9–15.
[15] W. R. Swartout, J. D. Moore, Explanation in second generation expert systems, in: Second generation expert systems, Springer, 1993, pp. 543–585.
[16] W. R. Swartout, Rule-based expert systems: The mycin experiments of the stanford heuristic programming project: Bg buchanan and eh shortliffe,(addison-wesley, reading, ma, 1984); 702 pages, 1985.
[17] A. Dhurandhar, V. Iyengar, R. Luss, K. Shanmugam, Tip: Typifying the interpretability of procedures, arXiv preprint arXiv:1706.02952 (2017).
[18] G. A. Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information., Psychological review 63 (1956) 81.
[19] C. Molnar, G. Casalicchio, B. Bischl, Quantifying model complexity via functional decomposition for better post-hoc interpretability, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2019, pp. 193–204.
[20] J. Fürnkranz, D. Gamberger, N. Lavrač, Rule learning in a nutshell, in: Foundations of Rule Learning, Springer, 2012, pp. 19–55.
[21] H. Yang, C. Rudin, M. Seltzer, Scalable bayesian rule lists, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 3921–3930.
[22] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization., in: ICISSP, 2018, pp. 108–116.
[23] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, A. A. Ghorbani, Characterization of tor traffic using time based features., in: ICISSP, 2017, pp. 253–262.
[24] Scikit-learn: Machine learning in python, https://scikit-learn.org/stable/, 2019.
[25] Tensorflow, https://www.tensorflow.org/, 2019.
[26] domain-knowledge-aided dataset, https://github.com/ SheikhRabiul/domain-knowledge-aided-explainable-ai-for -intrusion-detection-and-response/tree/master/data/ combined_sampled.zip, 2019.
[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: synthetic minority over-sampling technique, Journal of artificial intelligence research 16 (2002) 321–357.
[28] domain-knowledge-aided code, https://github.com/ SheikhRabiul/domain-knowledge-aided-explainable-ai-for-intrusion-detection-and-response, 2019.

[29] H. Zhang, The optimality of naive bayes, AA 1 (2004) 3.
[30] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, K. Li, A parallel random forest algorithm for big data in a spark cloud computing environment, IEEE Transactions on Parallel and Distributed Systems 28 (2016) 919–933.

# A. Appendix

**Table 2**
Mapping of network attack with related component of CIA principles

| Attack | Related component of CIA |
|---|---|
| DoS GoldenEye | A |
| Heartbleed | C |
| DoS hulk | A |
| DoS Slowhttp | A |
| DoS slowloris | A |
| SSH-Patator | C |
| FTP-Patator | C |
| Web Attack | C, I, A |
| Inflitration | C |
| Bot | C, I, A |
| PortScan | C |
| **DDoS** | **A** |

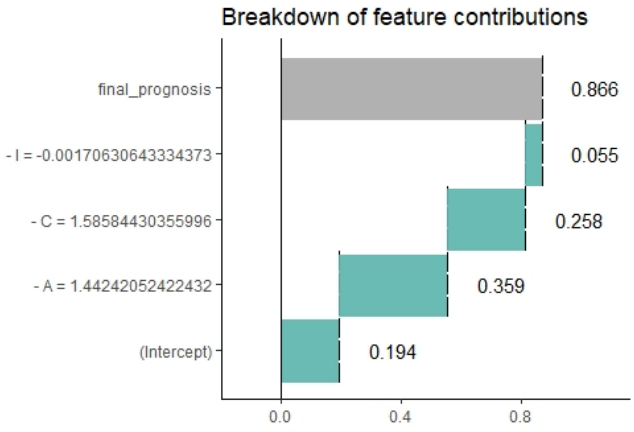


**Figure 4:** Breakdown of the prediction for a random sample in terms of C, I, and A.

**Table 3**
Performance using all features vs selected features

| Alg. | Acc. | Prec. | Rec. | F-score | AUC |
|---|---|---|---|---|---|
| RF-A | **0.9987** | **0.9965** | **0.9971** | **0.9968** | **0.9997** |
| RF-S | 0.9986 | 0.9962 | 0.9966 | 0.9964 | **0.9997** |
| *Difference* | 0.0002 | 0.0003 | 0.0006 | 0.0005 | 0.0000 |
| ET-A | 0.9981 | 0.9951 | 0.9951 | 0.9951 | 0.9994 |
| ET-S | 0.9980 | 0.9950 | 0.9950 | 0.9950 | 0.9994 |
| *Difference* | 0.0001 | 0.0002 | 0.0001 | 0.0001 | 0.0000 |
| ANN-A | 0.9802 | 0.9155 | 0.9908 | 0.9516 | 0.9984 |
| ANN-S | 0.9740 | 0.8929 | 0.9860 | 0.9372 | 0.9968 |
| *Difference* | 0.0062 | 0.0226 | 0.0047 | 0.0145 | 0.0017 |
| SVM-A | 0.9109 | 0.6996 | 0.9595 | 0.8092 | 0.9780 |
| SVM-S | 0.8869 | 0.6433 | 0.9565 | 0.7692 | 0.9746 |
| *Difference* | 0.0239 | 0.0563 | 0.0030 | 0.0400 | 0.0034 |
| GB-A | 0.9960 | 0.9854 | 0.9944 | 0.9899 | 0.9995 |
| GB-S | 0.9957 | 0.9840 | 0.9945 | 0.9892 | 0.9996 |
| *Difference* | 0.0003 | 0.0014 | (0.0001) | 0.0007 | (0.0001) |
| NB-A | 0.7753 | 0.4371 | 0.4888 | 0.4615 | 0.8601 |
| NB-S | 0.7621 | 0.4144 | 0.5019 | 0.4539 | 0.8508 |
| *Difference* | 0.0132 | 0.0228 | (0.0131) | 0.0076 | 0.0093 |

**Table 4**
Performance using domain features vs constructed features

| Alg. | Acc. | Prec. | Rec. | F-score | AUC |
|---|---|---|---|---|---|
| RF-D1 | **0.9973** | **0.9920** | **0.9945** | **0.9932** | **0.9993** |
| RF-D2 | 0.9511 | 0.9446 | 0.7985 | 0.8654 | 0.9572 |
| *Difference* | 0.0463 | 0.0475 | 0.1960 | 0.1278 | 0.0421 |
| ET-D1 | 0.9969 | 0.9913 | 0.9932 | 0.9923 | 0.9989 |
| ET-D2 | 0.9756 | 0.9321 | 0.9448 | 0.9384 | 0.9954 |
| *Difference* | 0.0214 | 0.0592 | 0.0483 | 0.0538 | 0.0036 |
| ANN-D1 | 0.9497 | 0.8300 | 0.9362 | 0.8799 | 0.9865 |
| ANN-D2 | 0.5952 | 0.3241 | 0.9721 | 0.4862 | 0.7921 |
| *Difference* | 0.3544 | 0.5059 | (0.0359) | 0.3937 | 0.1945 |
| SVM-D1 | 0.8489 | 0.5747 | 0.8968 | 0.7005 | 0.9252 |
| SVM-D2 | 0.7195 | 0.3739 | 0.6281 | 0.4687 | 0.7886 |
| *Difference* | 0.1294 | 0.2008 | 0.2687 | 0.2318 | 0.1366 |
| GB-D1 | 0.9881 | 0.9513 | 0.9904 | 0.9705 | 0.9986 |
| GB-D2 | 0.9230 | 0.7692 | 0.8701 | 0.8165 | 0.9789 |
| *Difference* | 0.0652 | 0.1821 | 0.1204 | 0.1539 | 0.0198 |
| NB-D1 | 0.7982 | 0.4881 | 0.5028 | 0.4953 | 0.8553 |
| NB-D2 | 0.5591 | 0.2687 | 0.7195 | 0.3913 | 0.6591 |
| *Difference* | 0.2391 | 0.2194 | (0.2167) | 0.1040 | 0.1962 |

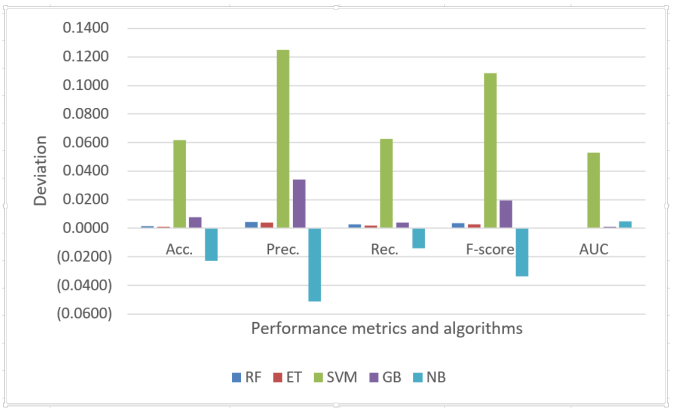


**Figure 5:** Performance deviations of using domain knowledge infused features from baseline

**Table 5**
Performance using all features vs domain features

| Alg. | Acc. | Prec. | Rec. | F-score | AUC |
|---|---|---|---|---|---|
| RF-A | **0.9987** | **0.9965** | **0.9971** | **0.9968** | **0.9997** |
| RF-D1 | 0.9973 | 0.9920 | 0.9945 | 0.9932 | 0.9993 |
| *Difference* | 0.0014 | 0.0045 | 0.0027 | 0.0036 | 0.0004 |
| ET-A | 0.9981 | 0.9951 | 0.9951 | 0.9951 | 0.9994 |
| ET-D1 | 0.9969 | 0.9913 | 0.9932 | 0.9923 | 0.9989 |
| *Difference* | 0.0011 | 0.0038 | 0.0020 | 0.0029 | 0.0004 |
| ANN-A | 0.9802 | 0.9155 | 0.9908 | 0.9516 | 0.9984 |
| ANN-D1 | 0.9497 | 0.8300 | 0.9362 | 0.8799 | 0.9865 |
| *Difference* | 0.0305 | 0.0855 | 0.0546 | 0.0717 | 0.0119 |
| SVM-A | 0.9109 | 0.6996 | 0.9595 | 0.8092 | 0.9780 |
| SVM-D1 | 0.8489 | 0.5747 | 0.8968 | 0.7005 | 0.9252 |
| *Difference* | 0.0619 | 0.1249 | 0.0627 | 0.1087 | 0.0528 |
| GB-A | 0.9960 | 0.9854 | 0.9944 | 0.9899 | 0.9995 |
| GB-D1 | 0.9881 | 0.9513 | 0.9904 | 0.9705 | 0.9986 |
| *Difference* | 0.0079 | 0.0341 | 0.0039 | 0.0194 | 0.0009 |
| NB-A | 0.7753 | 0.4371 | 0.4888 | 0.4615 | 0.8601 |
| NB-D1 | 0.7982 | 0.4881 | 0.5028 | 0.4953 | 0.8553 |
| *Difference* | (0.0229) | (0.0510) | (0.0140) | (0.0338) | 0.0048 |

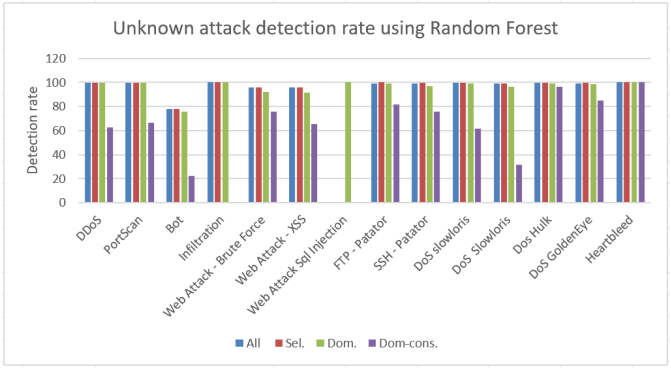


**Figure 6:** Unknown attack detection rate using Random Forest

**Table 6**
Performance of unseen attack detection using RF

| Attack | Count | All(%) | Sel.(%) | Dom.(%) | Cons.(%) |
|---|---|---|---|---|---|
| Ddos | 4184 | 99.90 | 99.90 | 99.90 | 62.86 |
| PortScan | 4973 | 99.90 | 99.94 | 99.94 | 66.28 |
| Bot | 54 | 77.78 | 77.78 | 75.93 | 22.22 |
| Infiltration | 1 | 100 | 100 | 100 | 0.00 |
| Web Attack-BF | 49 | 95.92 | 95.92 | 91.84 | 75.51 |
| Web Attack-XSS | 23 | 95.65 | 95.65 | 91.30 | 65.22 |
| Web Attack-Sql | 1 | **0.00** | **0.00** | **100** | **0.00** |
| FTP-Patator | 251 | 99.20 | 100 | 99.20 | 81.67 |
| SSH-Patator | 198 | 98.99 | 99.49 | 96.97 | 75.76 |
| DoS slowloris | 188 | 99.47 | 99.47 | 98.94 | 61.70 |
| DoS Slowloris | 174 | 99.43 | 99.43 | 96.55 | 31.61 |
| Dos Hulk | 7319 | 99.71 | 99.73 | 99.34 | 96.19 |
| DoS GoldenEye | 314 | 99.36 | 99.68 | 98.41 | 85.03 |
| Heartbleed | 1 | 100 | 100 | 100 | 100 |

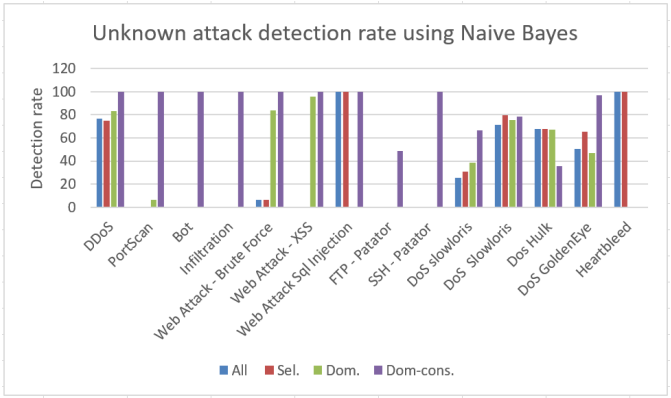


**Figure 7:** Unknown attack detection rate using Naive Bayes

**Table 7**
Performance of unseen attack detection using NB

| Attack | Count | All(%) | Sel.(%) | Dom.(%) | Cons.(%) |
|---|---|---|---|---|---|
| Ddos | 4184 | 76.94 | 74.59 | 83.22 | **100** |
| PortScan | 4973 | 0.18 | 0.18 | 6.64 | **100** |
| Bot | 54 | 0.00 | | 0.00 | **100** |
| Infiltration | 1 | 0.00 | 0.00 | 0.00 | **100** |
| Web Attack-BF | 49 | 6.12 | 6.12 | 83.67 | **100** |
| Web Attack-XSS | 23 | 0.00 | 0.00 | 95.65 | **100** |
| Web Attack-Sql | 1 | **100** | **100** | 0.00 | **100** |
| FTP-Patator | 251 | 0.00 | 0.00 | 0.00 | 48.61 |
| SSH-Patator | 198 | 0.00 | 0.00 | 0.00 | **100** |
| DoS slowloris | 188 | 25.53 | 30.85 | 38.30 | **66.49** |
| DoS Slowloris | 174 | 71.26 | **79.89** | 75.29 | 78.16 |
| Dos Hulk | 7319 | **67.78** | **67.78** | **67.03** | 35.37 |
| DoS GoldenEye | 314 | 50.32 | 65.29 | 47.13 | **96.82** |
| Heartbleed | 1 | **100** | **100** | 0.00 | 0.00 |