Algorithm Selection Framework for Cyber Attack Detection

Marc Chalé
Air Force Institute of Technology
Wright-Patterson AFB, Ohio, USA
marc.chale@afit.edu

Nathaniel D. Bastian
Army Cyber Institute
West Point, New York, USA
nathaniel.bastian@westpoint.edu

Jeffery Weir
Air Force Institute of Technology
Wright-Patterson AFB, Ohio, USA
jeffery.weir@afit.edu

ABSTRACT

The number of cyber threats against both wired and wireless computer systems and other components of the Internet of Things continues to increase annually. In this work, an algorithm selection framework is employed on the NSL-KDD data set and a novel paradigm of machine learning taxonomy is presented. The framework uses a combination of user input and meta-features to select the best algorithm to detect cyber attacks on a network. Performance is compared between a rule-of-thumb strategy and a meta-learning strategy. The framework removes the conjecture of the common trial-and-error algorithm selection method. The framework recommends five algorithms from the taxonomy. Both strategies recommend a high-performing algorithm, though not the best performing. The work demonstrates the close connectedness between algorithm selection and the taxonomy for which it is premised.

CCS CONCEPTS

• **Networks** → *Packet classification*; • **Security and privacy** → **Mobile and wireless security**; *Denial-of-service attacks.*

KEYWORDS

machine learning, algorithm selection, meta learning, feature engineering, cybersecurity

ACM Reference Format:
Marc Chalé, Nathaniel D. Bastian, and Jeffery Weir. 2020. Algorithm Selection Framework for Cyber Attack Detection. In *2nd ACM Workshop on Wireless Security and Machine Learning (WiseML '20), July 13, 2020, Linz (Virtual Event), Austria.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3395352.3402623

1 INTRODUCTION

People, organizations and communities rely on the Internet of Things (IoT) to aid in almost any conceivable task that was previously performed manually. As technology advances, components of IoT have progressed into the wireless domain [10]. These emerging systems are susceptible to attack by malicious actors wishing to degrade the system or steal proprietary information [7]. Intrusion detection systems (IDS) are central to maintaining the security of modern computer networks from malicious actors [12]. IDS have been successfully demonstrated in both the wired and wireless domain of IoT [10]. The task assigned to an IDS is to *classify* network traffic as malicious or normal. Numerous studies [10] have explored meta-models to detect malicious behavior in computer networks. Maxwell et al. [12] further focused on intelligent cybersecurity feature engineering for various meta-models.

Learning algorithms may be used to formulate a meta-model. Selection of the best machine learning (ML) algorithm, including hyper-parameters, for a particular problem instance is a difficult and time-consuming task [20]. Cui et al. [6] has confirmed conclusions of [18] and [25] that meta-models' performance varies among problem types and problem instances. Wolpert et al. [28] uses *The Extended Bayesian Formalism* to show that given a set of learning algorithms and problems, each algorithm will outperform the others for some (equally sized) subset of problems. This phenomena has driven researchers to a trial-and-error strategy of identifying the best meta-model for a given problem. The preferred meta-model is selected by comparison of model performance metrics such as accuracy [5]. Unfortunately, the computational run time and human investment required to select a learning algorithm by trial-and-error is generally prohibitive of finding the optimal choice.

This paper aims to advance the IDS body of knowledge by incorporating recent work in algorithm selection. Accordingly, an *algorithm selection framework* is introduced. The algorithm selection framework leverages a taxonomy of ML algorithms. The framework narrows down the list of applicable algorithms based on problem characterization. Two strategies are presented to select the most preferred algorithm: *rules-of-thumb* and *meta-learner*. If successful, the algorithm selection framework promotes high-performance results of the IDS and assuages the computational cost of performing multiple ML algorithms.

This paper includes Related Works in Section 2. The Methodology is presented in Section 3. Section 4 contains the Results and Section 5 is the Conclusion.

2 RELATED WORKS

2.1 Intrusion Detection

As in any other domain, criminals and adversaries seek to inflict harm by exploiting weaknesses in cybersecurity systems. The rate of system intrusion incidents is increasing annually [16]. Landwehr et al. [11] provides a taxonomy of all known flaws in computer systems. Special attention is provided to the category of flaws that allow exploitation by malicious actors. Commonly, a malicious actor, or their code, appears benign to a computer security system for a long enough time to exploit information or degrade the attacked system. The Trojan horse is among the most prevalent categories of a malicious attack on computer systems. It is characterized as a code that appears to provide a useful service but instead steals information or degrades the targeted system. A Trojan horse containing self replicating code is known as a virus. A trapdoor is a

WiseML '20, July 13, 2020, Linz (Virtual Event), Austria

ACM ISBN 978-1-4503-8007-2/20/07…$15.00
https://doi.org/10.1145/3395352.3402623

malicious attack in which an actor covertly modifies a system in such a way that they are permitted undetected access. Finally, a time bomb is an attack that accesses a system covertly and lies dormant until a detonation time. Upon detonation, the time bomb will inflict damage to the system either by disrupting service or destroying information.

Intrusion detection systems are a layer of network security that tracks activity patterns in a computer system to detect malicious actors before they can inflict harm. Debar et al. [7] describes efforts as early as 1981 and Sobirey [21] maintains a repository of prominent IDS projects. According to Debar et al. [7], the success of these systems has spawned a commercial market of IDS software including brands such as Sysco Systems, Haystack Labs, Secure Networks, among others. Typically, the IDS employs a detector module that monitors system status. The detector catalogues patterns of both normal and malicious activity in a database. The detector also monitors patterns in the current system configuration. Further, the detector provides an audit of events occurring within the system. The detector leverages these data channels to generate an alarm for suspicious activity and countermeasures if necessary. An IDS is evaluated by its *accuracy* of attack detection (false positive), *completeness* to detect all threats (false negatives) and *performance* to detect threats quickly.

NSL-KDD is a publicly available benchmark data set of network activity. NSL-KDD improves on several flaws of the well-known KDD Cup '99 benchmark data set. Most notably, NSL-KDD has rectified the 78% and 75% duplicate records in training and test sets, respectively [16]. Four classes of attacks are recorded in the data set. *Denial of service* attacks bombard a network with an overwhelming quantity of data such that the computing resources are exhausted. As a result, the system cannot fulfil any legitimate computing processes. *User to Root* attackers enter the network disguised as a legitimate user but seek security vulnerabilities which grant them elevated system privileges. A *remote to user* attack is performed by sending data to a private network and identifying insecure access points for exploitation. *Probing* is the attack technique by which the assailant studies an accessible system for vulnerabilities which will be exploited at a later time [13].

Maxwell et al. [12] and Viegas [24] describe IDS tools that incorporate ML models. Unfortunately, raw network traffic data is not a suitable input for building accurate and efficient ML models. Instead, the data must be transformed as a set of vectors representing the raw data. The process of constructing such vectors is known as feature engineering, which is a non-trivial task that requires both domain knowledge and mastery of ML to capture all available information in the model. It is shown experimentally that varying the feature engineering strategy does affect classification accuracy of the IDS but no single feature is known to be superior to others.

Kasongo [10] explores IDS procedures catered for the wireless domain. The UNSW-NB15 data set was selected to derive both the training and test data sets. A wrapper-based feature extractor generated many feature vectors for comparison from a full set of 42 features. The experiment was performed for both binary and multi-class classification in which the type of attack was predicted. Candidate algorithms included decision Trees, Random Forest, Naïve Bayes, K-Nearest Neighbor, Support Vector Machines, and Feed-forward Artificial Neural Networks (FFANN). The optimal feature set consisted of 26 columns. The FFANN reflected the best classification accuracy on the full data set with 87.10% binary and 77.16% multi-class. Random Forest, Decision Tree, and Support Vector Machine were close behind. When the feature set and neural network hyper-parameters were optimized, the classification accuracy of the FFANN improved to 99.66% and 99.77% for binary and multi-class classification, respectively.

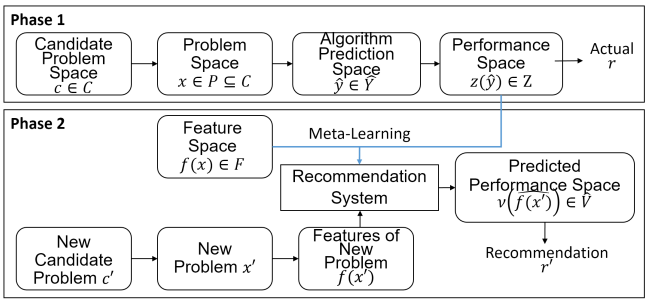

Figure 1: The meta-learner version of Rice's framework [29].

2.2 Algorithm Selection Problem

Rice's algorithm selection framework was presented in 1976 [17]. The framework is performed by employing all algorithms under consideration on all problems in a problem set. One or more performance metrics are chosen, and the performance of each algorithm on each problem is reported. Upon completion of the process, the preferred algorithm for each problem is taken as the one with the best performance metrics [17]. Woods [29] presents a modern depiction of Rice's framework as phase 1 in Figure 1.

The classic approach of learning algorithms is known as base learning. That is an ML algorithm which builds a data-driven model for a specific application [5]. Meta-learning, however, is an approach introduced by [23] which algorithms learn on the learning process itself. A meta-learning algorithm extracts meta-features $f(x) \in$ space F from a problem $x \in$ problem space P. The meta-model is trained to recommend the best-known base learning algorithm $a \in A$ to solve x. Works such as [15] and [3] further contribute to the theory of meta-learning recommendation systems [5].

In 2014, Smith [19] proposes the concept of applying meta-learning to Rice's model. It was not until 2016, however, that [5] implemented the concept. Figure 1 demonstrates that Cui et al. [5] trained a meta-learning model to correlate problem features to algorithm performance and that the trained model could be used to recommend the algorithm for unobserved problems within Rice's framework. The meta-learner correctly recommended the best algorithm in 91% of test problems. Further, it demonstrated that time to perform algorithm selection could be reduced from minutes to seconds compared to trial-and-error techniques [5]. Follow on studies by [29] and [26] expanded on this work by exploring various meta-features and meta-learner response metrics.

3 METHODOLOGY

The assigned task for an IDS is to classify network traffic records as normal or malicious. This task is investigated from the broader perch of the algorithm selection problem. Figure 2 shows that within

the analysis process, three factors drive the analytical approach and analytical technique selection. They are the input to the algorithm selection framework.

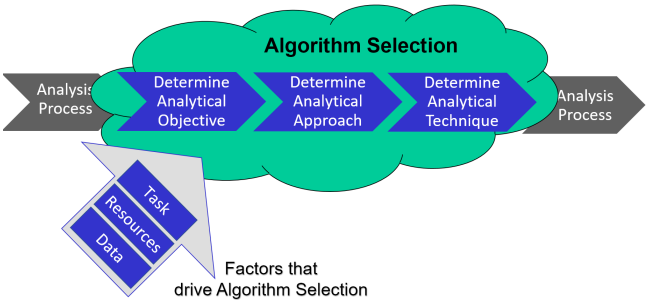

Figure 2: The factors identified are superimposed with the stages of algorithm selection which they impact.

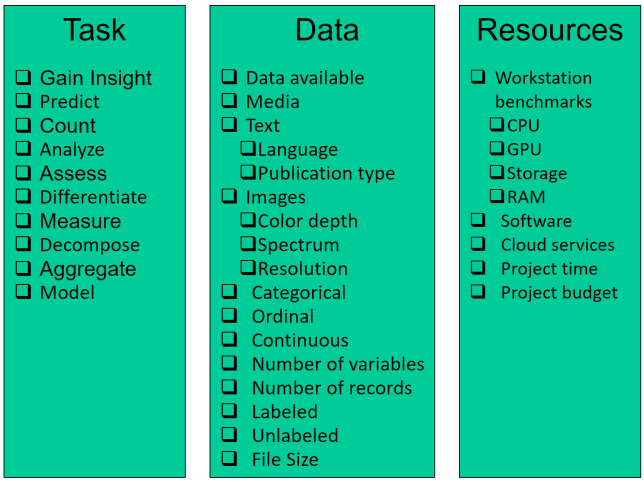

Figure 3: The considerations are shown for each factor which drives analytical approach and analytical technique selection.

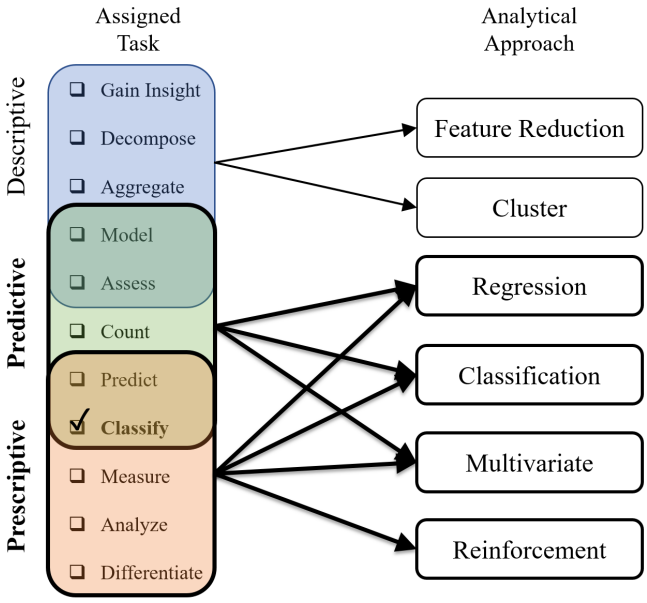

Figure 4: The assigned task for an IDS is classify. Classify is one of 11 common assigned tasks. It belongs to the predictive and prescriptive categories of analysis.

3.1 Characterizing the Problem

The framework is a mechanism to characterize an analysis problem and to determine the algorithms that best matches the problem characterization. The three factors each drive analytical approach selection and analytical technique selection. The factor *assigned task* pertains to the problem provided by a decision-maker. The analyst must decipher the intent of the assignment from the lexicon of the decision-maker into specific analytical terms, which are listed under the *Task*. This list of terms, called *considerations* is shown in Figure 3 for each factor. The considerations for the factor *data* describe the different formats analysts commonly receive data for analysis problems. The *data* factor is important because it relates to the problem's compatibility with the mathematical mechanics of the analysis technique. Likewise, the considerations for the *resources* factor help the analyst identify which algorithms are compatible with the available resources. The analyst should refer to Figure 3 to evaluate and record the considerations for each factor prior to beginning step 1.

3.2 Step 1: Map Problem to Category and Approach

Step 1 leverages information from the *problem characterization* to identify the appropriate *analytical approaches*. Each *consideration* selected from the *assigned task* factor maps to one or more *categories of analysis*. The *categories of analysis* describe the general goal of the analysis problem [4]. Each *category of analysis* can be implemented by certain *analytical approaches*. The *analytical approach* a technique class referring to the specific type of response the techniques produce. Therefore, the framework leverages a hierarchical taxonomy that groups techniques grouped by both categories of analysis and analytical approaches. Figure 4 shows the mapping from *assigned task* to *category of analysis*, and the mapping of *category of analysis* to *analytical approach*.

Since the task of an IDS is to *classify* network users, the *prescriptive* and *predictive* categories of analysis are selected.

An excerpt of the proposed taxonomy is presented in Figure 5. The taxonomy is built with an object-oriented structure to promote flexibility and expandability. As an example, techniques are shown within the *regression* and *classification analytical approaches*. The text *predictive* and *descriptive* appears at the bottom edge of the regression panel to indicate that regression techniques produce results suitable for either of these two *categories of analysis*. The requirements, or required considerations, for each factor are presented with the technique. Compatible training styles are listed to the right of the technique name. The object-oriented structure

allows new techniques to be easily added and new attributes to be included.

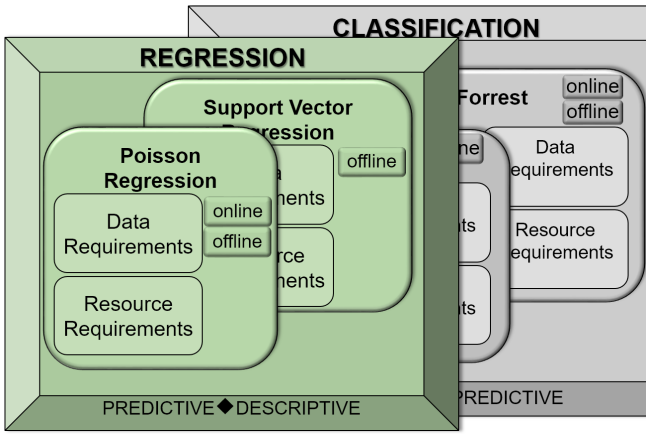

Figure 5: A portion of the proposed taxonomy is highlighted to show its structure.

3.3 Step 2: Rank Techniques

The framework identifies a subset of techniques that are compatible for the problem according to application. Next, the framework leverages the remaining three factors *data, resources* and *experience* to discern aspects of technique compatibility relating to the mechanics of the mathematical model. Step two predicts the utility scores of each algorithm from these factors according to two strategies: *rules of thumb* and *meta-learning*. They are presented in parallel below.

3.3.1 Rules-of-Thumb. A logical decision tree is used to assign a preference rank among candidate algorithms. The decision tree is built according to rules-of-thumb regarding features of the data. The features pertaining to data also impact the compatibility of techniques in respect to resources. Thus, it is justified to use the same decision tree, Figure 6, to adjudicate the scores for both factors.

3.3.2 Meta-Learning. A meta-learner is constructed in Python 3.7. All data sets are pre-processed according to best practices for data mining. Base learning is performed on 14 benchmark data sets with 20 repetitions. For each repetition, the data sets were split into training and test sets with an 80/20 ratio and with stratification. The KDDTest+ and KDDTrain+ sets were obtained having already been split into a master test set and a training set. 12 meta-features of each data set were stored as predictor data for the meta-learner; the mean observed recall was stored as the target. The meta-learner was trained to model recall as a function of meta-features using a support vector regression algorithm. The radial basis function kernel was selected. The regularization parameter was set at 1.0, and the kernel coefficient was auto-scaled as a function of the number of features and predictor variance. All other parameters followed Scikit-learn defaults [14]. Pseudo-code of the meta-learner is presented in Algorithm 1.

The candidate algorithms are selected because they are members analytical approaches derived in step 1 of Section 3.2. Clearly, these

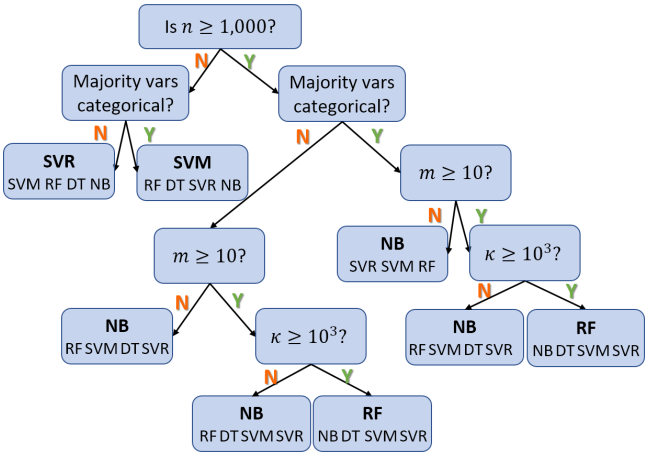

Figure 6: The decision tree represents the logical tests used to rank the recommendations via the rules of thumb strategy

.

Algorithm 1: Pseudo-code of Meta-learner

input : Repository of Datasets, Candidate Algorithms
output: Predicted Recall & Observed Recall of Test Dataset using Candidate Algorithms

```
Pre-processing
for all Dataset in Repository do
    for all column in Dataset do
        if column is numerical then
            miniMax(0,1)
            PCA()
            miniMax(0,1);
        else column is categorical;
            OneHot()
Feature Extraction
for all Dataset in Repository do
    featureExtract()
Base Learning
for all Algorithm do
    for all Dataset in Repository do
        trainBaseLearner()
        TestBaseLearner()
        RecordRecall()
        RankAlgorithms()
Meta Learning
for all Algorithm do
    TrainMetaLearner()
    PredictRecall()
    RecordRecall()
    RankAlgorithms()
```

are not the only algorithms that fall into the applicable analytical approach. Rather, they represent a demonstrative taxonomy. Note

that the Scikit-learn default settings are selected on each algorithm to show generality of the algorithm selection framework.

Training data sets are selected from easily accessible benchmark repositories. The first five are selected to provide diversity of meta-features to the meta-learner and improve the robustness of the model. The nine subsets of KDDTrain+ are selected to provide statistical information consistent with the KDDTest+ data set. KDDTrain+ is split into subset to promote diversity of meta-features but also reduce the number of records in the training set which is an order of magnitude greater than the number of records in the test set. A uniform random number generator is used to determine the number of rows allocated to each training set. Rows are not re-arranged during the subset process.

(1) Training Data
  (a) Heart: Predict presence of heart disease from 13 predictor variables [9]
  (b) Framingham: Predict presence of heart disease in the Framingham study from 15 predictor variables [1]
  (c) Spam: Predict if an email is spam based on six predictor variables [2]
  (d) Loan: Predict whether a consumer purchases a loan from Thera Bank based on 12 predictor variables [8]
  (e) Cancer: Predict whether a patient has breast cancer from 30 predictor variables collected in a fine needle aspirate procedure [27]
  (f) Nine subsets of KDDTrain+: Predict whether a network activity record is normal or malicious from four categorical and 39 numerical predictor variables [22]
(2) Test Data
  (a) KDDTest+: Predict whether a network activity record is normal or malicious from four categorical and 39 numerical predictor variables [22]

The choice of meta-features was adopted from [26]. The following meta-features were used as predictor data by the meta-learner to model expected recall.

- Number of Rows
- Number of Columns
- Rows to Columns Ratio
- Number of Discrete Columns
- Maximum number of factors among discrete columns
- Minimum number of factors among discrete columns
- Average number of factors among discrete columns
- Number of continuous columns
- Gradient average
- Gradient minimum
- Gradient maximum
- Gradient standard deviation

4 RESULTS

The algorithm selection framework is applied to the task of classifying network traffic as malicious or normal. Problem characterization is performed in step 1 to identify *prescriptive* and *predictive* as the categories of analysis. This leads to four analytical approaches, namely regression, classification, multivariate, and reinforcement. Five example algorithms which meet this criteria are taken from a notional taxonomy.

In step 2, candidate algorithms are ranked in order of preference by each recommendation strategy, rules-of-thumb and meta-learner. Both strategies yielded support vector regression as the most highly recommended algorithm. According to the mean observed recall, random forest was the best performing algorithm to detect malicious activity from the KDDTest+ data set. The standard deviations of observed recall on each algorithm were very low. The 90% Bonferroni confidence interval for support vector machine (SVM) and support vector regression (SVR) overlapped, indicating statistically identical recall performance. All other mean values were statistically unique. Further, the Spearman's coefficient of rank correlation was not statistically significant, largely due to the small size of the rank scheme. Since neither recommendation strategy succeeded in predicting the best performing algorithm, recall efficiency is introduced. Recall efficiency, Equation 1, is the ratio of the recall observed by the top recommended algorithm to best observed recall.

$$E_R = \frac{R_{bestRec}}{R_{bestObs}} \tag{1}$$

The recall efficiency for SVR, the top recommendation of both strategies, is 0.98. Table 1 presents a summary of the results including observed mean recall, meta-learner predicted recall, mean runtime, standard deviation of observed recall, observed ranks, rule-of-thumb predicted ranks, and meta-learner predicted ranks. Figure 7 outlines the mean observed recall and the recall predicted by the meta-learner for each algorithm.

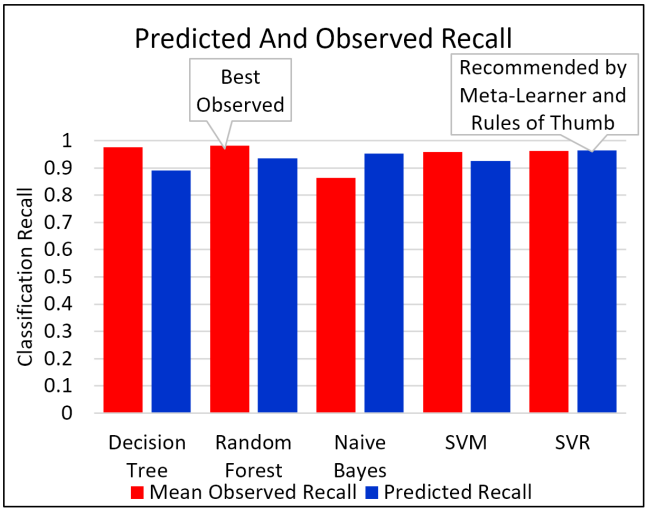

Figure 7: The predicted recall and observed mean recall are compared for each algorithm.

It is difficult to ascertain whether either of the recommendation strategies employed in this study were successful. While the recall efficiency is very high, the rank correlation was not conclusive. All base learners produced very high and very similar recalls. It would therefore be difficult for any model to discern the true rank preference. The meta-model employed the meta-features according to the precedent set by [26], however there were many more proposed by [5] that were not used. Furthermore, the meta-learner was trained

Table 1: Results compare the recommendations of each strategy to observed algorithm performance.

	Observed Mean Recall	Meta-Learner Predicted Recall	Mean Runtime (s)	SD of Observed Recall	Observed Ranks	Rules-of-Thumb Predicted Ranks	Meta-Learner Predicted Ranks
Decision Tree	0.975340865	0.891227551	0.188214	0.004141	2	4	5
Random Forest	0.982216595	0.935765698	1.593553	0.003063	**1**	3	3
Naive Bayes	0.863400857	0.952576618	0.007137	0.007961	5	2	2
SVM	0.959329957	0.925262066	8.602973	0.003163	4	5	4
SVR	0.962446436	0.96525973	7.647529	0.003088	3	**1**	**1**

by only 14 data sets, significantly less than used in [29]. Providing more training sets, especially from the domain of network traffic, would likely improve the predictive capability of the meta-learner.

As a whole, the framework is beneficial even when it does not recommend the true best performer. The framework consistently filters techniques that are incompatible with the problem characterization. Further, the framework identifies five viable options, each of which perform excellently.

5 CONCLUSION

Cyber attack detection from an IDS using any of the recommended algorithms could reasonably be deemed successful. Neither of the two recommendation strategies demonstrated perfect results. They did, however, show enough promise to motivate further investigations. Fundamentally, the meta-data and user input collected by the framework does contain information capable of consistently predicting a good analysis technique for a problem. Notably, there were algorithms from distinct analytical approaches that performed well on the same task. The process of problem characterization fits well into the framework but does require further refining. The rule-of-thumb decision tree provided intelligible recommendation logic whereas the meta-learner is a black box model. Future work should use the Gini criterion to optimize the decision tree. Further, the meta-learner should be improved to include more meta-features and training sets. There is a close connectedness in having a useful taxonomy of algorithms and a successful algorithm selection. This relationship is only beginning to be understood. Wireless IDS already provide good classification performance, however, algorithm selection, hyper-parameter tuning and feature engineering suffer from the time-costly trial-and-error practice. The algorithm selection framework may be a step forward in reducing this cost.

REFERENCES

[1] Aman Ajmera. 2019. *Framingham Data Set.* Retrieved May 15, 2020 from www.kaggle.com/dileep070/heart-disease-prediction-using-logistic-regression#framingham.csv

[2] Vincent Arel-Bundock. 2019. *Spam Data Set.* Retrieved May 15, 2020 from http://vincentarelbundock.github.io/Rdatasets/doc/DAAG/spam7.html

[3] Pavel Brazdil, João Gama, and Bob Henery. 1994. Characterizing the Applicability of Classification Algorithms Using Meta-Level Learning. In *European Conference on Machine Learning*. Springer, 83–102.

[4] James J Cochran. 2018. *INFORMS Analytics Body of Knowledge.* John Wiley & Sons.

[5] Can Cui, Mengqi Hu, Jeffrey D. Weir, and Teresa Wu. 2016. A Recommendation System for Meta-modeling: A Meta-learning Based Approach. *Expert Systems With Applications* 46 (2016), 33–44.

[6] Can Cui, Teresa Wu, Mengqi Hu, Jeffery D Weir, and Xianghua Chu. 2014. Accuracy vs. Robustness: Bi-Criteria Optimized Ensemble of Metamodels. In *Proceedings of the Winter Simulation Conference 2014*. IEEE, 616–627.

[7] Herve Debar, Marc Dacier, and Andreas Wespi. 1999. Towards a taxonomy of intrusion-detection systems. *Computer Networks* 31 (1999), 805–822.

[8] Sunil Jacob. 2018. *Personal Loan Data Set.* Retrieved May 15, 2020 from https://www.kaggle.com/itsmesunil/bank-loan-modelling/?

[9] Andras Janosi, William Steinbrunn, Matthias Pfisterer, and Robert Detrano. 2018. *Heart Data Set.* Retrieved May 15, 2020 from https://archive.ics.uci.edu/ml/datasets/Heart+Disease

[10] Sydney Mambwe Kasongo and Yanxia Sun. 2020. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Computers & Security* 92 (2020), 101752.

[11] Carl E Landwehr, Alan R Bull, John P McDermott, and William S Choi. 1994. A taxonomy of computer program security flaws. *ACM Computing Surveys (CSUR)* 26, 3 (1994), 211–254.

[12] Paul Maxwell, Elie Alhajjar, and Nathaniel D Bastian. 2019. Intelligent Feature Engineering for Cybersecurity. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 5005–5011.

[13] Swati Paliwal and Ravindra Gupta. 2012. Denial-of-service, probing & remote to user (R2L) attack detection using genetic algorithm. *International Journal of Computer Applications* 60, 19 (2012), 57–62.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[15] Larry A Rendell, Raj Sheshu, and David K Tcheng. 1987. Layered Concept-Learning and Dynamically Variable Bias Management. In *IJCAI*. 308–314.

[16] S Revathi and A Malathi. 2013. A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT)* 2, 12 (2013), 1848–1853.

[17] John Rice. 1976. The Algorithm Selection Problem. *Advances in Computers* 15 (1976), 75–152.

[18] Timothy W Simpson, Jesse Peplinski, Patrick N Koch, and Janet K Allen. 1997. On the Use of Statistics in Design and the Implications for Deterministic Computer Experiments. *Design Theory and Methodology* 14 (1997), 14–17.

[19] Michael R. Smith, Logan Mitchell, Christophe Giraud-Carrier, and Tony Martinez. 2014. Recommending Learning Algorithms and Their Associated Hyperparameters. In *CEUR Workshop Proceedings*. Aachen University, 39–40.

[20] Michael R. Smith, Logan Mitchell, Christophe G. Giraud-Carrier, and Tony R. Martinez. 2014. Recommending Learning Algorithms and Their Associated Hyperparameters. *CoRR* abs/1407.1890 (2014), 1–2. arXiv:1407.1890 http://arxiv.org/abs/1407.1890

[21] Michael Sobirey. 2000. *Michael Sobirey's Intrusion Detection Systems page.* Retrieved May 15, 2020 from https://isl.cse.sc.edu/mirrorSobireys.shtml

[22] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. 2009. *A Detailed Analysis of the KDD CUP 99 Data Set.* Retrieved May 15, 2020 from https://www.ee.ryerson.ca/~bagheri/papers/cisda.pdf

[23] Paul E Utgoff. 1986. Shift of Bias for Inductive Concept Learning. *Machine Learning: An Artificial Intelligence Approach* 2 (1986), 107–148.

[24] E. Viegas, A. O. Santin, A. França, R. Jasinski, V. A. Pedroni, and L. S. Oliveira. 2017. Towards an Energy-Efficient Anomaly-Based Intrusion Detection Engine for Embedded Systems. *IEEE Trans. Comput.* 66, 1 (2017), 163–177.

[25] G Gary Wang and Songqing Shan. 2007. Review of Meta-Modeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical design* 129, 4 (2007), 370–380.

[26] Clarence O. Williams. 2020. *Meta Learning Recommendation System for Classification.* Master's thesis. Air Force Institute of Technology.

[27] William H. Wolberg, W. Nick Street, and Olvi L. Mangasarian. 1995. *Breast Cancer Data Set.* Retrieved May 15, 2020 from https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[28] David H Wolpert. 2002. The Supervised Learning No-Free-Lunch Theorems. *Soft Computing and Industry* (2002), 25–42.

[29] Megan K. Woods. 2020. *A Metamodel Recommendation System Using Meta-Learning.* Master's thesis. Air Force Institute of Technology.