In-depth Comparative Evaluation of Supervised Machine Learning Approaches for Detection of Cybersecurity Threats

Laurens D'hooge, Tim Wauters, Bruno Volckaert and Filip De Turck
Ghent University - imec, IDLab, Department of Information Technology, Technologiepark-Zwijnaarde 126, Gent, Belgium

Keywords: Intrusion Detection, CICIDS2017, Supervised Machine Learning, Binary Classification.

Abstract: This paper describes the process and results of analyzing CICIDS2017, a modern, labeled data set for testing intrusion detection systems. The data set is divided into several days, each pertaining to different attack classes (Dos, DDoS, infiltration, botnet, etc.). A pipeline has been created that includes nine supervised learning algorithms. The goal was binary classification of benign versus attack traffic. Cross-validated parameter optimization, using a voting mechanism that includes five classification metrics, was employed to select optimal parameters. These results were interpreted to discover whether certain parameter choices were dominant for most (or all) of the attack classes. Ultimately, every algorithm was retested with optimal parameters to obtain the final classification scores. During the review of these results, execution time, both on consumer- and corporate-grade equipment, was taken into account as an additional requirement. The work detailed in this paper establishes a novel supervised machine learning performance baseline for CICIDS2017. Graphics of the results as well as the raw tables are publicly available at https://gitlab.ilabt.imec.be/lpdhooge/cicids2017-ml-graphics.

1 INTRODUCTION

Intrusion detection is a cornerstone of cybersecurity and an active field of research since the 1980s. Although the early research focused more on host intrusion detection systems (HIDS), the principal aims of an intrusion detection system (IDS) have not changed. A well-functioning IDS should be able to detect a wide range of intrusions, possibly in real-time, with high discriminating power, improving itself through self-learning, while being modifiable in its design and execution (Denning and Neumann, 1985). The advent of computer networking and its ever greater adoption, shifted part of the research away from HIDS to network intrusion detection systems (NIDS). This paper details the experiment and results of analyzing a modern intrusion detection dataset (CICIDS2017) and it is structured as follows. First an overview of the related work in intrusion detection and network security dataset generation is given, then the implementation of the analysis is described (section 3). Third and most important is the discussion of the results (section 4), summarized in the conclusion . Key findings in this work are the outstanding performance both in terms of classification and time metrics of tree-based classifiers, especially ensemble learners, the surprising effectiveness of simple distance-based methods and the classification difficulty for all methods on one attack class.

2 RELATED WORK

2.1 Intrusion Detection

The field of network intrusion detection developed two main visions on solving the problem of determining whether observed traffic is legitimate. The chronologically first approach, is the use of signature-based systems (also called misuse detection systems). Within this category different strategies have been researched (Axelsson, 2000), including state modelling, string matching, simple rule based systems and expert systems (emulating human expert knowledge, by making use of a knowledge base and an inference system). All systems in this category, while being great at detecting known signatures, do not generalize. That's a violation of the principles for intrusion detection systems, namely the system's ability to improve itself through learning.

The second approach, anomaly detection, has

D'hooge, L., Wauters, T., Volckaert, B. and De Turck, F.
In-depth Comparative Evaluation of Supervised Machine Learning Approaches for Detection of Cybersecurity Threats.
DOI: 10.5220/0007724801250136
In *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS 2019)*, pages 125-136
ISBN: 978-989-758-369-8

been around almost as long, but the methods have changed drastically in the past years. Early systems based their decisions on rules, profiles and heuristics, often derived from relatively simple statistic methods. These systems could be self-learning in the sense that their heuristics could be recomputed and thus become a dynamic part of the system. Advances in the last ten years in terms of distributed computation and storage have enabled more advanced statistical methods to become feasible. Work by Buczak et al. (Buczak and Guven, 2016) concluded that making global recommendations is impossible and the nature of the data and types of attacks to be classified should be taken into account when designing an IDS. Furthermore they stress the requirement for training data in the field of network intrusion detection and an evaluation approach that considers more than just accuracy. On a final note, the authors included recommendations for machine learning (ML) algorithms for anomaly detection (density based clustering methods and one-class SVMs) and for misuse detection (decision trees, association rule mining and Bayesian networks).

Very recent work by Hodo et al. (Hodo et al., 2017) examines the application of shallow and deep neural networks for intrusion detection. Their graphical overview of IDS techniques clearly shows the dominant position of anomaly based methods, driven by the adoption of machine learning techniques. Their main contribution is a chapter explaining the algorithm classes of neural networks. The work differentiates between artificial neural networks (ANN) (shallow) and deep networks (DN), with subdivisions between supervised and unsupervised methods for ANNs and generative versus discriminative methods for DNs. Their conclusion is that deep networks show a significant advantage for DNs in detection. They note that the adoption of either class is still in its early stages, when applied to network intrusion detection.

2.2 Datasets

Self-learning systems require data to train and test their efficacy. All techniques used in this work are supervised, machine learning algorithms. This means that they do not just require data, but that data has to be labeled. The dataset landscape in intrusion detection has a.o. been described by Wu et al. (Wu and Banzhaf, 2010) as part of a review paper on the state of computational intelligence in intrusion detection systems and more succinctly by Shiravi et al. (Shiravi et al., 2012), as prelude to their efforts in generating a new approach for dataset creation.

This work will only offer a very brief overview of the most studied datasets. KDDCUP99 (KDD99), the subject of ACM's yearly competition on Data mining and Knowledge Discovery in 1999 is by far the most studied data set for intrusion detection. Its origin is to be found in a DARPA funded project, run by the Lincoln Lab at MIT, which was tasked with evaluating the state of the art IDSs at the time. Apart from being based on twenty year old data by now, it has also been criticized by McHugh in 2000 (McHugh, 2000), by Brown et al. in 2009 (Brown et al., 2009) and by Tavallaee et al. also in 2009 (Tavallaee et al., 2009).

The persistence of a single dataset for almost two decades and its improved version, which now also is nearly a decade old, called for new research into dataset generation. The Canadian Institute for Cybersecurity (CIC), a coalition of academia, government and the public sector, based at the University of New Brunswick is the front runner in this field of research. The analysis by Tavallaee et al. of the dataset resulted in a new dataset, named NSL-KDD, in which structural deficiencies of KDD99 were addressed. NSL-KDD does not have redundant records in the training data, removed duplicates from testing data, reduced the total number of records so that the entire dataset could be used, instead of needing to sample it. Finally, to improve the variability in ability of the learners they tested, items which were hard to classify (a minority of the learners classified them properly), were added to NSL-KDD with a much higher frequency than items which most of the classifiers identified correctly. Because NSL-KDD is a derivation of KDD99, it is not completely free from its origin's issues.

2.2.1 ISCXIDS2012 & CICIDS2017

After publishing NSL-KKD, the CIC started a new project to create modern, realistic datasets in a scalable way. The first results from this project are documented in (Shiravi et al., 2012). Their system uses alfa and beta profiles. Alfa profiles are abstracted versions of multi-stage attacks, which would ideally be executed fully automatically, but human execution remains an option. Beta profiles are per protocol abstractions ranging from statistical distributions to custom user-simulating algorithms. Building on this foundation, published in 2012, a new dataset was published in 2017. The main difference is that CICIDS2017 (Sharafaldin et al., 2018) is geared more towards machine learning, with its 80 flow-based features, whereas, ISCXIDS2012 had 20 packet features. Both datasets give access to the raw pcap files, for further analysis. The flow features were gathered with CICFlowMeter, an open source flow generator and analyzer. CICIDS2017 added an HTTPS beta profile, which was necessary to keep up with the surge in HTTPS adoption on the web (Google transparency report). This rest of this work will cover an analysis of the newest dataset, CICIDS2017.

3 ARCHITECTURE AND IMPLEMENTATION

The evaluation of this dataset is a project in Python, supported by the Pandas (McKinney,), Numpy and Sklearn (Pedregosa et al., 2011) modules. The following subsections detail the engineering effort and choices to produce a robust, portable solution to evaluate any dataset. Google's guide (Zinkevich,), Rules of Machine Learning: Best Practices for ML Engineering, by Martin Zinkevich, has been influential on the implementation (mainly rules 2, 4, 24, 25, 32 and 40), as well as the detailed guides offered by Scikit-Learn. The current implementation makes use of nine supervised machine learning classifiers. Four tree-based algorithms: a single decision tree (CART) (dtree), a random forest ensemble learner (rforest), a bagging ensemble learner (bag) and an adaboost ensemble learner (ada). Two neighbor-based ones: the K-nearest neighbor classifier (knn) and the N-centroid classifier (ncentroid), two SVM-based methods: linearSVC (liblinear subsystem) (linsvc) and RBFSVC (libsvm subsystem) (rbfsvc) and one logistic regression (L-BFGS solver) (binlr).

3.1 Data Loading & Preprocessing

The dataset consists of labeled flows for eight days. A merged version has also been created. Details about the content are listed in table 1. Each day has the same features, 84 in total (label not included), though it should be noted that the "Fwd Header Length" feature is duplicated, an issue of CICFlowMeter that has been fixed in the source code, but persisted in the dataset. Another caveat when importing this data for analysis, is the presence of the literal value Infinity. String-type data like this results in run time crashes, when mixed with numeric data. This was rectified by replacing the strings with NaN values.

The Label column was binarized. While this does incur information loss, it is justified for an outer defense layer to classify first between benign and malign traffic. For each file the distribution over attack - normal traffic is summarized in table 2.

3.2 Cross-validation and Parameter Optimization

The implementation has two main branches, cross-validated parameter optimization and single execution testing. The branches share all code up to the point where the choice of algorithm is done. For parameter tuning, K-fold cross-validation is employed, with k = 5. The splits are stratified, taking samples proportional to their representation in the class distribution.

Parameter tuning is done with grid search, evaluating all combinations of a parameter grid. This is multiplicative: e.g. for two parameters, respectively with three and five values, fifteen combinations are tested. An overview of the algorithms and their pa-

Table 1: CICIDS2017 day, attack type, size mapping.

Dataset files		
Day	Attack type(s)	Size (MB)
Mon	No attacks	231
Tue	FTP / SSH bruteforce	173
Wed	Layer 7 DoS and Heartbleed	283
Thu AM	Web attacks	67
Thu PM	Infiltration	108
Fri AM	Ares botnet	75
Fri PM 1	Nmap port scanning	101
Fri PM 2	Layer 4 DDoS	95
Merged	All	1100

Table 2: CICIDS2017 day, benign samples, malign samples.

Dataset attack distribution		
Day	Benign	Malign
Mon	529918	0
Tue	432074	13835
Wed	440031	252672
Thu AM	168186	2180
Thu PM	288566	36
Fri AM	189067	1966
Fri PM 1	97718	128027
Fri PM 2	127537	158930
Merged	2273097	557646

Table 3: CICIDS2017 algorithm, parameters, ranges.

Parameter tuning search space		
Algorithm	Parameters	Search space
dtree	max_features max_depth	2..columns 1 1 .. 35 1
rforest	max_features max_depth	2..columns 5 1 .. 35 5
bag	max_features max_samples	0.1 .. 1.0 .1 0.1 .. 1.0 .1
ada	n_estimators learning_rate	5 .. 50 5 0.1 .. 1.0 .1
knn	n_neighbors distance_metric	1 .. 5 1 manhattan euclid
linsvc	max_iterations tolerance	$10e3. * 10e6$ 10 $10e-3. * 10e-5$ .1
binlr	max_iterations tolerance	$10e3. * 10e6$ 10 $10e-3. * 10e-5$ .1

rameter search spaces can be seen in table 3.

Special care went into avoiding model contamination. The data given for cross-validated parameter tuning is two thirds of that day's data. Optimal parameters are derived from only that data. The results from cross-validation are stored. These results include: the total search time, the optimal parameters, the parameter search space and the means of five metrics (balanced accuracy, precision, recall, f1-score and ROC-AUC).

3.3 Metric Evaluation and Model Selection

The only point of interaction between the cross-validation (cv) code and the single execution is in gathering the optimal model parameters from the result files, written to disk by the cv code. The optimal parameters are chosen, based on a voting system. That voting system looks at the ranks each set of tested parameters gets on the following five metrics:
Balanced Accuracy: combined per class accuracy, useful for skewed class distributions. Obtained through evaluating equation 1 for each class, averaged, compared to accuracy itself which also uses equation 1 over all classes simultaneously. TP, TN, FP, FN respectively stand for true / false positive / negative.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

Precision: Of the items that were tagged as positive, how many are actually positive (equation 2).

$$PR = \frac{TP}{TP+FP} \tag{2}$$

Recall: Of the items that were tagged as positive, how many did we tag compared to all positive items (equation 3).

$$RC = \frac{TP}{TP+FN} \tag{3}$$

F1-score: defined as the harmonic mean of precision and recall (equation 4), the F1-score combines these metrics in such a way that the impact of poor scores on either of the metrics, heavily impacts the final score. In order to achieve a high F1-score, it is not only sufficient to be precise in prediction (discriminative power), but equally high in finding a generalized representation of positive samples.

$$F1 = \frac{2 * precision * recall}{precision + recall} \tag{4}$$

ROC-AUC: the receiver operator characteristic (ROC) is a visual metric of the relationship between the true positive rate (recall) on the y-axis and the false positive rate (equation 5) on the x-axis at different classifying thresholds. The thresholds are implicit in the curve. In essence it shows how well a classifier is able to separate the classes. To avoid having to interpret the plot, the area under the curve (AUC) is calculated. An AUC of 1 would mean that the classifier is able to completely separate the classes from each other. An AUC of 0.5 indicates that the class distributions overlay each other fully, meaning that the classifier isn't better than random guessing. The AUC reduces the ROC curve to a single number. If special care has to be given to the avoidance of false positives or to maximal true positive rate, then the AUC metric is no longer helpful. For unbalanced data sets, the ROC curve is a great tool, because the imbalance is irrelevant to the outcome.

$$FPR = \frac{FP}{FP+TN} \tag{5}$$

The optimal parameters are decided by a voting mechanism that works as follows: 1: Find the highest ranked set of parameters for each of the five metrics. 2: Aggregate across the found sets. 3: Pick the most prevalent set. Some algorithms showed a high preference for certain parameter values. The results are summarized in table 4.

Table 4: CICIDS2017 algorithm, parameters, results.

Parameter tuning search results		
Algorithm	Parameters	Search results
dtree	max_features max_depth	no clear winners no clear winners
rforest	max_features max_depth	no clear winners no clear winners
bag	max_features max_samples	0.7 / 0.8 (18/21) 0.9 / 1.0 (19/21)
ada	n_estimators learning_rate	no clear winners 0.6 (7/21)
knn	n_neighbors distance_metric	1 (20/21) manhattan (17/21)
linsvc	max_iterations tolerance	1000 (16/21) $10e-5$ (21/21)
binlr	max_iterations tolerance	1000 (13/21) $10e-3$ (21/21)

3.4 Algorithm Retesting with Optimal Parameters

For each day (each attack scenario), the algorithms were retested with optimal parameters. Execution of the fixed parameter functions, yields a dictionary with the used parameters, the run time, the predicted labels on the test set and the actual labels for the test set. Seven metrics are gathered, namely the five metrics used to evaluate in the cross-validation phase (described in paragraph 3.3). In addition the accuracy score is kept as well as the confusion matrix.

4 EVALUATION RESULTS

This section describes the results from the retesting with optimized parameters. In total 9 algorithms were tested. It should be noted that for two of these no cross-validation was done. The N-centroid classifier does not use optimized parameters, due to a limitation of Scikit-learn. For the RBF-SVC classifier, parameter optimization was skipped due to the excessive run times of forced single-core execution. The results are described in their respective algorithmic classes in subsection 4.1. All testing was done on two types of infrastructure, roughly reflecting corporate and private environments. The corporate grade server was equipped with 2X Intel Xeon E5-2650v2 @ 2.6GHz (16 cores) and 48GB of RAM, while the consumer-grade host had 1X Intel Core i5-4690 @ 3.5GHz (4 cores) and 16 GB of RAM.

4.1 Algorithm Comparison

This section details the results of testing the various classification algorithms. The algorithms are grouped, based on their underlying classifier. Due to page constraints only results of single execution on the consumer-grade hardware are contained in the paper in tabular format in appendix 5.1. It is advised to use this paper with the full collection of tables and derived graphs that are publicly available at https://gitlab.ilabt.imec.be/lpdhooge/cicids2017-ml-graphics. A sample result graphic is shown in figure 1.

4.1.1 Tree-based Classifiers

On the whole, the tree-based classifiers obtained the best results for all attack types, on all metrics. Even a single decision tree is able to achieve 99+% on all metrics for the DoS / DDoS and Botnet attack types. Another interesting finding is that building the tree without scaling the features, improves the performance on all metrics for detection of the brute force and port scanning traffic, to be near-perfect. Results on the merged dataset reveal that identification across different attack classes works with equally great results to the best-identified classes. It should however be noted that good performance on the merged data set includes the attack classes with the most samples

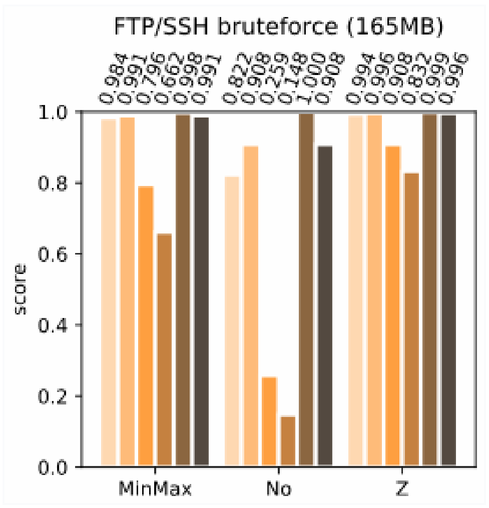

Figure 1: Sample result, full results available at https://gitlab.ilabt.imec.be/lpdhooge/cicids2017-ml-graphics.

(DoS, port scan & DDoS) and might obfuscate worse performance on the less prevalent attack classes.

When introducing meta-estimators, techniques at a higher level of abstraction that introduce concepts to improve the underlying classifier(s), several benefits were discovered. Random forests improved the results on port scanning traffic when applying either of the scaling methods, compared to a decision tree. For the other attack types results are very similar, with a noted reduction on botnet traffic classification, but only when using MinMax scaling. A reduction on multiple classification metrics is observed for the infiltration attacks as well. This attack type consistently is the hardest to classify, not least because the dataset only contains 36 of these flows, compared to the 288566 benign samples in the same set (2).A modest improvement on all metrics is observed for the merged data set compared to a single decision tree.

The bagging classifier proved itself to be a more potent meta-estimator, reaching near-perfect scores on all metrics for the brute force, Dos, DDoS, botnet and port scanning traffic. The improved performance and stability in the brute force and botnet classes compared to plain decision trees and random forests is its main advantage. Scoring on the infiltration attacks did take a big hit. This is believed to be a result of the additional sampling employed by the bagging classifier, when the number of samples to learn from is already very low. The almost perfect classification on five of the seven attack classes generalized to the evaluation of the entire data set.

The overall best meta-estimator was discovered as adaboost. Adaboost retains the near perfect scores, achieved by the bagging classifier on the previously mentioned classes. Thanks to its focus on improving classification for difficult samples, it was the best classifier for infiltration attacks, reaching balanced accuracies between 75% and 79.2%. Unfortunately the recall was never higher than 58.3%, meaning that about half of the infiltration flows were misclassified. Similarly to the bagging classifier, performance was stable and equally high on the merged data set. In total, 5 of the seven attack classes could be discovered with very high reliability, irrespective of the employed feature scaling method. On the web attacks (brute force, XSS and SQLi), the random forest and bagging classifiers had a slight, but stable edge compared to single decision trees and adaboost. The only class on which classification underperformed on all metrics was infiltration. Results pertaining to execution times are described in subsection 4.2.

4.1.2 SVM-based Classifiers and Logistic Regression

This subsection covers three more algorithms, two support vector machines: one with a linear kernel and one with a radial basis function kernel and a logistic regression classifier. Full results are listed in table 7.

The linear support vector machine consistently has very high scores on all metrics for the DoS, DDoS and port scan attack classes. Recall on the ftp/ssh brute force, web attacks and botnet attack classes is equally high, but gets offset by lower precision scores especially in the absence of feature scaling. The algorithm thus succeeds in recognizing most of the attacks, but has higher false-positive rates compared to the tree-based methods. Precision on the infiltration attack classes is extremely poor. The logistic regression with binomial output results, tells a similar story. Like the linear support vector classifier (linSVC), it performs best and most stably on the DoS, DDoS and port scanning traffic. Furthermore, recall scores on the brute force, web and botnet traffic classes are high to very high, but paired with worse precision scores compared to the linSVC, the applicability of this model gets reduced. One consistent result is that feature scaling is a necessity for the logistic regression classifier, preferring standardization over minmax scaling. The generalization or lack thereof is clearly visible in the results on the merged data set. Performance is reasonable with standardized features (98.4% recall & 80.4% precision), but not even close to the performance of the tree-based classifiers. This performance reduction does indicate that misclassification on the less prevalent classes is impactful on the final scores. The last algorithm in this category, a support vector classifier with radial basis function in the kernel demonstrates the importance of proper method selection for feature scaling. This classifier has dismal performance when features are used as input without scaling, not reaching acceptable performance on any of the seven attack classes. Minmax scaling gives great results on the DoS, port scan and DDoS attacks (the classes with the most samples to learn from 2). Standardizing (Z) scaling improves performance on these classes even further and succeeds much better at recognizing brute force, web and botnet traffic, making the classifier a valid contender for use on five of the seven attack classes.

4.1.3 Neighbor-based Classifiers

Despite its simplicity, the k-nearest neighbors algorithm, generally looking at only one neighbor and using the Manhattan (block) distance metric, is a high-performer in 6/7 attack scenarios. In four scenarios, 99.9% on all metrics is almost invariably obtained, with metrics for the other attack classes never below 95.7%. The elusive class to recognize remains infiltration attack traffic. Interestingly enough, even though this too is a distance-based algorithm, all feature scaling methods, yielded very similar results. Another conclusion is the loss of perfect classification, compared to the tree-based classifiers. While the reduction in classification performance is minute, it is observable and stable. Performance on the merged data set shows generalization capability, but this comes at a cost further described in subsection 4.2. The last algorithm, nearest centroid classifier is equally simple and has some interesting properties. Its results resemble the SVM results, with high, stable recognition of DDoS and port scanning traffic, mediocre but stable results in the DoS category and medium to high recall on the ftp/ssh brute force, web attack and botnet traffic, but again paired with low precision scores. Combined with its run time profile, it has application potential for the recognition of DDoS and port scanning traffic. The perfect scores on recall, regardless of scaling method for the FTP / SSH brute force and web attacks are also in interesting property. On the merged data set results show degraded performance, reflective of the poor classification scores on the other attack classes. Detailed results are available in the appendix in table 6.

4.2 Time Performance Comparison

Real-world intrusion detection systems have constraints, when evaluating traffic. Some example constraints are a.o.: required throughput, minimization of false positives, maximization of true positives, evaluation in under x units of time, real-time detection. To gain insight in the run time requirements of the different algorithms, summarizing charts can be seen in figures 2 and 3. Five main takeaways should be noted from these charts. First, as long as the dataset stays under 200 megabytes, evaluation can be done in under 20 seconds by most algorithms. To give an idea of the amount of flows in 200 MB, the Tuesday data set, 173 MB in size contains 445909 flows. Second, the algorithms split in two categories when looking at execution times: all tree and neighbor methods keep their execution time under one minute on both types of infrastructure when evaluating data sets under 300 MB. The SVM and regression models take more time to run, with outliers that are caused by not scaling the features. Applying a standardizing scaler to the data, drastically reduces the execution time. Third, the consumer-grade infrastructure holds its own against the corporate server. The reason for this is twofold. Purely on a hardware level, the lower core count, but substantially higher clock speeds (and thermal headroom for aggressive frequency scaling) keep the systems in competition. Another factor is how well an algorithm is suited for parallel execution. For example knn makes full use of all available cores, because the problem is easily separable, while the implementation of the rbfsvc in LIBSVM locks execution to a single core. Fourth, the nearest centroid classifier is as good as insensitive to data set size. In combination with the classifier's ability in recognizing DDoS and port scanning traffic, it is conceivable to employ it in real-time on IoT networks to identify compromised devices, taken over to execute DDoS attacks (blog,). Fifth and finally, execution times on the merged data set influence algorithm choice. Execution time on the consumer-grade hardware stays under ten minutes for the full, 1.1 GB data set for the ada, bag, dtree, ncentroid and rforest classifiers (around and under five minutes when ignoring the bagging classifier). Knn is very resource intensive and doesn't scale well into larger data sets. Evaluation on the merged data set took almost 1 hour and 30 minutes. The logistic regression completed evaluation of the full dataset in just over 30 minutes. In general using the corporate infrastructure with higher core counts reduces the execution time, but as mentioned earlier, this potential speedup depends on the implementation.

4.3 Result Comparison to State of the Art

Because of the recency of CICIDS2017, published research is still limited. Nonetheless a comparison to a selection of relevant research is already possible.

Attak et al. (Attak et al.,) focus on the DARE (data analytics and remediation engine) component of the SHIELD platform, a cybersecurity solution for use in software defined networks (SDN) with network function virtualization (NFV). The machine learning methods that were tested are segmented into two classes: those for anomaly detection and those for threat classification. Comparison to this work is apt for the threat classification portion. The researchers kept only a 10-feature subset of the flow data. This subset was chosen, not produced by a feature selection technique. The threat classification made use of the random forest and multi-layer perceptron classifiers. Optimal models from a 10-fold cross-validation were used on 20% of the data that was held out for validation. The random forest classifier obtained the best results on accuracy, precision and recall, often reaching perfect classification. The multi-layer perceptron had similar accuracy scores, but much greater variability on precision and recall. Overall their research shows favourable results for the random forest, both in terms of performance on the tested classification metrics, but also on execution time.

Marir et al. (Marir et al., 2018) propose a system for intrusion detection with deep learning for feature extraction followed by an ensemble of SVMs for classification, built on top of Spark, the distributed in-memory computation engine. The feature extraction is done by a deep belief network (DBN), a stack of restricted Boltzmann machines. Next, the dimension-reduced sample set is fed to a layer of linear SVMs for classification. Layers further in the stack of SVMs are given samples for which previous layers weren't confident enough. Because both the DBN and the SVMs operate in distributed fashion, the master node decides whether enough data is present to build a new layer. If not, then the ensemble of SVMs are the models produced in the final layer. This approach was tested on four datasets, namely KDD99, NSL-KDD, UNSW-NB15 and CICIDS2017. The results of testing their approach reveal that the combination of deep learning for feature extraction and supervised learning with an emphasis on retraining for difficult samples, yields better performance on the classification metrics, but requires more training time than the application of the individual parts on the classification task.

5 CONCLUSIONS AND FUTURE WORK

This paper contains a detailed analysis of CICIDS2017, a modern data set geared towards the application of machine learning to network intrusion detection systems. The design and implementation have

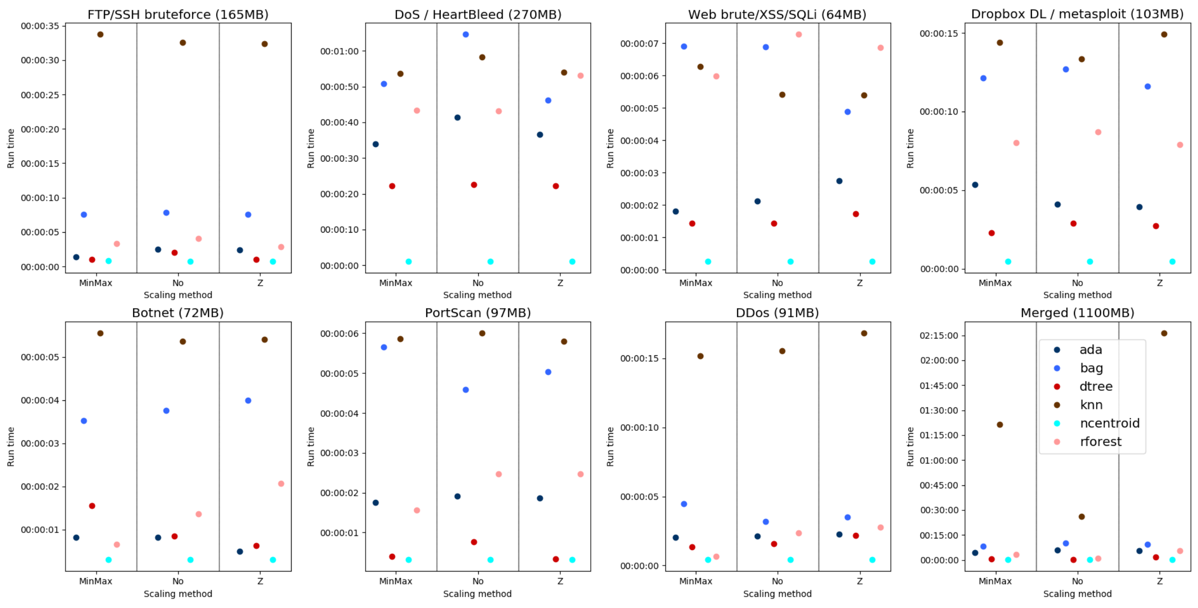

Figure 2: Algorithm run times i5 scatter part 1.

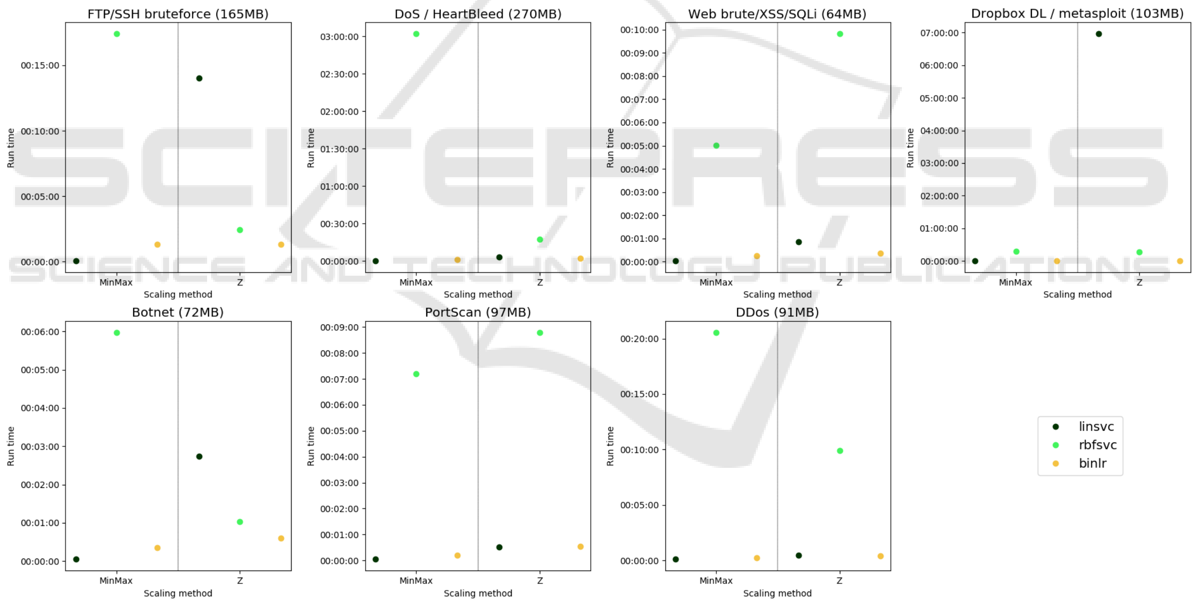

Figure 3: Algorithm run times i5 scatter part 2.

been laid out in section 3, focusing on the principles and application of solid machine learning engineering. The main section of this paper conveys the results of applying nine supervised learning algorithms with optimized parameters to the data. Results were gathered for every individual day, containing traffic from a specific attack class, as well as for a merged version, containing all attack types. The analysis was run on both consumer- and professional-grade hardware, to confirm stability of the results further and to investigate differences in execution time. In general, it can be stated that the tree-based classifiers performed best. Single decision trees are capable of recognizing DoS, DDoS and botnet traffic. Meta-estimators based on decision trees improved performance to the point where they are practically applicable for six of the seven attack classes. This was most true for the bagging- and adaboost classifiers. Another improvement of meta-estimators over single decision trees is their ability to abstract over the choice of feature scaling. In addition performance generalized to the merged data set, without incurring heavy increases in execution time. The elusive attack class to classify was infiltration. One reason for this might be the severe lack of positive training samples for this category, a natural consequence of the low network footprint of infiltration attacks. Only k-nearest neighbors (knn) came close to a decent precision score for this class, but it lacked in recall. Despite its simplicity knn is a potent classifier for five of the seven attack classes. It is held back by its steep increase in execution time for larger data sets, even though it generalizes as well as the tree-base meta-estimators. Opposite to this is the nearest centroid classifier, being nigh insensitive to dataset size, while applicable for the classification of port scan and DDoS traffic and for perfect detection of brute force and DoS traffic. The final algorithms, two support vector machines with different kernels and logistic regression are useful for recognition of port scan, DoS and DDoS traffic, provided the features are scaled (preferably normalized). However, these classifiers are not favoured when pitted against the tree-based classifiers, because the attack classes on which they perform well are only a subset of the classes for which the tree-based perform equally well.

REFERENCES

Attak, H., Combalia, M., Gardikis, G., Gastón, B., Jacquin, L., Litke, A., Papadakis, N., Papadopoulos, D., and Pastor, A. Application of distributed computing and machine learning technologies to cybersecurity. *Space*, 2:I2CAT.

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy.

blog, C. Inside the infamous mirai iot botnet: A retrospective analysis.

Brown, C., Cowperthwaite, A., Hijazi, A., and Somayaji, A. (2009). Analysis of the 1999 darpa/lincoln laboratory ids evaluation data with netadhict. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–7. IEEE.

Buczak, A. L. and Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2):1153–1176.

Denning, D. and Neumann, P. G. (1985). *Requirements and model for IDES-a real-time intrusion-detection expert system*. SRI International.

Hodo, E., Bellekens, X., Hamilton, A., Tachtatzis, C., and Atkinson, R. (2017). Shallow and deep networks intrusion detection system: A taxonomy and survey. *arXiv preprint arXiv:1701.02145*.

Marir, N., Wang, H., Feng, G., Li, B., and Jia, M. (2018). Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark. *IEEE Access*, 6:59657–59671.

McHugh, J. (2000). The 1998 lincoln laboratory ids evaluation. In Debar, H., Mé, L., and Wu, S. F., editors, *Recent Advances in Intrusion Detection*, pages 145–161, Berlin, Heidelberg. Springer Berlin Heidelberg.

McKinney, W. pandas: a foundational python library for data analysis and statistics.

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

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSP*, pages 108–116.

Shiravi, A., Shiravi, H., Tavallaee, M., and Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers & security*, 31(3):357–374.

Tavallaee, M., Bagheri, E., Lu, W., and Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set. In *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, pages 1–6. IEEE.

Wu, S. X. and Banzhaf, W. (2010). The use of computational intelligence in intrusion detection systems: A review. *Applied soft computing*, 10(1):1–35.

Zinkevich, M. Rules of machine learning: Best practices for ml engineering.

APPENDIX

5.1 Intel Core i5-4960 Full Results

Tables 5, 6 and 7 contain all results. Similar tables with detailed results for the execution times are available online at https://gitlab.ilabt.imec.be/lpdhooge/cicids2017-ml-graphics. Mirror tables and accompanying graphics of testing on the Intel Xeon E5-2650v2 are also available via the aforementioned link.

Table 5: Intel Core i5-4690 dtree, rforest and bag full results.

algorithm	day	scaling	accuracy	balanced accuracy	F1	precision	recall	ROC-AUC
dtree		MinMax	0.9922	0.9959	0.8893	0.8007	0.9998	0.9959
	1: FTP / SSH bruteforce	**No**	0.9999	0.9996	0.9991	0.9989	0.9993	0.9996
		Z	0.9920	0.9959	0.8862	0.7957	1.0000	0.9959
		MinMax	0.9998	0.9998	0.9998	0.9997	0.9998	0.9998
	2: DoS / Heartbleed	No	0.9998	0.9998	0.9997	0.9997	0.9998	0.9998
		Z	0.9998	0.9998	0.9997	0.9997	0.9997	0.9998
		MinMax	0.9998	0.9922	0.9907	0.9971	0.9844	0.9922
	3: Web attacks	No	0.9995	0.9936	0.9815	0.9755	0.9876	0.9936
		Z	0.9995	0.9937	0.9817	0.9758	0.9878	0.9937
		MinMax	0.9999	0.8182	0.7000	0.7778	0.6364	0.8182
	4: Infiltration	No	0.9999	0.7500	0.5455	0.6000	0.5000	0.7500
		Z	0.9999	0.7500	0.6316	0.8571	0.5000	0.7500
		MinMax	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	5: Botnet	No	0.9999	0.9984	0.9940	0.9910	0.9970	0.9984
		Z	0.9999	0.9985	0.9957	0.9943	0.9971	0.9985
		MinMax	0.8717	0.8521	0.8983	0.8154	1.0000	0.8521
	6: Portscan	**No**	0.9998	0.9998	0.9998	0.9999	0.9998	0.9998
		Z	0.8716	0.8516	0.8983	0.8154	1.0000	0.8516
		MinMax	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
	7: DDoS	No	0.9998	0.9998	0.9998	0.9998	0.9998	0.9998
		Z	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
		MinMax	0.9998	0.9997	0.9995	0.9995	0.9996	0.9997
	8: Merged (all types)	No	0.9973	0.9946	0.9931	0.9961	0.9901	0.9946
		Z	0.9998	0.9998	0.9996	0.9996	0.9996	0.9998
rforest		MinMax	0.9919	0.9958	0.8855	0.7946	1.0000	0.9958
	1: FTP / SSH bruteforce	No	1.0000	0.9997	0.9997	1.0000	0.9993	0.9997
		Z	0.9922	0.9960	0.8866	0.7963	1.0000	0.9960
		MinMax	0.9999	0.9999	0.9999	0.9999	0.9998	0.9999
	2: DoS / Heartbleed	No	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
		Z	0.9999	0.9999	0.9998	0.9999	0.9998	0.9999
		MinMax	0.9997	0.9898	0.9870	0.9945	0.9796	0.9898
	3: Web attacks	No	0.9997	0.9927	0.9901	0.9947	0.9855	0.9927
		Z	0.9998	0.9944	0.9923	0.9958	0.9889	0.9944
		MinMax	0.9999	0.8437	0.7586	0.8462	0.6875	0.8437
	4: Infiltration	No	0.9999	0.7000	0.5714	1.0000	0.4000	0.7000
		Z	0.9999	0.6875	0.4800	0.6667	0.3750	0.6875
		MinMax	0.9961	0.8102	0.7657	1.0000	0.6204	0.8102
	5: Botnet	No	1.0000	0.9992	0.9976	0.9968	0.9984	0.9992
		Z	0.9999	0.9976	0.9961	0.9969	0.9953	0.9976
		MinMax	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	6: Portscan	No	0.9999	0.9999	0.9999	1.0000	0.9998	0.9999
		Z	0.9999	0.9999	0.9999	1.0000	0.9998	0.9999
		MinMax	0.9999	0.9999	0.9999	1.0000	0.9998	0.9999
	7: DDoS	No	0.9999	1.0000	1.0000	1.0000	0.9999	1.0000
		Z	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
		MinMax	0.9999	0.9998	0.9998	0.9999	0.9997	0.9998
	8: Merged (all types)	No	0.9996	0.9992	0.9991	0.9997	0.9984	0.9992
		Z	0.9999	0.9998	0.9998	0.9999	0.9997	0.9998
bag		MinMax	1.0000	0.9999	0.9997	0.9996	0.9998	0.9999
	1: FTP / SSH bruteforce	No	1.0000	0.9998	0.9996	0.9996	0.9996	0.9998
		Z	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
		MinMax	0.9999	0.9999	0.9999	0.9999	0.9998	0.9999
	2: DoS / Heartbleed	No	0.9998	0.9998	0.9997	0.9999	0.9996	0.9998
		Z	0.9997	0.9997	0.9997	0.9998	0.9995	0.9997
		MinMax	0.9997	0.9939	0.9899	0.9919	0.9879	0.9939
	3: Web attacks	**No**	0.9998	0.9948	0.9905	0.9912	0.9897	0.9948
		Z	0.9997	0.9890	0.9889	1.0000	0.9780	0.9890
		MinMax	0.9999	0.7333	0.6364	1.0000	0.4667	0.7333
	4: Infiltration	No	0.9999	0.7500	0.6667	1.0000	0.5000	0.7500
		Z	0.9999	0.5833	0.2667	0.6667	0.1667	0.5833
		MinMax	0.9999	1.0000	0.9969	0.9938	1.0000	1.0000
	5: Botnet	No	1.0000	0.9977	0.9977	1.0000	0.9954	0.9977
		Z	0.9999	0.9970	0.9955	0.9970	0.9940	0.9970
		MinMax	0.9999	0.9999	0.9999	1.0000	0.9999	0.9999
	6: Portscan	No	0.9999	0.9999	0.9999	0.9999	0.9998	0.9999
		Z	0.9999	0.9999	0.9999	1.0000	0.9998	0.9999
		MinMax	0.9999	0.9999	0.9999	1.0000	0.9999	0.9999
	7: DDoS	No	0.9999	0.9999	1.0000	1.0000	0.9999	0.9999
		Z	0.9999	0.9999	0.9999	1.0000	0.9999	0.9999
		MinMax	0.9999	0.9998	0.9998	0.9998	0.9997	0.9998
	8: Merged (all types)	No	0.9999	0.9998	0.9997	0.9998	0.9997	0.9998
		Z	0.9999	0.9999	0.9998	0.9999	0.9998	0.9999

Table 6: Intel Core i5-4690 ada, knn, ncentroid, full results.

algorithm	day	scaling	accuracy	balanced accuracy	F1	precision	recall	ROC-AUC
		MinMax	1.0000	0.9997	0.9993	0.9993	0.9993	0.9997
	1: FTP / SSH bruteforce	No	1.0000	0.9999	0.9995	0.9991	0.9998	0.9999
		Z	0.9999	0.9997	0.9988	0.9980	0.9996	0.9997
		MinMax	0.9998	0.9999	0.9998	0.9997	0.9999	0.9999
	2: DoS / Heartbleed	No	0.9998	0.9998	0.9998	0.9997	0.9998	0.9998
		Z	0.9998	0.9998	0.9997	0.9997	0.9998	0.9998
		MinMax	0.9995	0.9901	0.9798	0.9792	0.9805	0.9901
	3: Web attacks	No	0.9996	0.9936	0.9841	0.9807	0.9875	0.9936
		Z	0.9997	0.9957	0.9896	0.9875	0.9917	0.9957
		MinMax	0.9999	0.7917	0.6364	0.7000	0.5833	0.7917
	4: Infiltration	No	0.9999	0.7500	0.6316	0.8571	0.5000	0.7500
ada		Z	0.9999	0.7916	0.6087	0.6364	0.5833	0.7916
		MinMax	1.0000	0.9992	0.9977	0.9969	0.9985	0.9992
	5: Botnet	No	1.0000	1.0000	0.9992	0.9985	1.0000	1.0000
		Z	0.9999	0.9976	0.9931	0.9908	0.9954	0.9976
		MinMax	0.9998	0.9998	0.9998	0.9998	0.9999	0.9998
	6: Portscan	No	0.9998	0.9998	0.9998	0.9998	0.9999	0.9998
		Z	0.9999	0.9999	0.9999	0.9999	0.9998	0.9999
		MinMax	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
	7: DDoS	No	0.9999	0.9999	0.9999	0.9998	0.9999	0.9999
		Z	0.9999	0.9999	0.9999	0.9999	1.0000	0.9999
		MinMax	0.9998	0.9997	0.9996	0.9995	0.9996	0.9997
	8: Merged (all types)	No	0.9999	0.9998	0.9996	0.9996	0.9997	0.9998
		Z	0.9998	0.9998	0.9996	0.9995	0.9997	0.9998
		MinMax	0.9996	0.9976	0.9944	0.9933	0.9954	0.9976
	1: FTP / SSH bruteforce	No	0.9997	0.9981	0.9948	0.9932	0.9965	0.9981
		Z	0.9996	0.9984	0.9939	0.9907	0.9972	0.9984
		MinMax	0.9969	0.9970	0.9957	0.9939	0.9975	0.9970
	2: DoS / Heartbleed	No	0.9966	0.9968	0.9953	0.9934	0.9973	0.9968
		Z	0.9967	0.9969	0.9955	0.9935	0.9976	0.9969
		MinMax	0.9994	0.9882	0.9773	0.9780	0.9766	0.9882
	3: Web attacks	No	0.9995	0.9903	0.9802	0.9795	0.9809	0.9903
		Z	0.9996	0.9936	0.9834	0.9794	0.9875	0.9936
		MinMax	0.9999	0.7692	0.6667	**0.8750**	0.5385	0.7692
	4: Infiltration	No	0.9999	0.6666	0.4000	0.5000	0.3333	0.6666
knn		Z	0.9999	0.7000	0.5455	0.8571	0.4000	0.7000
		MinMax	0.9994	0.9794	0.9723	0.9859	0.9590	0.9794
	5: Botnet	No	0.9994	0.9839	0.9744	0.9808	0.9680	0.9839
		Z	0.9994	0.9786	0.9738	0.9909	0.9574	0.9786
		MinMax	0.9990	0.9989	0.9991	0.9990	0.9992	0.9989
	6: Portscan	No	0.9987	0.9987	0.9989	0.9986	0.9992	0.9987
		Z	0.9986	0.9986	0.9988	0.9984	0.9991	0.9986
		MinMax	0.9994	0.9994	0.9994	0.9993	0.9996	0.9994
	7: DDoS	No	0.9994	0.9993	0.9994	0.9992	0.9997	0.9993
		Z	0.9993	0.9992	0.9993	0.9992	0.9995	0.9992
		MinMax	0.9994	0.9993	0.9986	0.9979	0.9992	0.9993
	8: Merged (all types)	No	0.9970	0.9964	0.9925	0.9896	0.9954	0.9964
		Z	0.9990	0.9986	0.9975	0.9970	0.9980	0.9986
		MinMax	0.8489	0.9220	0.2946	0.1727	**1.0000**	0.9220
	1: FTP / SSH bruteforce	No	0.8494	0.9223	0.2937	0.1721	**1.0000**	0.9223
		Z	0.8495	0.9224	0.2886	0.1686	**1.0000**	0.9224
		MinMax	0.8380	0.7869	0.7291	0.9325	0.5986	0.7869
	2: DoS / Heartbleed	No	0.8370	0.7858	0.7275	0.9324	0.5964	0.7858
		Z	0.8376	0.7868	0.7292	0.9330	0.5984	0.7868
		MinMax	0.8733	0.9358	0.1683	0.0919	**1.0000**	0.9358
	3: Web attacks	No	0.8739	0.9362	0.1657	0.0904	**1.0000**	0.9362
		Z	0.8761	0.9373	0.1680	0.0917	**1.0000**	0.9373
		MinMax	0.9695	0.8776	0.0075	0.0038	0.7857	0.8776
	4: Infiltration	No	0.9702	0.8697	0.0070	0.0035	0.7692	0.8697
ncentroid		Z	0.9376	0.9233	0.0034	0.0017	0.9091	0.9233
		MinMax	0.9644	0.8139	0.2713	0.1707	0.6603	0.8139
	5: Botnet	No	0.9638	0.7924	0.2584	0.1634	0.6174	0.7924
		Z	0.9635	0.7894	0.2564	0.1622	0.6117	0.7894
		MinMax	0.9659	0.9608	0.9708	0.9433	1.0000	0.9608
	6: Portscan	No	0.9654	0.9600	0.9704	0.9426	1.0000	0.9600
		Z	0.9667	0.9615	0.9714	0.9445	1.0000	0.9615
		MinMax	0.9429	0.9356	0.9512	0.9069	1.0000	0.9356
	7: DDoS	No	0.9433	0.9364	0.9513	0.9070	1.0000	0.9364
		Z	0.9425	0.9354	0.9507	0.9060	1.0000	0.9354
		MinMax	0.7975	0.8247	0.6285	0.4921	0.8696	0.8247
	8: Merged (all types)	No	0.8856	0.9273	0.7743	0.6333	0.9962	0.9273
		Z	0.6990	0.7024	0.4810	0.3642	0.7080	0.7024

Table 7: Intel Core i5-4690 binlr, linsvc and rbfsvc full results.

algorithm	day	scaling	accuracy	balanced accuracy	F1	precision	recall	ROC-AUC
binlr		MinMax	0.9841	0.9906	0.7959	0.6621	**0.9976**	0.9906
	1: FTP / SSH bruteforce	No	0.8222	0.9081	0.2586	0.1485	**0.9998**	0.9081
		Z	0.9937	0.9960	0.9078	0.8322	**0.9985**	0.9960
		MinMax	0.9958	0.9965	0.9943	0.9899	0.9988	0.9965
	2: DoS / Heartbleed	No	0.8814	0.8732	0.8383	0.8336	0.8430	0.8732
		Z	0.9995	0.9996	0.9993	0.9985	1.0000	0.9996
		MinMax	0.9848	0.9923	0.6271	0.4568	1.0000	0.9923
	3: Web attacks	No	0.9480	0.9277	0.3086	0.1859	0.9068	0.9277
		Z	0.9970	0.9978	0.8958	0.8122	0.9986	0.9978
		MinMax	0.9905	0.9952	0.0257	0.0130	1.0000	0.9952
	4: Infiltration	No	0.0001	0.5000	0.0003	0.0001	1.0000	0.5000
		Z	0.9969	0.9151	0.0641	0.0333	0.8333	0.9151
		MinMax	0.9723	0.9791	0.4227	0.2690	0.9861	0.9791
	5: Botnet	No	0.7737	0.8818	0.0828	0.0432	0.9923	0.8818
		Z	0.9820	0.9856	0.5315	0.3633	0.9892	0.9856
		MinMax	0.9949	0.9941	0.9955	0.9912	0.9998	0.9941
	6: Portscan	No	0.9557	0.9489	0.9624	0.9276	1.0000	0.9489
		Z	0.9985	0.9984	0.9987	0.9980	0.9995	0.9984
		MinMax	0.9974	0.9972	0.9977	0.9965	0.9989	0.9972
	7: DDoS	No	0.9279	0.9193	0.9389	0.8861	0.9984	0.9193
		Z	0.9987	0.9986	0.9988	0.9981	0.9995	0.9986
		MinMax	0.9433	0.9566	0.8719	0.7863	0.9784	0.9566
	8: Merged (all types)	No	0.8232	0.7416	0.5750	0.5460	0.6071	0.7416
		Z	0.9498	0.9627	0.8853	0.8045	0.9841	0.9627
linsvc		MinMax	0.9927	0.9957	0.8948	0.8103	0.9989	0.9957
	1: FTP / SSH bruteforce	No	0.9753	0.9867	0.7154	0.5572	0.9989	0.9867
		Z	0.9973	0.9979	0.9582	0.9210	0.9985	0.9979
		MinMax	0.9991	0.9993	0.9988	0.9977	0.9999	0.9993
	2: DoS / Heartbleed	No	0.9964	0.9971	0.9951	0.9905	0.9998	0.9971
		Z	1.0000	1.0000	0.9999	0.9999	1.0000	1.0000
		MinMax	0.9951	0.9975	0.8395	0.7233	**1.0000**	0.9975
	3: Web attacks	No	0.9952	0.9969	0.8407	0.7260	**0.9986**	0.9969
		Z	0.9991	0.9995	0.9645	0.9313	**1.0000**	0.9995
		MinMax	0.9877	0.9522	0.0184	0.0093	0.9167	0.9522
	4: Infiltration	No	0.9918	0.8709	0.0225	0.0114	0.7500	0.8709
		Z	0.9993	0.9580	0.2418	0.1392	0.9167	0.9580
		MinMax	0.9810	0.9820	0.5164	0.3502	0.9831	0.9820
	5: Botnet	No	0.9229	0.9611	0.2108	0.1178	1.0000	0.9611
		Z	0.9837	0.9872	0.5562	0.3867	0.9908	0.9872
		MinMax	0.9969	0.9964	0.9973	0.9947	0.9998	0.9964
	6: Portscan	No	0.9858	0.9866	0.9874	0.9946	0.9802	0.9866
		Z	0.9992	0.9993	0.9993	0.9996	0.9991	0.9993
		MinMax	0.9974	0.9972	0.9977	0.9963	0.9990	0.9972
	7: DDoS	No	0.9441	0.9389	0.9514	0.9185	0.9868	0.9389
		Z	0.9995	0.9995	0.9995	0.9996	0.9994	0.9995
rbfsvc		MinMax	0.9803	0.9892	0.7586	0.6116	0.9987	0.9892
	1: FTP / SSH bruteforce	No	0.9698	0.5137	0.0533	1.0000	0.0274	0.5137
		Z	0.9996	0.9997	0.9936	0.9875	0.9998	0.9997
		MinMax	0.9933	0.9932	0.9908	0.9887	0.9929	0.9932
	2: DoS / Heartbleed	No	0.6417	0.5088	0.0346	1.0000	0.0176	0.5088
		Z	0.9999	0.9999	0.9998	0.9997	0.9999	0.9999
		MinMax	0.9779	0.9874	0.5361	0.3666	0.9972	0.9874
	3: Web attacks	No	0.9872	0.5000	0.0000	0.0000	0.0000	0.5000
		Z	0.9996	0.9998	0.9843	0.9690	1.0000	0.9998
		MinMax	0.9885	0.9526	0.0197	0.0100	0.9167	0.9526
	4: Infiltration	No	0.9999	0.5000	0.0000	0.0000	0.0000	0.5000
		Z	0.9998	0.7499	0.3429	0.2609	0.5000	0.7499
		MinMax	0.9677	0.9822	0.3885	0.2412	0.9969	0.9822
	5: Botnet	No	0.9902	0.5239	0.0912	1.0000	0.0478	0.5239
		Z	0.9951	0.9914	0.8053	0.6797	0.9877	0.9914
		MinMax	0.9919	0.9907	0.9929	0.9862	0.9997	0.9907
	6: Portscan	No	0.5687	0.5018	0.7245	0.5680	1.0000	0.5018
		Z	0.9997	0.9997	0.9997	0.9998	0.9996	0.9997
		MinMax	0.9963	0.9961	0.9967	0.9952	0.9981	0.9961
	7: DDoS	No	0.5582	0.5039	0.7152	0.5567	1.0000	0.5039
		Z	0.9999	0.9999	0.9999	1.0000	0.9997	0.9999