A Study on the Effect of Feature Selection on Malware Analysis using Machine Learning

Kehinde Oluwatoyin Babaagba
Edinburgh Napier University
Scotland, UK EH10 5DT
kehinde.babaagba@napier.ac.uk

Samuel Olumide Adesanya
Department of Mathematical Sciences,
Redeemer's University Ede, Nigeria
adesanyaolumide@yahoo.com

ABSTRACT
In this paper, the effect of feature selection in malware detection using machine learning techniques is studied. We employ supervised and unsupervised machine learning algorithms with and without feature selection. These include both classification and clustering algorithms. The algorithms are compared for effectiveness and efficiency using their predictive accuracy, among others, as performance metric. From the studies, we observe that the best detection rate was attained for supervised learning with feature selection. The supervised learning algorithm used was Multilayer Perceptron (MLP) algorithm. The analysis also reveals that our system can detect viruses from varying sources.

CCS Concepts
• Computing methodologies→Machine learning; Feature selection • Security and privacy→Malware and its mitigation.

Keywords
Malware Detection, Feature Selection and Machine Learning

1. INTRODUCTION
In recent times, malware detection and analysis are becoming key issues. This is because malware is increasingly posing a threat in a number of computer systems and networks owned by companies and individuals, coupled with the fact that malicious software can easily be created and launched. Upon analyzing common security risks, it can be observed that the rate at which Internet crimes are increasing surpasses the strategies most companies employ in curbing them. Each year, cyber criminals launch novel attacks that are capable of causing more harm than the previous years.

It is evident that hundreds of millions of new samples of malware such as computer viruses, among other malicious software, have been created in recent years. This implies that almost one million new threats are released daily. In view of this, several researchers have put in a lot of effort in analyzing malware. One common definition of malware is that of [1], that explained a malware as any kind of code modification by a software system deliberately aimed at damaging or preventing a system from performing optimally. [2] also described malware as a term that refers generally to all forms of spywares, viruses, trojans, among other harmful and malicious software. [3] categorized any program that has a malevolent objective as a malware.

Malware are generally created to compromise the confidentiality, integrity, or availability of the data/information in a computer system or network. Since it is evident that better hypotheses can be made upon knowing what the malware does, some of the categories in which most malware fall into as explained by [4] are outlined herewith: backdoor, botnet, downloader, information-stealing malware, launcher, rootkit, scareware, spam-sending malware, viruses, worms, trojan horses.

Interestingly, computer networks especially the internet are increasingly becoming determining factors in the smooth running of many organizations, hence the need to secure them is increasingly important. In the previous paragraphs, we have defined malware. However, securing our computer networks will require detecting these malware. Two basic approaches to malware analysis and detection include: static analysis (observing the malware without running it), and dynamic analysis (observing the malware while running it). Several malware analysts have proposed static analysis techniques for malware detection. However, these techniques can be problematic.

[5] explored the shortcomings of static analysis for detecting malware, and they argued that the use of pattern matching to identify malware can be easily evaded by simply changing the code structure. [6] explained that the diversity and amount of malicious software variants severely undermine the effectiveness of classical signature-based detection. [3] presented a malware detection algorithm that helps in curbing the limitation caused by including instruction semantics to detect malicious program characteristics. From their experimental evaluation, we see that their malware detection algorithm can discover various kinds of malware with a reasonably low run-time overhead. Also, their semantics-aware malware detection algorithm is resilient to prevalent obfuscation techniques used by hackers. Interested readers can go through refs [[7], [8], [9], [10], [11] and [12]] for more literature on malware detection.

Due to the limitations of the static means of malware detection and analysis, the need for intelligent approaches to malware detection is thus imperative. One of such intelligent approaches is machine learning which can be seen as the acquisition of structural descriptions from examples. The kind of descriptions found can be used for prediction, explanation, and understanding [13]. At the forefront of this research is the work done by [14] who proposed a method of identifying previously unseen malware by collectively classifying them. From their work, we see that most machine learning models try to identify malicious software by training classification algorithms. They do the training using datasets that consist of many typical features of malicious code. They pointed out the fact that using Byte n-gram representation, for instance, we can train our machine learning classifiers to be able to detect unknown malicious software.

[15] introduced a scalable clustering approach to detect and group malware samples that demonstrate similar behavior. The aim of their system was to cluster large groups of malware instances on the basis of their behavioral structure. Their system attempted to find a partitioning of a specific set of malware software to ensure that subsets share some common traits. [16] explained that some of

the machine learning processes required for malware analysis include firstly taking characteristic features of all binary files in the training and test dataset. In the training set there will be different combinations of malware types and clean files. Machine learning algorithms are then applied to the aforementioned, tuning the necessary parameters. Finally, the various processes in the malware detection on the training dataset are analyzed. [17] proposed a methodology for detecting malicious office documents using machine learning techniques. The office documents studied were XML-based and they achieved a high detection rate of malicious content comparable to the best antivirus engines. The works of [[18], [19], [20] and [21]] provide resource materials for further reading.

The proposed work involves the use of machine learning in detecting malware with an aim to study the effect of feature selection. In this work, the use of supervised machine learning with feature selection produced the best results. Supervised learning is referred to as learning with a teacher. It is a type of learning comprising of both input and output variables and we employ an algorithm to learn the function mapping the inputs to the outputs. In this type of learning, labelled examples are available [22]. Motivated by the works of [15],[16],[17],[18],[19],[20] and [21], the specific objective of this write-up is to analyze the effect of feature selection in malware analysis using machine learning which is important and to the best of our knowledge, after exhaustive survey of literature, has not been addressed in this manner in other literature, for which reason the current research is essential.

In order to carry out the analysis, we use Virustotal which contains a collection of antivirus search engines, for the static analysis. We go further to carryout dynamic analysis using a sandbox. Finally, we use some feature selection and machine learning algorithms in the malware detection process and compare their performance. The rest of the paper is organized into four sections. Section two comprises of the research method. The third section is the evaluation and result description. Finally, section four concludes our findings.

2. RESEARCH METHOD

An experimental research methodology is adopted in this work. Firstly, we present some research questions, and then we evaluate our research outcomes in a bid to put our research questions to test. The research questions include:

- Is machine learning-based malware detection effective?
- Are dynamic malware detection methods like sandboxing effective?
- Is there a difference between supervised and unsupervised machine learning-based malware detection?
- Does feature selection affect the results of the machine learning-based malware detection?
- What metrics are best used in measuring the performance of the machine-learning algorithms?

The research is conducted in four phases as explained below:

Data Collection: We collected malware samples as well as clean samples and a total of 149 samples were analyzed. There were 68 malicious samples gotten from www.virusign.com; the clean samples (81) on the other hand, were system files located in the "System32" directory of a Windows XP operating system.

Table 1: Dataset Description

Action Period	Between November 2015- January 2016
Action Location	Website: www.virusign.com for malicious samples and "System32" directory of a Windows XP Operating System
Data Set Size	149 (68 malware and 81 benign samples)
Number of reports logs issued by the malware analysis tools	Two (one each for both static and dynamic analysis)

Static Analysis: We used Virustotal for the static analysis. We chose Virustotal because it consists of different updated antivirus engines which are used for static malware analysis. We uploaded the files and recorded the reports generated from each scan.

Sand-boxing: The dynamic analysis tool used was Malwr. Malwr is based on the Cuckoo sandbox and it has been tested and proven to be efficient for dynamic analysis. We uploaded the files to the Malwr site and downloaded an XML report of each scan action upon its analysis.

Data Transformation: The xml reports generated from the sand-box were parsed using JDOM parser and the required features selected. The parsed files were represented as comma-separated values to serve as input for machine learning.

Machine Learning: Finally, we carried out the machine learning task. Supervised and unsupervised learning were carried out. The supervised learning also known as learning with a teacher was done using Random Forest, Decision Table, Bayesian Classifiers, Multilayer Perceptron (MLP), LazyIBK and LogitBoost algorithms. The Unsupervised learning was done using EM (Expectation-Maximization) algorithm.

3. RESULT AND DISCUSSION

One of the key objectives of this research was to study the behaviour of malware through learning and close observation of their features. The results of the dynamic analysis carried out in the sandbox show some API calls, which have been modeled as attributes that malicious and clean files make. These attributes were used to learn the behavior of the dataset.

WEKA API was used for feature selection and machine learning; we employed Information Gain algorithm for feature selection, Random Forest, Decision Table, Bayesian Classifiers, Multilayer Perceptron (MLP), LazyIBK and LogitBoost were used for supervised learning; and Estimation-Maximization algorithm was employed for unsupervised learning.

We began the machine learning by running experiments with the dataset with and without feature selection in order to study the effect of feature selection. Then, we compared the results. The experimental procedure is outlined as follows: Firstly, we carried out supervised learning with feature selection. Then, we carried out unsupervised learning with feature selection. Furthermore, we carried out supervised learning without feature selection. Finally, unsupervised learning without feature selection was done.

3.1 Evaluation Metric

In order to measure the performance of the algorithms, we used the following metrics [23]:

Classification Accuracy: This is used to determine the machine learning algorithm's accuracy. It is defined as:

number of correct predictions / total number of predictions

Confusion Matrix: This is a matrix that gives a holistic view of the algorithm's performance. It is described in table:2.

Table 2: Confusion Matrix

	Positive	Negative
True	True Positive	True Negative
False	False Positive	False Negative

Area Under the Curve (AUC): As the name suggests, AUC is the area under the curve of the plot of false positive rate ((**FalsePositive**)/(**FalsePositive** + **TrueNegative**)) and true positive rate ((**TruePositive**)/(**FalseNegative** + **TruePositive**)) at various points within [0, 1].

F1 Score: This represents the harmonic mean between the precision and recall values.

Training Time: This represents the time taken to train the model.

3.2 Machine Learning Algorithms

The Supervised Learning Algorithms used are described below.

Random Forest: It also referred to as random decision forest. This is a machine learning ensemble that combines several algorithms to derive better learning results [24].

Decision Table: This is a means of knowledge representation using tables in which outcomes are jointly determined by a group of conditions [25].

Bayesian Classifiers: Bayesian classifiers e.g. Bayes Networks work by assigning the most probable class to a particular example described by its feature vector. Assuming that characteristics and features are not dependent on a given class makes learning such classifiers greatly simplified [26].

Multilayer Perceptron (MLP): This is a deep artificial neural network with several layers comprising at least of an input layer, a hidden layer and an output layer. The input layer is often used for input reception, the hidden layer is the computation engine and the output layer is used for decision making or predictive analysis [27].

LazyIBK: This is one of the simplest machine learning algorithms which implements the k-Nearest Neighbor algorithm. It is an instance-based learning algorithm whose function is a local approximation. It does not compute any values until classification [28].

LogitBoost: This is a boosting classification algorithm that reduces the logistic loss. It greedily optimizes the classification probability, provided that the base learner reduces the squared error [29].

The Unsupervised Learning Algorithm used is described below.

EM (Expectation-Maximization) algorithm: This is a way of determining an approximation of the maximum likelihood of parameters in a distribution. It is suited for data sets containing values that are either missing or incomplete [30].

Feature Selection Algorithm

The filter method of feature selection is a very straight- forward and less computationally expensive method of feature selection [31] which is why it was chosen as the feature selection algorithm to be used. The filter method used Information Gain for attribute evaluation. This assesses the value of an attribute by evaluating the information gain with respect to the class. This is given in the equation below;

$$\text{InfoGain}(\text{Class}, \text{Attribute}) = \text{K}(\text{Class}) - \text{K}(\text{Class}|\text{Attribute})$$

where K is the information entropy

3.3 Machine Learning without Feature Selection

The results of the unsupervised and supervised machine learning algorithms without feature selection are described with the table and diagram below. Recall that the machine learning algorithms used include Random Forest, Decision Table, Bayesian Classifiers, Multilayer Perceptron (MLP), LazyIBK and LogitBoost for the supervised learning and EM algorithm for the unsupervised learning.

The results from table:3 and figure:1 show that for machine learning without feature selection with algorithms Random Forest, Decision Table, Bayesian Network, Multilayer Perceptron, LazyIBK and LogitBoost, the values obtained for accuracy are 73.1544%, 76.5101%, 68.4564%, 77.1812%, 74.4966% and 76.5101% respectively.

From the values obtained for accuracy and other metrics such as TPR, FPR, AUC and F1Score, we observe that Multilayer Perceptron performs the best with an accuracy of 77.1812% for supervised learning without feature selection. It however takes the longest time to train, with a training time of 14.87seconds.

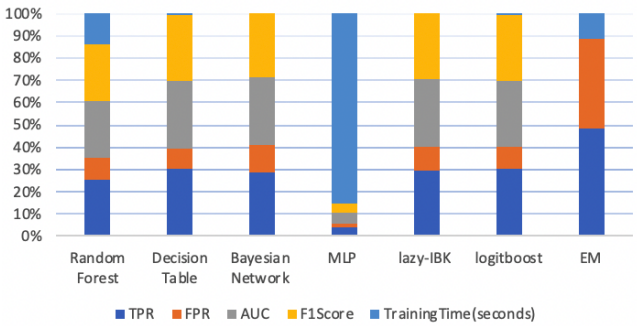

Figure 1: Machine Learning without Feature Selection

Table 3:Machine Learning without Feature Selection

Algorithms	Accuracy	TPR	FPR	AUC	F1Score	Time(secs)
Supervised Learning						
Random Forest	73.1544	0.732	0.287	0.766	0.728	0.41
Decision Table	76.5101	0.765	0.242	0.752	0.765	0.01
Bayesian Network	68.4564	0.685	0.293	0.749	0.681	0
MLP	77.1812	0.772	0.232	0.786	0.772	14.87
LazyIBK	74.4966	0.745	0.271	0.759	0.742	0
Logitboost	76.5101	0.765	0.240	0.746	0.765	0.01
Unsupervised Learning						
EM	54.3624	0.544	0.456	-	-	0.13

3.4 Machine Learning with Feature Selection

In this section, we follow the processes described in subsection:3.3 above. However, we employ feature selection. Information Gain algorithm was used for feature selection. The result description from table:4 and figure:2 shows that for machine learning with feature selection with algorithms Random Forest, Decision Table, Bayesian Network, Multilayer Perceptron, LazyIBK and LogitBoost, the values obtained for accuracy are 74.4966%,

76.5101%, 69.7987%, 77.1812%, 75.1678% and 77.1812% respectively.

According to the values obtained for accuracy and other metrics such as TPR, FPR, AUC and F1Score, we observe that Multilayer Perceptron still performs the best with an accuracy of 77.1812% for supervised learning with feature selection. It however still takes the longest time to train, with a training time of 14.87seconds.

In the previous section, we focused solely on the effect of feature selection on supervised machine learning. From the tables:3 and 4 as well as the figures:1 and 2, we can also see the effect of feature selection on unsupervised machine learning. It can be seen that when feature selection is applied to EM algorithm, the accuracy increases to 74.4966%. This is in contrast to the 54.5624% accuracy obtained when no feature selection was applied.

In general, it can be noted that for supervised learning, the use of feature selection does not very noticeably increase the accuracy of the algorithms. In algorithms such as Decision Table and MLP, the results for accuracy even remains the same with or without feature selection. However, for unsupervised learning we see the accuracy jump from 54.3624% to 74.4966%. The best accuracy for both supervised and unsupervised machine learning with or without feature selection is obtained by MLP with an accuracy of 77.1812%.

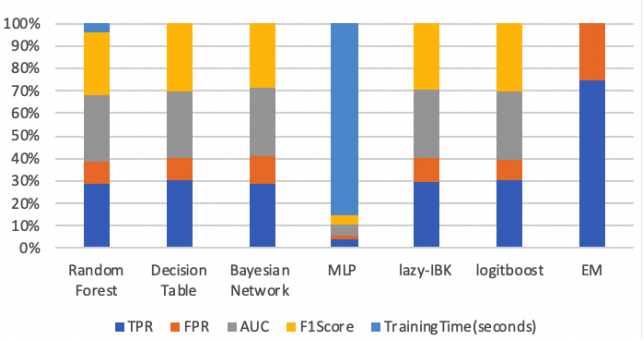

Figure 2: Machine Learning with Feature Selection

Table 4: Machine Learning with Feature Selection

Algorithms	Accuracy	TPR	FPR	AUC	F1Score	Time(secs)
Supervised Learning						
Random Forest	74.4966	0.745	0.271	0.767	0.742	0.1
Decision Table	76.5101	0.765	0.242	0.747	0.765	0
Bayesian Network	69.7987	0.698	0.284	0.733	0.696	0
MLP	77.1812	0.772	0.232	0.786	0.772	14.87
LazyIBK	75.1678	0.752	0.263	0.774	0.749	0
Logitboost	77.1812	0.772	0.234	0.761	0.772	0
Unsupervised Learning						
EM	74.4966	0.745	0.255	-	-	0

4. CONCLUSION

In this work, we developed a system that allows a malware analyst to analyze and detect malicious code using machine-learning techniques. Given the threat that malicious software in the form of viruses and trojan horses, just to name a few, continuously pose, it is obvious, as we have discussed that the static analysis tools normally used are inefficient in discovering these malware.

The use of the feature selection method based on Information Gain led to higher values for accuracy. Upon comparing the performance of the machine learning algorithms with and without feature selection, MLP emerged with the best result with an accuracy of 77.1812%, TPR of 0.772, FPR of 0.232, AUC of 0.786 and F1Score of 0.772.

The results show that machine learning-based malware detection is effective. We were also able to describe the difference between supervised and unsupervised machine learning in terms of their detection accuracy. Although in this work, almost all the performance metrics employed were useful in determining the best machine learning algorithm, some were more useful than others. The training time for instance, was not very relevant as most of the models were built in less than one second.

There are a number of areas in this work that are still open for further research work. In this work, a lot of focus was placed on viruses and its attacks, another area to look at would be on other forms of malware like trojans and spyware. Also, more samples can be analyzed, say thousands of samples, to observe the machine learning process more accurately.

5. REFERENCES

[1] G. McGraw and G. Morrisett. Attacking malicious code: A report to the infosec research council. IEEE Software, 17(5):33–41, Sep. 2000. ISSN 0740-7459. doi: 10.1109/52.877857.

[2] Amit Vasudevan and Ramesh Yerraballi. Spike: Engineering malware analysis tools using unobtrusive binary-instrumentation. In Proceedings of the 29th Australasian Computer Science Conference - Volume 48, ACSC '06, pages 311–320, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc. ISBN 1-920682-30-9.

[3] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant. Semantics- aware malware detection. In 2005 IEEE Symposium on Security and Privacy (S P'05), pages 32–46, May 2005. doi: 10.1109/SP.2005.20.

[4] Michael Sikorski and Andrew Honig. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. No Starch Press, San Francisco, CA, USA, 1st edition, 2012. ISBN 1593272901, 9781593272906.

[5] A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), pages 421–430, Dec 2007. doi: 10.1109/ACSAC.2007.21.

[6] Konrad Rieck, Thorsten Holz, Carsten Willems, Patrick Du¨ssel, and Pavel Laskov. Learning and classification of malware behavior. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2008. ISBN 3540705414. doi: 10.1007/978- 3- 540- 70542- 0_6.

[7] Fei Tong and Zheng Yan. A hybrid approach of mobile malware detection in android. Journal of Parallel and Distributed Computing, 103:22 – 31, 2017. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2016.10.012.

[8] Shamsul Huda, Jemal Abawajy, Mamoun Alazab, Mali Abdollalihian, Rafiqul Islam, and John Yearwood. Hybrids of support vector machine wrapper and filter-based framework for malware detection. Future Generation Computer Systems, 55: 376 – 390, 2016. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2014.06.001.

[9] Ping Wang and Yu-Shih Wang. Malware behavioral detection and vaccine development by using a support vector model classifier. Journal of Computer and System Sciences,

81(6):1012-1026, 2015. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2014.12.014.

[10] Mohd Faizal Ab Razak, Nor Badrul Anuar, Rosli Salleh, and Ahmad Firdaus. The rise of fimalwarefi: Bibliometric analysis of malware study. Journal of Network and Computer Applications, 75:58 – 76, 2016. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2016.08.022.

[11] Nizar Kheir. Behavioral classification and detection of malware through http user agent anomalies. Journal of Information Security and Applications, 18(1):2 – 13, 2013. ISSN 2214-2126. doi: https://doi.org/10.1016/j.jisa.2013.07.006.

[12] Adriana Leite and Rosario Girardi. A hybrid and learning agent architecture for network intrusion detection. Journal of Systems and Software, 130:59 – 80, 2017. ISSN 0164-1212. doi: https://doi.org/10.1016/j.jss.2017.01.028.

[13] I H Witten, Eibe Frank, and M A Hall. Data Mining Practical Machine Learning Tools and Techniques. 2005. ISBN 0080890369. doi: 0120884070,9780120884070.

[14] Igor Santos, Yoseba K Penya, Jaime Devesa, and Pablo G Bringas. N-Grams-Based File Signatures for Malware Detection. In Iceis 2009: Proceedings of the 11Th International Conference on Enterprise Information Systems (ICEIS), 2009. ISBN 978-989-8111-85-2. doi: 10.1016/j.ijpharm.2015.02.045.

[15] Ulrich Bayer, Paolo Milani Comparetti, Clemens Hlauschek, Christopher Kruegel, and Engin Kirda. Scalable, Behavior-Based Malware Clustering. Proceedings of the Network and Distributed System Security Symposium, NDSS 2009, 2009. ISSN 00278424. doi: 10.1073/pnas.1835725100.

[16] D. Gavrilu, M. Cimpoeu, D. Anton, and L. Ciortuz. Malware detection using machine learning. In 2009 International Multiconference on Computer Science and Information Technology, pages 735–741, Oct 2009. doi: 10.1109/IMCSIT.2009. 5352759.

[17] Aviad Cohen, Nir Nissim, Lior Rokach, and Yuval Elovici. Sfem: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods. Expert Systems with Applications, 63:324 – 343, 2016. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2016.07.010.

[18] Rafiqul Islam, Ronghua Tian, Lynn M. Batten, and Steve Versteeg. Classification of malware based on integrated static and dynamic features. Journal of Network and Computer Applications, 36(2):646 – 656, 2013. ISSN 1084-8045. doi: https: //doi.org/10.1016/j.jnca.2012.10.004.

[19] JuiHsi Fu, PoChing Lin, and SingLing Lee. Detecting spamming activities in a campus network using incremental learning. Journal of Network and Computer Applications, 43:56 – 65, 2014. ISSN 1084-8045. doi: https://doi.org/10. 1016/j.jnca.2014.03.010.

[20] Rafiqul Islam and Jemal Abawajy. A multi-tier phishing detection and filtering approach. Journal of Network and Computer Applications, 36(1):324 – 335, 2013. ISSN 1084-8045. doi: https://doi.org/10.1016/j.jnca.2012.05.009.

[21] Martin Grill, Tom Pevn, and Martin Rehak. Reducing false positives of network anomaly detection by local adaptive multivariate smoothing. Journal of Computer and System Sciences, 83(1):43 – 57, 2017. ISSN 0022-0000. doi: https://doi.org/10. 1016/j.jcss.2016.03.007.

[22] Taiwo Oladipupo Ayodele. Types of machine learning algorithms. In Yagang Zhang, editor, New Advances in Machine Learning, chapter 3. IntechOpen, Rijeka, 2010. doi: 10.5772/9385. URL https://doi.org/10.5772/9385.

[23] Aditya Mishra. Metrics to Evaluate your Machine Learning Algorithm, 2018. URL https://towardsdatascience.com/ metrics- to- evaluate- your- machine- learning- algorithm-f10ba6e38234.

[24] Katrina Wakefield. A guide to machine learning algorithms and their applications, 2018. URL https://www.sas.com/en{ }gb/insights/articles/analytics/ machine- learning-algorithms.html.

[25] E Lima, C Mues, and B Baesens. Domain knowledge integration in data mining using decision tables: case studies in churn prediction. Journal of the Operational Research Society, 60(8):1096–1106, 2009. ISSN 1476-9360. doi: 10.1057/jors.2008. 161.

[26] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. Machine Learning, 29(2):131–163, Nov 1997. ISSN 1573-0565. doi: 10.1023/A: 1007465528199.

[27] Skymind. A Beginner's Guide to Multilayer Perceptrons (MLP), 2018. URL https://skymind.ai/wiki/multilayer-perceptron.

[28] Guilherme O. Campos, Arthur Zimek, Jo¨rg Sander, Ricardo J. G. B. Campello, Barbora Micenkova´, Erich Schubert, Ira Assent, and Michael E. Houle. On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study. Data Mining and Knowledge Discovery, 30(4):891–927, Jul 2016. ISSN 1573-756X. doi: 10.1007/s10618-015-0444-8. URL https://doi.org/10.1007/ s10618- 015- 0444- 8.

[29] S. B. Kotsiantis and P. E. Pintelas. Logitboost of simple Bayesian classifier. In Informatica (Ljubljana), 2005.

[30] Maya R. Gupta and Yihua Chen. Theory and use of the em algorithm. Found. Trends Signal Process., 4(3):223–296, March 2011. ISSN 1932-8346. doi: 10.1561/ 2000000034.

[31] M Shardlow. An Analysis of Feature Selection Techniques. Student- net.Cs.Manchester.Ac.Uk, 2007.