Available Online at www.ijcsmc.com/journal

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 11, Issue. 11, November 2022, pg.35 – 47

ANOMALY DETECTION IN CYBER-PHYSICAL SYSTEMS BASED ON HYBRID NEURAL NETWORKS MODELS

Mustafa Abdalkhudhur Jasim[1]; Maha Khalid Khadhum[2]; Ahmed Abdulkhudher Jasim[3]; Eman Khaled Khalaf[4]

Al-Hadi University College[1,2,3,4]
Email: [1] mustafa.a.jasim@huc.edu.iq; [2] maha.khalid@huc.edu.iq; [3] ahmed.a@huc.edu.iq; [4] eman.khalid@huc.edu.iq
ORCiD: [1] https://orcid.org/0000-0002-7973-866X; [2] https://orcid.org/0000-0001-6337-1819; [4] https://orcid.org/0000-0002-2567-2140

DOI: https://doi.org/10.47760/ijcsmc.2022.v11i11.005

Abstract: ***Nowadays cyber-physical systems are widely used in different application domains. In parallel, machine learning algorithms are used widely to detect the anomalies in the behavior of these systems. However, this detection is limited to two states: normal behavior and faulty functioning. The goal of this thesis was to create a hybrid neural network that could distinguish between three states of a cyber-physical system: normal behavior, a fault, and an attack. First and foremost, a power system is provided as a working example. Then, three potential ways for achieving the initial goal were presented: changing the values of features, using distances between samples, and using the Hidden Markov Model. They were tested on three different machine learning classifiers - Decision Tree, Random Forest and Multilayer Perceptron. Later, various tools for feature analysis are presented and an algorithm to find the features that contributed the most into the false predictions is described. Finally, three solutions to the initial problem are presented and evaluated. For Decision Tree classifier the most efficient enhancement method was using the proposed method that works on distances between samples. For Random Forest classifier, the most effective method was modifying the values of features. And for Multilayer Perceptron classifier all the proposed methods failed.***

Keywords: ***Machine Learning, Cyber-Physical Systems, Anomaly Detection, Random Forest, Power System***

1. INTRODUCTION

Cyber-Physical Systems (CPS) are nowadays widely used in different application domains, such as smart-homes, smart-cities, hospitals, etc. They are mainly composed of two entities: a cyber-part consisting in a computing and networking component, and a physical part consisting in different controllers and sensors. The existence of a connected cyber part implies its susceptibility to multiple cyber threats. The malfunctioning of these systems, due to a cyber-threat, can cause severe impacts on the real life and the safety of the community, for example a blackout or water contamination. That is why many algorithms have been designed for the

security monitoring of those systems, in particular the anomaly and attack detection. Nowadays, machine and deep learning algorithms are used to detect those anomalies and intrusions.
However, this is not the first time such a fusion is examined. In the literature various approaches of the fusion of neural networks with theory-based models were presented. Due to research, five different methods were found. First, an approach that add physic based features like [1]. Then, another approach generating huge datasets artificially and learn the machine learning model to find the suspicious values of sensors rather than deciding in which status is the system, like in [2]. Third, a method that reduces the dimensionality of the problem, like in [3]. Fourth, a method that works on data defined over graphs, like in [4], and finally an approach that decomposes the problem into smaller ones, like in [5].

The first mentioned approach takes into consideration a physical model of the system. Based on the sensors' data, the physical model will generate additional physical features. Finally, those physical features, along with sensor's data are passed to the machine learning algorithm.

In [1] this approach was used to model water temperature in a lake at varying times and depths. As physical model, they used the state-of-the-art general lake model [6] to obtain the temperatures obtained via simulation. Moreover, their approach incorporates a loss-function composed of three main components: the empirical error, the structural error and the physical inconsistency. During the training process, the machine learning algorithm strives to reduce the loss function. The empirical error assesses the difference between anticipated and expected values, whereas the structural error is proportional to model complexity, whereas the physical inconsistency evaluates if the predicted value is physically correct. The authors of [1] have proven that their approach gives better results that other attempts to resolve the same problem.

The second approach, on other hand, presented in [2], takes into consideration an attack generation engine. This engine takes as an input a configuration file describing the system then it creates thousands of samples with various features describing sensors readings and others physic based. Then, the attacks are simulated by spoofing the values of samples. The dataset obtained this way is then trained to a machine learning algorithm, which will be able to tell to which extent the values of the features are physically correct. All the discussed methods have one in common: they can only detect the malfunctioning in general, without differentiating between attacks and a normal fault. In this paper, an attempt will be made to develop an algorithm able to detect the system status - normal behaviour, a fault and an attack. In order to do that, the scope of this paper includes a description of an example of cyber physical system, the description of different machine learning algorithms along with the comparison of their performances, the analysis of the importance of features and finally the proposal of three different solutions to the initial problem. The final research topic is based on the necessity for a validation technique for each proposed unsupervised method.

• Question 1: Is there a reliable approach for detecting CPS anomalies using unsupervised techniques with little or no truth data?

• Question 2: With little or no truth data, what unsupervised semantic analysis of automobile time series is possible?

2. PROPOSED SYSTEM

A possible better solution for the posed problem may be using a physical model of the power system that could estimate on its own some features and combining it with a machine learning technique. There exist some python toolkits created to model a power system like PyPSA or panda power , but they are complex and require specific advanced technical skills. This paper in addition to all that, provides some other important conclusions. First, the different machine learning toolkits do not work always the same way and the results may differ from one toolkit to another, like in this case and the differences in results between Weka and scikit-learn. Second, in order to the results from a work to be reproducible, all the details concerning the used parameters and the version of the software must be provided, in other case it may be hard to get the same result. Finally, it exists plenty of tools for machine learning classifiers evaluation and each toolkit provides its own interpretation of results.

2.1 Power system as a CPS example

In order to focus on the implementation of the hybrid machine learning algorithm, a CPS, with ready to use datasets, was chosen from a list provided in [7]: the power system [8], which network diagram was represented in figure 1. The system is composed of two power generators who are alimenting the whole system. Intelligent Electronic Devices (IEDs) R1 to R4 and the breakers BR1 to BR4 can be found connected directly to those generators. Each IED switches its corresponding breaker when a fault is detected, valid or fake. The communication between the IEDs and the Substation Switch is done wirelessly. On the other hand, the Substation Switch is connected with the Primary Domain Controller (PDC) and the Control Room.

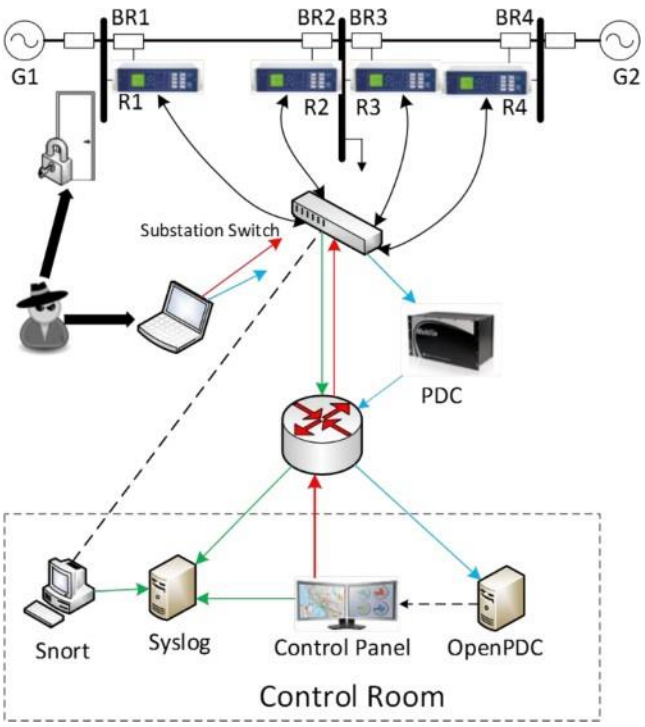


Figure 1: Power system network diagram

The operation of this power system can be described following 6 main scenarios:
• normal behavior,
• short-circuit,
• line maintenance,
• remotely opening the breakers (attack),
• disruption of fault protection system (attack),
• fault imitation (attack).

2.2 Contribution

A solid set of approaches and validation strategies to improve future research is also expected to assist the scientific community. Access to truth data for validating approaches and findings is a persistent difficulty for research in the CPS network domain. A possible better solution for the posed problem may be using a physical model of the power system that could estimate on its own some features and combining it with a machine learning technique. There exist some python toolkits created to model a power system like PyPSA [38] or panda power [39], but they are complex and require specific advanced technical skills. This study describes a set of strategies for automating the analysis of proprietary non-text network protocol payloads used by Cyber-Physical Systems (CPS). Unsupervised lexical analysis approaches for extracting logically different pieces of information from non-text payloads and then enumerating the associated and causal relationships that exist among that information are among such techniques.

2.3 Used machine learning algorithms

Before talking about the comparison of machine learning algorithms, the used classifiers are briefly described in this section in order to better understand how do they work. In parallel, values of classifiers' parameters used during all the tests are presented in tables 1, 2 and 3, and, in majority, they represent the default values found in Weka, in order to reproduce the results from [11].

2.3.1 Random Forest

Random Forest, as the name indicates, is an ensemble of a given number of trees, and more exactly decision trees. This given number of trees is set using the parameter numIterations in Weka and estimators in scikit-learn. Each of decision trees is created based on a randomly chosen set of features of the dataset. The number of features is given by the parameter *num Features* in Weka and *max_features* in scikit-learn, which indicate the way it is calculated - in this case as the logarithm of base 2 of the number of features.

On the other hand, decision trees represent a structure capable of determining the class of the predicted sample. It is composed of nodes connected to each other in a form of tree. Each node corresponds to a condition related to the features, for example if a particular feature is higher than a certain number. Each node, except the decision ones, has two child nodes corresponding to the answer as yes or no to the condition from from parent node. For each of two answers, particular classes are assigned, for example, given the case study in this work, yes answer can correspond to classes Attack and Natural and the answer no to class NoEvents. When running a tree on a particular sample, the nodes are followed given the answers on conditions, until arriving to the decision node. The class corresponding to this node is taken as the final output of

the decision tree. The number of layers of nodes is called the depth of the tree and it is set using the parameter *maxDepth* in both Weka and scikit-learn.

2.3.2 SVM

SVM, or more exactly Support Vector Machine, is a classifier that creates an N-dimensional space, in which all the samples from the training dataset are put, and then divided, geometrically, into regions, where each region corresponds to one class. The N dimensions of this space corresponds to all the features of the datasets and their transformations (for example a square of one of the features). Those transformations are done using a kernel function which can be set in Weka and scikit-learn using the kernelType/kernel parameter. Three kernel types are more commonly used: linear, polynomial and radial basis function (exponent based). The default kernel in both Weka and scikit-learn is radial basis function. Each type of kernel function comes with a set a parameter itself, like the degree in the case of polynomial and all of them can be set through the parameters of both used machine learning toolkits.

In fact, the transformations do not calculate additional dimensions for every sample, but calculate the distance between points in the N-dimensional space. This distance is mathematically defined as the dot product of the two points. The dot product on other hand is defined of the sum of products of coordinates for every dimension.

The division is made by determining so called hyperplane, which for example in a 2D features space is just a line. The points from each class that are the closest to the hyperplane are called support vector points, from where comes the name of this classifier [17] .

The algorithm is running within a set number of iterations (can be set within Weka and scikit-learn) dividing the training dataset to smaller tranches using the cross-validation technique. The goal is to find the hyperplane that way so the number of misclassified samples is the smallest in that particular iteration.

2.3.3 Naïve Bayes

Naïve Bayes is a classifier based on the Bayes theorem, which states that the probability of event A, given that the event B happened is equal to the product of the probability of B given that A occurred and the probability of A, divided by the probability of B:

$$P(A|B) = P(B|A)P' \frac{A}{P(B)} \qquad (1)$$

While the naivety of the algorithm is due to it does not taking into consideration the eventual relations between the features. The Naïve Bayes classifier can be both multinomial and gaussian. In the multinomial case the probabilities are calculated the classical way (number of occurrences over all samples), whereas in the gaussian case, the probabilities are taken from the gaussian distribution of a given event. For both Weka and scikit-learn the gaussian case were used.

2.4 Multilayer perceptron

The multilayer perceptron is a classifier inspired by the structure of the human brain. It is constituted from a set of interconnected neurons, which are divided into at least 2 layers: input layer, hidden layers and output layer. Each layer consists of a number of neurons. The input layer has always a number of neurons equal to the number of features, and the output layer has always this number equal to the number of outputs of the problem. That is why, in the analyzed case the input layer has 128 neurons and the output layer only one neuron corresponding to the status of the power plant (normal behaviour, natural fault, attack). Each of the mentioned neurons has a weight. The neurons are interconnected. The output of a neuron is considered as the input for the neurons of the following layer. Those neurons of the following layer calculate the weighted sum of the inputs and passes the results through a function, called activation function. The result obtained this way is considered as the output of the given neuron and goes to the following layer. The only exception is the input layer, which input is the values of features and its neurons just copy the input to the output.

The training process consists in updating the weights of the neurons during a predefined number of iterations called epochs. Before starting the training process, the initial weights values for hidden and output layers should be chosen. This choice can be done empirically or randomly. Moreover, a real positive number η called training step must be fixed.

The scikit-learn implementation of multilayer perceptron classifier is limited because it does not give a full control of each hidden layer and its activation function. It integrates however the possibility to determine the number of epochs called iterations and the training step called learning rate.

There exists multiple python frameworks offering much more flexibility when creating neural net- works like PyTorch or Keras framework, which were created in order to offer the possibility to create complex neural networks able to be used in large-scale applications. The mentioned frameworks will not be used in this work because this flexibility is not required in this case. However, it does not mean that it is not possible to make the results even better using those frameworks.

2.5 Metrics for classifiers comparison

All those classifiers give predictions that can be divided into 4 main groups. Assuming for a moment, for illustration purposes, that normal fault class is positive and attack class is negative, it could be differentiated between:

• True Positive predictions (TP): correct prediction of the normal fault class,

• True Negative predictions (TP): correct prediction of the attack class,

• False Positive predictions (FP): incorrect prediction of normal fault class,

• False Negative predictions (FN): incorrect prediction of attack class.

For that two classes, several metrics can be defined, that will be useful to compare between the used classifiers:

- **Accuracy**: ratio of correct classifications over the total number of samples, or in other words:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

- **Precision:** ratio of correct classifications for a particular class over all classifications that indicated that class, or:

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

- **Recall:** ratio of correct classifications for a particular class, over all samples corresponding for this class, or:

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

- **F-measure:** weighted average of precision and recall given by the equation:

$$f - measure = 2 \times \frac{Precison \times Recall}{Precision+Recall} \quad (4)$$

Given that those metrics work only with binary problems with a positive and a negative class, for multiclass problems the accuracy is always calculated as the ratio of true classifications over all classifications, however for precision, recall and f-measure, the average is calculated, and that in 3 different ways [29]:

• micro average: the metrics are determined globally by calculating true positives as all correct predictions, false negatives and false positives as all incorrect predictions (assuming two classes A and B, when A is misclassified as B is a false positive for A, and in the same time a false negative for B). In this case precision, recall, f-measure and accuracy have exactly the same value,

• macro average: the metrics are calculated for each class, the concerned class is considered as positive, while the sum of others as negative. Then their arithmetic mean value is calculated,

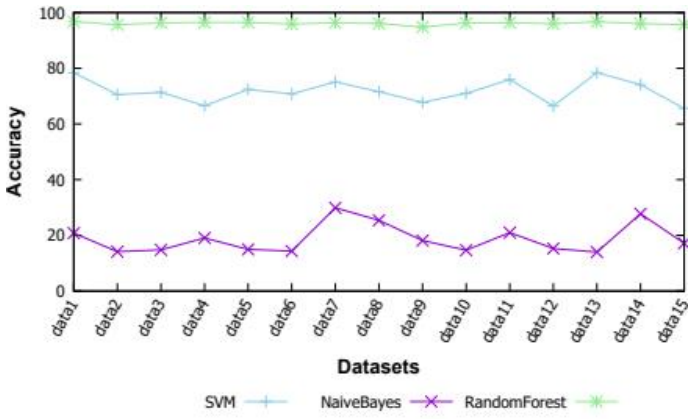
• weighted average: the metrics are calculated for each class, then it calculates their weighted average value by the number of true instances for each class.

3. RESULTS AND DISCUSSION

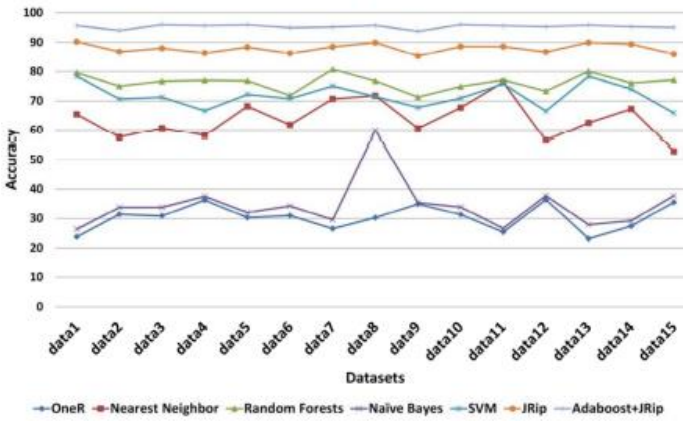
In order to reproduce the results presented in [11], Weka version 3.8.4 and scikit-learn version 0.23.1, running on Anaconda 3.18.11 with Python 3.7.6. final.0, were used. SVM in Weka comes from a package untitled libsvm available for this toolkit. The results are displayed in the same order as in [11], thus, the accuracy was presented in figures 2 and 3 for the three class, binary and all 37 class cases respectively. Then, the precision, recall and f-measure can be found in figures 2 and 3. For these three metrics the weighted average was used to obtain the results. For Naïve Bayes the precision and f-measure could not be extracted using Weka for unknown reason.

Figure 4 shows the accuracy values per dataset file for the three-class case using Weka, scikit- learn and compared to the original results for different classifiers. An accuracy superior to 50% is observed for SVM and Random Forest classifiers using Weka and for MLP and Random Forest using scikit-learn. The original results show that all the classifiers have an average accuracy superior to 50% except Naïve Bayes and One R classifiers. Figure 3 shows the accuracies in the binary case. The observations are similar, except for scikit-learn, where Naïve Bayes has this time an average accuracy superior to 50%. Figure 2 shows the accuracies for the multiclass datasets. The accuracies superior to 50% are Random Forest and SVM for Weka, Random Forest only for scikit-learn and Adaboost+JRip, JRip and Random Forest for the original results.

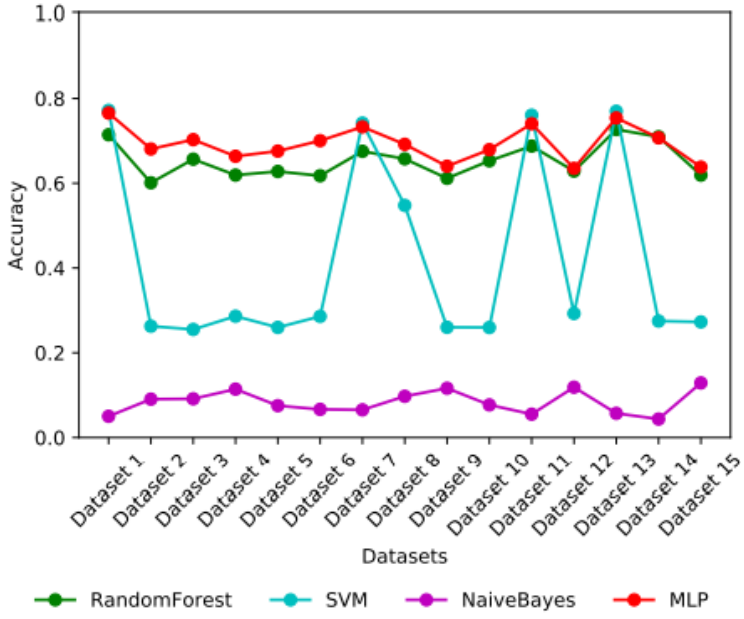
Figure 2 shows the precision metric values for different classifiers calculated using Weka and scikit-learn and compared to the original results. A precision superior to 0.5 is observed for all classifiers using Weka and for Random Forest and partially for MLP (binary and three-class), SVM (binary and three-class) and Naïve Bayes (binary) using scikit-learn. The original results, on other hand, show a precision superior to 0.5 for Random Forest, JRIP and Adaboost+JRIP and additionally for the binary case using SVM. Figure 3 shows the recall metric values. A recall superior to 0.5 is observed for Random Forest and SVM in Weka's results, for Random Forest, binary class Naïve Bayes, binary and 3-class MLP in scikit-learn's results. The original results show a value superior to 0.5 for One R, Naïve Bayes, JRip and Adaboost+JRip classifiers.



(a) Our attempt - Weka

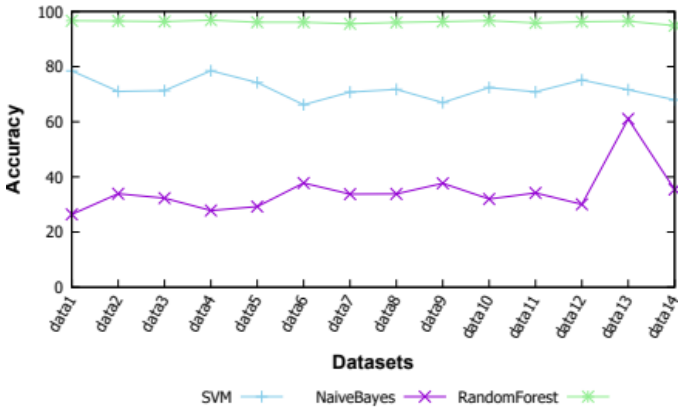


(b) Original results [11]

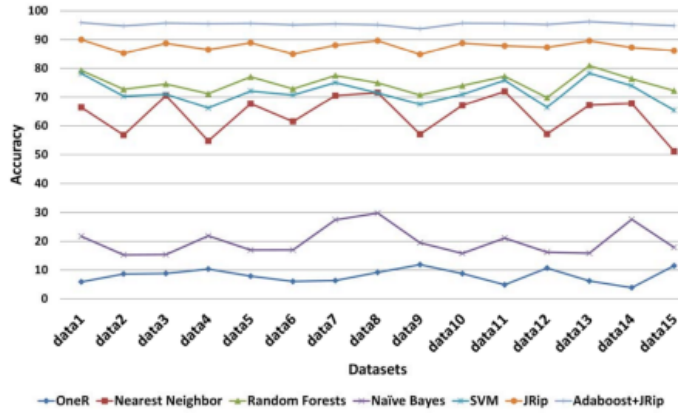


(c) Our attempt - scikit-learn

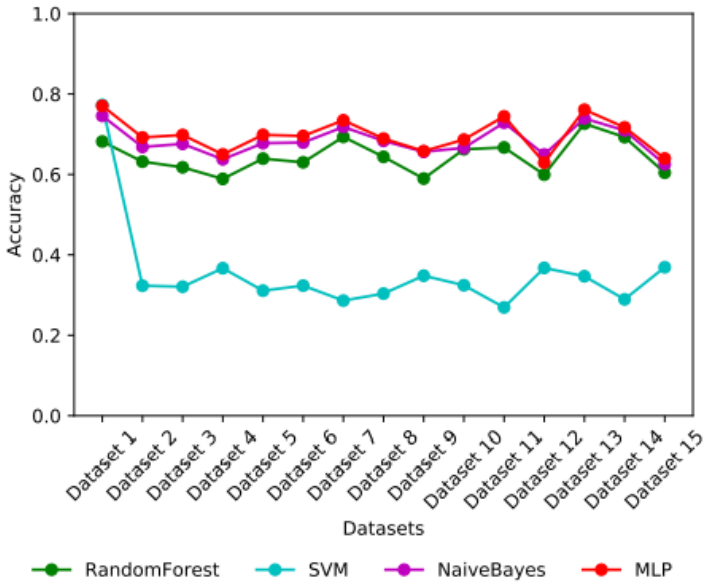
Figure 2: Plots showing the accuracy values per dataset file for different machine learning techniques using multiclass datasets of the power system calculated



(a) Our attempt - Weka



(b) Original results [11]



(c) Our attempt - scikit-learn

Figure 3: Plots showing the accuracy values per dataset file for different machine learning techniques using binary datasets of the power system calculated

The accuracies in the three-class case are quite similar on the three plots, except SVM in scikit- learn, what shows a huge discrepancy of accuracy between datasets. Moreover, a remarkably higher values of accuracy for Random Forest classifier are observed in the attempt using Weka, compared to other plots.

In the binary and multiclass cases, the Random Forest classifier in the attempts using Weka always shows higher values comparable to other plots. Scikit-learn, on other hand, for Random Forest classifier, gives results similar to those made by [11]. For SVM classifier, the discrepancy, this time, in the plots made by scikit-learn, is remarkably less important and the values oscillate around a constant value. For the binary case, this value is about 0.3 in scikit-learn, while in Weka the obtained value is 0.75, which makes it comparable to the original results in [11]. On the other hand, in the multiclass case, the value oscillates around 0.1, like it is the case in the results from [11]. In the attempt made in Weka the accuracy is much higher at about 0.6. For the Naïve Bayes classifier, the results are comparable, except the results obtained in scikit-learn in the binary case.

For precision, recall and f-measure, some similarities can be observed especially in the case of Random Forest classifier and in the case of Naïve Bayes for multiclass. The results are generally different for the three metrics. For binary and multi-class cases the values of metrics are similar, except Naïve Bayes classifier case. The most similar to the results from [11] are the results obtained using scikit-learn, despite the differences. The more efficient classifier is certainly Random Classifier, especially when using Weka. However, MLP classifier gives also quite good results.

3.1 Comparison Results

In order to reproduce the results presented in [11], Weka version 3.8.4 and scikit-learn version 0.23.1, running on Anaconda 3.18.11 with Python 3.7.6.final.0, were used. SVM in Weka comes from a package untitled libsvm available for this toolkit. The results are displayed in the same order as in [11], thus, the accuracy was presented in figures 2 and 3 for the three class, binary and all 37 class cases respectively. Then, the precision, recall and f-measure can be found in figures 2 and 3. For these three metrics the weighted average was used to obtain the results. For Naïve Bayes the precision and f-measure could not be extracted using Weka for unknown reason.

To summarize, the figures show that Random Forest classifier has the higher capacities to distinguish between the occurrence of each class, or its absence. SVM tends to predict only No Events and Attacks but does not really succeed in distinguishing between them. Naïve Bayes fails to make true predictions; it considers everything of class natural. Finally, MLP, it succeeds in determining the class No Events, but does not distinguish over classes almost at all, despite the high accuracy (it is due because of the huge number of samples of class No Events).

The two methods of plotting the ROC curve does not give exactly the same results, however the conclusions remain the same in both cases. This fact may be due to a slightly different set of test data used for its creation.

Given this analysis, it can be deducted that Random Forest algorithm acts the best, and that is why it will be adapted in next sections, in which, at first, an analysis of features and their

importance will be made. However, a deeper look at the amelioration of will be also made, in order to succeed in differentiation between the classes.

4. CONCLUSION

This paper was a presentation of two machine learning toolkits, four different machine learning techniques and four metrics for the evaluations of those techniques. The values of those metrics were presented for all the techniques using the two toolkits, then compared between themselves and with the results provided by [11]. This permitted to determine the most accurate machine learning technique (Random Forest) which will be used in the next section. The next step in this master paper is the analysis of the importance of features, which will help in further modification of the random forest classifier.

This work may be continued by implementing the power system using one of the mentioned toolkits in order to try to further enhance the results. In addition to that, instead of using scikitlearn, Keras or PyTorch packages may be used in order to create a complex neural network and possibly get better results.

REFERENCES

[1] A. Karpatne, W. Watkins, J. Read, and V. Kumar, "Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling," arXiv:1710.11431 [physics, stat], Feb. 2018. http://arxiv.org/abs/1710.11431.

[2] E. M. Ferragut, J. Laska, M. M. Olama, and O. Ozmen, "Real-Time Cyber-Physical False Data Attack Detection in Smart Grids Using Neural Networks," in 2017 International Conference on Computational Science and Computational Intelligence (CSCI), (Las Vegas, NV, USA), pp. 1–6, IEEE, Dec. 2017. https://ieeexplore.ieee.org/document/8560712/.

[3] R. Swischuk, L. Mainini, B. Peherstorfer, and K. Willcox, "Projection-based model reduction: Formulations for physics-based machine learning," Computers & Fluids, vol. 179,

[4] A. S. Zamzam and N. D. Sidiropoulos, "Physics-Aware Neural Networks for Distribution System State Estimation," arXiv:1903.09669 [cs, math], July 2019. http://arxiv.org/abs/1903.0 9669.

[5] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar, "Theory-Guided Data Science: A New Paradigm for Scientific Dis- covery from Data," IEEE Transactions on Knowledge and Data Engineering, vol. 29, pp. 2318– 2331, Oct. 2017. http://ieeexplore.ieee.org/document/7959606/.

[6] M. Hipsey, L. Bruce, and D. Hamilton, "GLM - General Lake Model: Model overview and user information," The University of Western Australia, Perth, Australia, 2014.

[7] T. Morris, "Industrial Control System (ICS) Cyber Attack Datasets - Tommy Morris." https: //sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets.

[8] U. Adhikari, S. Pan, and T. Morris, "Power System Attack Datasets," Apr. 2014.

[9] "Three-phase electric power," Wikipedia, May 2020. https://en.wikipedia.org/w/index.p hp?title=Three-phase_electric_power&oldid=955675515.

[10] J.L. Kirtley Jr., "Introduction To Symmetrical Components," 6.061 Introduction to Power Systems, vol. Class Notes Chapter 4. https://ocw.mit.edu/courses/electrical-enginee ring-and-computer-science/6-061-introduction-to-electric-power-systems-sprin g-2011/readings/MIT6_061S11_ch4.pdf.

[11] R. C. Borges Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in 2014 7th International Symposium on Resilient Control Systems (ISRCS), (Denver, CO, USA), pp. 1–8, IEEE, Aug.

[12] Appendix B - The WEKA workbench," in Data Mining (Fourth Edition) (I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, eds.), pp. 553–571, Morgan Kaufmann, fourth edition ed., 2017. http://www.sciencedirect.com/science/article/pii/B9780128042915000246.

[13] 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, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[14] RandomForest."https://weka.sourceforge.io/doc.dev/weka/classifiers/trees/RandomForest.html.

[15] V. Patel, "Implementing a Multi-Layer Perceptron Neural Network in Python." https://towa rdsdatascience.com/implementing-a-multi-layer-perceptron-neural-network-in-python-b22b5a3bdfa3, May 2020.

[16] B. Shmueli, "Multi-Class Metrics Made Simple, Part II: The F1-score." https://towardsdat ascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1, June 2020.

[17] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the pre- dictions of any classifier," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pp. 1135–1144, 2016.

