Fine-Tuning Naïve Bayes for Imbalanced Datasets

Fahad S. Alenazi*, Khalil El Hindi*, Basil AsSadhan†
*Department of Computer Science, King Saud University, Riyadh, Saudi Arabia
†Department of Electrical Engineering, King Saud University, Riyadh, Saudi Arabia
e-mail: fahadsayer@gmail.com, khindi@ksu.edu.sa, bsadhan@ksu.edu.sa

Abstract—*In this work, we address the problem of the fine-tuning the Naïve Bayesian (NB) classifiers for imbalanced datasets. In particular, we propose a more suitable way to calculate the update step size for this type of domains. The new algorithm is different from the original FTNB algorithm in the way we calculate the probability-term update step. The new update step size is based on the harmonic average of the probability terms. We use the harmonic average of the probability terms to determine if the poor performance of the NB classifier is due to scarcity of data and to identify the probability terms to modify the most. We compare the performance of the proposed algorithm with NB and the original FTNB using two imbalanced datasets in the domain of cyber-security, namely, intrusion detection in Wireless Sensor Networks (WSNs) and ransomware attacks. Our empirical results reveal that the new algorithm significantly outperforms NB and the original FTNB for this type of problem.*

Keywords: Imbalanced datasets, Intrusion detection, Wireless Sensor Networks (WSNs), Ransomware attack classification, Machine learning.

1. Introduction

The Naïve Bayes (NB) learning algorithm is an efficient algorithm for training and classification. It is also robust to noise[1], and incremental. These properties make NB to be one of the top 10 algorithms in the data mining and machine learning community [2]. These properties also make NB suitable for intrusion detection in computer networks, where we need efficient training and detection mechanisms. Also, being an incremental algorithm is an extra advantage, because it allows us to use new training data without needing to retrain the classifier from scratch.

Given a query instance of the form $< a_1, a_2, \cdots, a_m >$, where $, a_i$ is the ith attribute value, the algorithm uses Eq. 1 to find the class with the highest probability given the vector of attribute values.

$$class = \underset{c \in C}{argmax}\ p(c) \cdot \prod_j p(a_j|c)\,, \qquad (1)$$

where,

- c is a vector of all class attribute values.
- $p(c)$ is the probability of class c.

Eq. 1 is simple because NB naively assumes that the attribute values are conditionally independent given the class value. This assumption is made to make better estimation of terms probabilities due to limited training data. Since in practive, most combinations of attribute values either not represented in the training data or not present in sufficient number.. However, NB performance degrades in domains where the independence assumption is not satisfied [3],[4].

Clearly, the performance of NB depends, also, on using the accurate estimation of the probability terms $p(c)$ and $p(a_j|c)$, which is difficult in domains where the training data is scarce [5], [6]. This work addresses the second problem. We modify the Fine Tuning Naïve Bayes (FTNB) learning algorithm to make it more suitable for imbalanced training data. The FTNB [7], [8] algorithm makes gradual changes (updates) to misclassified instance's terms by iteratively computing an update step size for each term and then adding it to the previous term's value. Our modification to FTNB in particular, is modifying the probability update steps to make them more suitable for domains with scarce and imbalanced training data.

The remainder of this paper is organized as follows. In Section 2, we review related work. In Section 3, we propose our FTNB-ID algorithm. In Section 4, we describe the experimental setup and results in details. In Section 5, we give our conclusions and suggestions for future research.

2. Background and Related work

The attempts to improve the classification accuracy of NB can be categorized into two main groups. The first focuses on alleviating the conditional independence assumption [9]–[17], while the second tackles the problem of the lack of training data [7], [8].

Bayesian networks (BN) [18] eliminate the naïve assumption of conditional independence, but finding the optimal BN is NP-hard [9], [19]. Therefore, approximate methods that restrict the structure of the network [15], [16] were proposed to make it more tractable. Other methods attempt to ease the independence assumption by using filtering and feature selections. The expectation here is that the independence assumption is more likely to be satisfied by a

small subset of features or training instances than by the entire set of training features and instances. Evolutional naïve Bayes (ENB) [20] uses a genetic algorithm to select a set of features that improves the classification accuracy of NB and then builds an NB classifier using the selected features.

In the second group, methods were proposed to improve the estimation of the values of probability terms when there is not enough training data. In [13],[14] instance cloning methods were used to deal with data scarcity. These methods are lazy because they build the NB classifier during classification. Therefore, the classification time is relatively high [21]. In [13] a method called LNB (for Lazy Naïve Bayesian) clones some instances based on their dissimilarity to a new instance, while [14] uses a greedy search algorithm to determine the instances to clone. Although, the cloning techniques in [13] and [14] were developed for improving the ranking performance of NB, their classification performance significantly outperforms NB in the same way [14]. The Discriminatively weighted Naïve Bayes (DWNB) [12] method assigns instances different weights depending on how difficult they are to classify. It begins by assigning every instance a weight of one. Then it iteratively increases the weight of instances such that difficult instances get larger weights. Thus, instances with large weights have more effect on the estimation of $P(c)$ and $P(a_j|c)$ than instance with small weights.

FTNB [7] and Selectively Fine-Tuning Bayesian Network (SFTBN) [10] were also proposed to address the problem of scarcity of data for NB classification. In this paper, we propose yet another fine-tuning method. The FTNB algorithm consists of two stages; in the first stage, it builds an initial NB classifier, and in the second stage it uses the misclassified training instances to update the probability terms. If a training instance, inst, of the form $< a_1, a_2, \cdots, a_m, c_{actual} >$, is misclassified then the predicted class, $c_{predicted}$, has higher probability than the actual class, c_{actual}, given the instance's other attribute values. During the fine-tuning stage, the probability terms are updated in such a way that $p(c_{predicted}|\ a_1, a_2, \cdots, a_m)$ is decreased, and $p(c_{actual}|\ a_1, a_2, \cdots, a_m)$ is increased. The tuning process continues as long as the classification accuracy improves. Figure 1 shows the details of the algorithm.

In Eqs. 4 and 5, FTNB determines the amount to update $p(a_i|c_{actual})$ and $p(a_i|c_{predicted})$, respectively.

$$\delta_{t+1}(a_i, c_{actual}) = \eta \cdot (\alpha \cdot p(max_i|c_{actual}) - p(a_i|c_{actual})) \cdot error \quad (3)$$

$$\delta_{t+1}(a_i, c_{predicted}) = -\eta \cdot (\alpha \cdot p(a_i|c_{predicted}) - p(min_i|c_{predicted})) \cdot error \quad (4)$$

The amount of update $\delta_{t+1}(a_i, c_{actual})$ and $\delta_{t+1}(a_i, c_{predicted})$ are proportional to the error, which is computed as:

$$error = |P(c_{predicted}|\ a_1, a_2, \cdots, a_m) - P(c_{actual}|\ a_1, a_2, \cdots, a_m)|, \quad (5)$$

where

$$P(c_o|\ a_1, a_2, \cdots, a_m) = \frac{p(c_o|\ a_1, a_2, \cdots, a_m)}{\sum_k^m p(c_k|\ a_1, a_2, \cdots, a_m)} \quad (6)$$

Eq. 6 is used to normalize to the probabilities.

Algorithm FTNB-ID (Training_instances)

phase 1

Use Training_instances to estimate the values of each probability term used by the NB algorithm

phase 2

t = 0

while training classification accuracy improves do

for each training instance, inst, do

let c_{actual} be the actual class of inst

let $c_{predicted}$ = classify(inst)

if $c_{predicted}$ <> c_{actual} //misclassified

for each attribute value, a_i, of inst do

compute $\delta_{t+1}(a_i, c_{actual})$

compute $\delta_{t+1}(c_{actual})$

let $p_{t+1}(a_i|\ c_{actual}) = p_t(a_i|\ c_{actual}) + \delta_{t+1}(a_i, c_{actual})$

let $p_{t+1}(c_{actual}) = p_t(c_{actual}) + \delta_{t+1}(c_{actual})$

compute $\delta_{t+1}(a_i, c_{predicted})$

compute $\delta_{t+1}(c_{predicted})$

let $p_{t+1}(a_i|\ c_{predicted}) = p_t(a_i|\ c_{predicted}) - \delta_{t+1}(a_i, c_{predicted})$

let $p_{t+1}(c_{predicted}) = p_t(c_{predicted}) - \delta_{t+1}(c_{predicted})$

endfor

endif

endfor

let $t = t + 1$

end while

Figure 1. The FTNB algorithm

The learning rate, η, which is a value between zero and one is used to decrease the update step. The update step size for the probability term $p(a_i|c_{actual})$ in Eq. 4, is designed to be large for small terms and small for large terms. It means that the update step is proportional to $\alpha \cdot p(max_i|c) - p(a_i|c)$, where α is a constant and max_i is the value of the i^{th} attribute with the maximum probability given c_{actual}. On the other hand, the update step size for the probability term $p(a_i|c_{predicted})$ in Eq. 5, is designed to be large for large terms and small for small terms; we use $\alpha \cdot p(a_i|c_{predicted}) - p(min_i|c_{predicted})$, where $\min_i$ is the value of the i^{th}attribute with the minimum probability, given that $c_{predicted}$. Similarly, α is a constant greater than or equal to one and used to control the update step of $p(max_i|c)$ and $p(min_i|c_{predicted})$. Setting α to one means these terms get zero as the update step size. Following [7], we set α to two in all our experiments.

According to [7], fine tuning $p(c_{actual})$ and $p(c_{predicted})$ did not improve the classification accuracy of NB. This is probably because they are estimated using many instances compared to the probability terms $p(a_i|c_{predicted})$ and $p(a_i|c_{actual})$.

It turned out the fine-tuning NB increases its variance, and therefore it becomes less tolerant to noise. In [8], a more noise tolerant FTNB is proposed. Increasing the variance of NB, however, makes it more suitable for building ensembles of classifiers using a method such as bagging [22]. In [23] it was shown that bagging an ensemble of FTNB classifiers for text classification is more effective than bagging an ensemble of NB classifiers. In [10], a selective FTNB algorithm was proposed to fine-tune Bayesian Networks. Thus, this work addresses the Naïve Bayesian assumption of NB and the scarcity of training data problem. In [24], the FTNB algorithm was compared and combined with the DWNB algorithm. In [20]and [25], the probability estimation problem was dealt with as an optimization problem and metaheuristic approaches were used to find good solutions for it.

3. FTNB for Imbalanced Datasets (FTNB-ID)

In many applications, the available datasets may contain many instances of one class and a small number of instances from a different class. For example, in data security datasets most instances, represent normal behavior and very few represent an attack. In this work, we propose an FTNB algorithm for Imbalanced Datasets, FTNB-ID. In FTNB-ID, we modify the probability update formulas based on the harmonic average of the probability terms.

When the training data is imbalanced, classifiers tend to predict the most common class correctly. It is the rare class that they tend to misclassify. It is easy to see this in the case of NB classifiers, because the probability terms $p(a_i|c_{actual})$ tends to be large if c_{actual} is a common class and small if c_{actual} is a rare class. Since the harmonic average is dominated by the small value, therefore, the harmonic average $p(a_i|c_{actual})$ and $p(a_i|c_{predicted})$ would be small, if c_{actual} is the rare class. Thus, the size of the update step should be large i.e., we need to substantially increase $p(a_i|c_{actual})$ and decrease $p(a_i|c_{predicted})$ because a small modification would not make much difference. However, if both of $p(a_i|c_{actual})$ and $p(a_i|c_{predicted})$ are large the harmonic average would also be large, and the update step size should be small. In FTNB-ID, we also use a decaying learning rate so that the size of the update step gets smaller as the training process progresses.

Thus in FTNB-ID, we calculate the amount of update for $p(a_i|c_{actual})$ and $p(a_i|c_{predicted})$ using Eqs. 7 and 8 instead of Eqs. 3 and 4, respectively.

$$\delta_{t+1}(a_i, c_{actual}) = \frac{\eta}{t^2} \cdot \left(1 - 2 \cdot \left(\frac{1}{p(a_i|c_{actual})} + \frac{1}{p(a_i|c_{predicted})}\right)\right) \quad (7)$$

$$\delta_{t+1}(a_i, c_{predicted}) = \frac{\eta}{t^2} \cdot \left(1 - 2 \cdot \left(\frac{1}{p(a_i|c_{actual})} + \frac{1}{p(a_i|c_{predicted})}\right)\right), \quad (8)$$

where *t* is the iteration (epochs) number.

Contrary to what was reported in [7], we found out that it is useful to fine-tune $p(c_{actual})$ and $p(c_{predicted})$. This is probably the case because we are dealing with imbalanced training data. To modify class probability, we apply Eqs 9 and 10 to find the step size that will be used in increasing $p(c_{actual})$ and decreasing $p(c_{predicted})$, respectively.

$$\delta_{t+1}(c_{actual}) = \frac{\eta}{t^2} \cdot \left(1 - 2 \cdot \left(\frac{1}{p(c_{actual})} + \frac{1}{p(c_{predicted})}\right)\right) \quad (9)$$

$$\delta_{t+1}\left(c_{predicted}\right) = \frac{\eta}{t^2} \cdot \left(1 - 2 \cdot \left(\frac{1}{p(c_{actual})} + \frac{1}{p(c_{predicted})}\right)\right) \quad (10)$$

4. Experimental Setup and Results

In this section, we validate the classification performance of the proposed FTNB-ID and compare it with the NB and FTNB algorithms. We evaluate the proposed FTNB-ID using two recently published datasets in the domain of cyber-security. The first dataset is dataset for intrusion detection in Wireless Sensor Networks (WSN) networks (named as WSN-DS) [26], [27], [28]. This dataset has been evaluated and

published recently and contains (374661 records and 19 numeric features) that represent four types of DoS attacks: Blackhole, Grayhole, Flooding, and Scheduling (TDMA) attack, in addition to the normal behavior (no-attack) records. WSN-DS is obviously an imbalanced dataset. Table (1) shows the class distribution.

Classes	**# instance**	**Percentage**
Normal	340066	90.77 %
Flooding	3312	0.88 %
Grayhole	14596	3.90 %
Blackhole	10049	2.68 %
TDMA	6638	1.77 %
Total	374661	100 %

Table (1) WSD data description

Class Name	# instance	Percentage
Goodware	942	61.80%
Citroni	50	3.30%
CryptLocker	107	7.00%
CryptoWall	46	3.00%
Kollah	25	1.60%
Kovter	64	4.20%
Locker	97	6.40%
Matsnu	59	3.90%
Pgpcoder	4	0.30%
Reveton	90	5.90%
TeslaCrypt	6	0.40%
Trojan-Ransom	34	2.20%

Table (2) Ransomware data description

The second dataset was created and verified in [29]. The authors have collected ransomware samples that are representative of the most popular versions and variants currently encountered in the wild (most of them belong to crypto-ransomware type). Then, they manually clustered each ransomware into a family name. The dataset contains (582) ransomware samples, and (942) of benign applications (goodware) and (30,967) features. The dataset is available from VirusShare website [29]; Table (2) shows its class distribution.

We evaluated FTNB-ID with respect to precision, recall, and F1 score which is a more suitable evaluation criterion than accuracy for domains with imbalanced datasets. We used 10-fold cross-validation. Tables (3) and (4) show the results of the NB, FTNB, and the proposed FTNB-ID for WSD and Ransomware datasets, respectively. In order to find the three performance metrics (precision, recall, and F1 score) , a muli-class confusion matrices are build. In a confusion matrix, the predicted classes are compared with the actual classes. Each row of the matrix represents the results of prediction for the corresponding class at that row, while each column represents the actual class. The diagonal cells show the number and percentage of correct classifi cations by the trained classifier, true positive (*TP*), while the off diagonal cells represent the misclassified predictions, false positive (*FP*) and false negative (*FN*).

The results reveal that FTNB-ID consistently and significantly outperforms NB and FTNB for all different classes with respect to the three performance metrics we used. For the WSD dataset, on average, there is 10% improvement in F1 score as a result of using FTNB-ID compared to NB. For some classes that have scarce data the improvement is even more. For example, there is a 25% improvement in the F1 score for TDMA and 11% for Grayhole. Surprisingly, there is a slight improvement for the most common class which is *Normal*.

For the Ransomware dataset, FTNB-ID improved the F1 score by an average of 36% compared to NB. And compared to FTNB, it improved the F1 score by 30% on average.

We conducted a corrected paired two-tailed t-test with 95% confidence to see if FTNB-ID significantly outperforms NB and FTNB and the result was positive in terms of accuracy and F1 score.

Classes	**NB**			**FTNB**			**FTNB-ID**		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
Normal	0.9967	0.9729	0.9847	0.9965	0.9904	0.9934	0.9971	0.9981	**0.9976**
Flooding	0.7888	0.9958	0.8803	0.9252	0.9444	0.9347	0.9576	0.9535	**0.9555**
Grayhole	0.7757	0.9498	0.8539	0.9081	0.9482	0.9277	0.9631	0.9679	**0.9655**
Blackhole	0.9555	0.9259	0.9405	0.9690	0.9523	0.9606	0.9822	0.9739	**0.9781**
TDMA	0.5347	0.9330	0.6798	0.7460	0.9217	0.8246	0.9630	0.9137	**0.9377**
Average	*0.8107*	*0.9555*	*0.8680*	*0.9090*	*0.9514*	*0.9282*	*0.9614*	*0.9726*	***0.9669***

Table (3) The result (Precision, Recall, and F1 score) for WSD dataset

Classes	NB			FTNB			FTNB-ID		
	Precision	Recall	F1 score	Precision	Recall	F1 score	Precision	Recall	F1 score
Goodware	0.6151	0.9671	0.7520	0.6632	0.9533	0.7822	0.9716	0.9437	**0.9575**
Citroni	0.0000	0.0000	0.0000	0.8421	0.3200	0.4638	0.6429	0.7200	**0.6792**
CryptLocker	0.1250	0.0187	0.0325	0.0000	0.0000	0.0000	0.6792	0.6729	**0.6761**
CryptoWall	0.0000	0.0000	0.0000	0.2500	0.0217	0.0400	0.3621	0.4565	**0.4038**
Kollah	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0588	0.0400	**0.0476**
Kovter	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6667	0.7188	**0.6917**
Locker	0.6000	0.0309	0.0588	0.1013	0.0825	0.0909	0.5488	0.4639	**0.5028**
Matsnu	0.5000	0.0508	0.0923	1.0000	0.0847	0.1563	0.4000	0.6102	**0.4832**
Pgpcoder	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**0.0000**
Reveton	0.0000	0.0000	0.0000	0.5556	0.1667	0.2564	0.7473	0.7556	**0.7514**
TeslaCrypt	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	**0.0000**
Trojan-Ransom	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0769	0.0882	**0.0822**
Average	*0.1533*	*0.0890*	*0.0780*	*0.2843*	*0.1357*	*0.1491*	*0.4295*	*0.4558*	***0.4396***

Table (4) The result (Precision, Recall, and F1 score) for Ransomware dataset

5. Conclusions

This work proposed a fine-tuning algorithm for NB that is suitable for imbalanced datasets. The proposed FTNB-ID algorithm determines the size of the update step based on the harmonic average of the probability terms it fine tunes. This makes the update step size large when rare classes are incorrectly classified. We evaluated the performance of the algorithm with respect to precision, recall, and F1 score. Our empirical analysis reveals that FTNB-ID significantly outperforms NB and the original FTNB algorithm with respect to the F1 score. In Fact, FTNB-ID also significantly outperforms NB and FTNB with respect to the classification accuracy. As future work, we intend to investigate using the harmonic average for instance weighing and features weighting in Bayesian classification.

6. Acknowledgement

This work was supported in part by the Gulf Science, Innovation and Knowledge Economy Programme of the U.K. Government under UK-Gulf Institutional Link Grant IL 279339985.

7. References

[1] D. F. Nettleton, A. Orriols-Puig, and A. Fornells, "A study of the effect of different types of noise on the precision of supervised learning techniques," *Artif. Intell. Rev.*, vol. 33, no. 4, pp. 275–306, Apr. 2010.

[2] X. Wu *et al.*, "Top 10 algorithms in data mining," *Knowl. Inf. Syst.*, vol. 14, no. 1, pp. 1–37, Jan. 2008.

[3] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian Network Classifiers," *Mach. Learn.*, vol. 29, no. 2, pp. 131–163, Nov. 1997.

[4] M. A. Palacios-Alonso, C. A. Brizuela, and L. E. Sucar, "Evolutionary Learning of Dynamic Naïve Bayesian Classifiers," *J. Autom. Reason.*, vol. 45, no. 1, pp. 21–37, Jun. 2010.

[5] E. Frank, M. Hall, and B. Pfahringer, "Locally Weighted Naïve Bayes," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, pp. 249–256, 2003.

[6] U. M. Fayyad and K. B. Irani, "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning," in *IJCAI*, 1993.

[7] K. El Hindi, "Fine tuning the Naïve Bayesian learning algorithm," *AI Commun.*, no. 2, pp. 133–141, 2014.

[8] K. El Hindi, "A noise tolerant fine tuning algorithm for the Naïve Bayesian learning

algorithm," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 26, no. 2, pp. 237–246, Jul. 2014.
[9] G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks," *Artif. Intell.*, vol. 42, no. 2–3, pp. 393–405, Mar. 1990.
[10] A. Alhussan and K. El Hindi, "Selectively Fine-Tuning Bayesian Network Learning Algorithm," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 30, no. 08, p. 1651005, Mar. 2016.
[11] L. Jiang, Z. Cai, H. Zhang, and D. Wang, "Naïve Bayes text classifiers: a locally weighted learning approach," *J. Exp. Theor. Artif. Intell.*, vol. 25, no. 2, pp. 273–286, Jun. 2013.
[12] L. Jiang, D. Wang, and Z. Cai, "Discriminatively weighted naïve bayes and its application in text classification," *Int. J. Artif. Intell. Tools*, vol. 21, no. 01, p. 1250007, Feb. 2012.
[13] L. Jiang and Y. Guo, "Learning lazy naïve Bayesian classifiers for ranking," in *17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05)*, pp. 5– 416, 2005.
[14] L. Jiang and H. Zhang, "Learning instance greedily cloning naïve Bayes for ranking," in *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pp. 8 , 2005.
[15] L. Jiang, H. Zhang, and Z. Cai, "A Novel Bayes Model: Hidden Naïve Bayes," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 10, pp. 1361–1371, Oct. 2009.
[16] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, pp. 462–467, May 1968.
[17] L. Jiang, Z. Cai, D. Wang, and H. Zhang, "Improving Tree Augmented Naïve Bayes for Class Probability Estimation," *Know-Based Syst*, vol. 26, pp. 239–245, Feb. 2012.
[18] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Publishers Inc., San Francisco, CA, USA: Morgan Kaufmann, 1988.
[19] D. M. Chickering, "Learning Bayesian Networks is NP-Complete," in *Learning from Data*, vol. 112, D. Fisher and H.-J. Lenz, Eds. New York, NY: Springer New York, pp. 121–130, 1996.
[20] L. Jiang, H. Zhang, Z. Cai, and J. Su, "Evolutional naïve bayes," in *Proceedings of the 2005 International Symposium on Intelligent Computation and its Application, ISICA*, pp. 344–350, 2005.
[21] L. Jiang, D. Wang, Z. Cai, and X. Yan, "Survey of Improving Naïve Bayes for Classification," in *Advanced Data Mining and Applications*, pp. 134–145, 2007.
[22] L. Breiman, "Bagging Predictors," *Mach Learn*, vol. 24, no. 2, pp. 123–140, Aug. 1996.
[23] K. El Hindi, H. AlSalman, S. Qasem, and S. Al Ahmadi, "Building an Ensemble of Fine-Tuned Naïve Bayesian Classifiers for Text Classification," *Entropy*, vol. 20, no. 11, p. 857, Nov. 2018.
[24] K. El Hindi, "Combining Instance Weighting and Fine Tuning for Training Naïve Bayesian Classifiers with Scant data," *INTERNATIONAL ARAB JOURNAL OF INFORMATION TECHNOLOGY 15 (6), 1099-1106,* 2016.
[25] D. M. Diab and K. M. El Hindi, "Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification," *Appl. Soft Comput.*, vol. 54, pp. 183–199, May 2017.
[26] "WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks." Journal of Sensors, Article ID 4731953, 16 pages, 2016.
[27] K. Kumar, R. Jha, and S. Afroz, "Data Mining Techniques for Intrusion Detection: A Review," vol. 3, no. 6, p. 5, 2014.
[28] M. Alenezi, S. Arabia, and Q. T. Obeidat, "Fault-Proneness of Open Source Systems: An Empirical Analysis," *International Arab Conference on Information Technology (ACIT2014), 164-169,* 2014.
[29] D. Sgandurra, L. Muñoz-González, R. Mohsen, and E. C. Lupu, "Automated Dynamic Analysis of Ransomware: Benefits, Limitations and use for Detection," *arXiv:1609.03020 [cs.CR]*, Sep. 2016.