Article

Complement Class Fine-Tuning of Naïve Bayes for Severely Imbalanced Datasets

Fahad S. Alenazi [1,*], Khalil El Hindi [1] and Basil AsSadhan [2]

[1] Department of Computer Science, King Saud University, Riyadh, Saudi Arabia; khindi@ksu.edu.sa
[2] Department of Electrical Engineering, King Saud University, Riyadh, Saudi Arabia; bsadhan@ksu.edu.sa
* Correspondence: fahadsayer@gmail.com

Abstract: In this work, we adapt the fine-tuning algorithm of the Naïve Bayesian (FTNB) classifier to make it more suitable for imbalanced datasets. In particular, we boost misclassified instance probability terms by an amount that is disproportional to the harmonic mean of actual and predicted classes. The intuition is that discriminative attributes when the instance is misclassified would have small probability term pair values in both the actual class due to data scarcity and the predicted class due to weak correlation. Conversely, if both values are relatively high, then the attribute has good data coverage (support) and it should not be a cause for misclassification. Since the harmonic average is dominated by the smaller value and we have an imbalanced dataset, we should enact a large update if both or either term probabilities of actual and predicted classes are small. We used several benchmark datasets (60 different balanced and imbalanced datasets) to determine if the poor performance of the NB classifier is due to the scarcity of data and compared the performance of the proposed algorithm with NB, original FTNB, and other relatively new SOTA Ensemble Imbalanced Classifiers. Our empirical results reveal that the new proposed algorithm significantly outperforms all other classifiers.

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 supports incremental learning. These properties make NB suitable for intrusion detection/prevention systems to detect new unknown "zero day" attack signatures. Incremental learning allows the classifier to amend new malicious training instances only to refine and improve the prediction efficiently without training the classifier on the whole dataset again [2, 3, 4]. Given a query instance of the form $< a_1, a_2, \cdots, a_m >$, where a_j is the j^{th} attribute value, the algorithm uses Eq. 1 to find the class with the highest probability given the vector of attribute values:

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

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 for efficiency reasons and to make it possible to estimate the values of all probability terms when the training data are limited, since in practice, many attribute values are not represented in the training data in sufficient numbers. However, the performance of NB degrades in domains where the independence assumption is not satisfied [5, 6]. Clearly, the performance of NB also depends on

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

using the accurate estimation of the probability terms $p(c)$ and $p(a_j|c)$, which is difficult in domains where the training data are scarce [7, 8].

This work addresses the second problem of scarce training data for imbalanced datasets. We modified the FTNB learning algorithm to make it more suitable for imbalanced training data. The FTNB algorithm [9, 10] makes gradual changes (updates) to the probability terms involved in classifying any training instance, 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 it more suitable for domains with scarce and imbalanced training data. The fine-tuning stage extends the multinomial NB to make gradual updates to the misclassified instances' probability terms.

Most existing approaches for class-imbalance problems apply class decomposition to divide a multi-class problem into multiple two-class problems, then conquer each two-class-imbalance sub-problem [11, 12]. In the past two decades, many solutions to the two-class-imbalance problem have been proposed. SMOTE [13] is the most influential technique for class-imbalance problems [14], which generates synthetic rare class samples based on the sample of k nearest neighbors with same class. However, SMOTE and its variants have the following two drawbacks in synthetic sample generation [15]:

1. Rare classes' probability distributions are not considered;
2. The generated minority class samples lack diversity and overlap heavily with major classes for sever imbalanced datasets.

Many recent published works addressed these drawbacks. Mathew et al. [15] proposed a weighted kernel-based SMOTE, which generates synthetic rare class samples in feature space. The authors in [16] proposed a SMOTE-based, class-specific extreme learning machine which exploits the benefits of both the minority oversampling and class-specific regularization to overcome the limitation of the linear interpolation of SMOTE.

In [17] , a generalized Dirichlet distribution was used as a prior for the multinomial NB classifier to find non-informative generalized Dirichlet priors for a multinomial Naïve Bayesian classifier so that its performance on high-dimensional imbalanced data could be largely improved compared to generating synthetic instances in a high-dimensional space.

In our proposed method, FTNB-ID, we have the rare class prior distribution incorporated, which tackles the first drawback, and we granularly and gradually update for the probability terms, which can eliminate the diversity and overlap drawback, since the update weight is disproportional to each probability term compared to its complement class term. Thus, the fine-tuning process will create a more granular and dynamic distribution for each rare class attribute. Nevertheless, FTNB-ID can be integrated with any data-level approaches for a class imbalance problem such as SMOTE.

In addition, NB in general is one of the fastest machine learning algorithms [1] and requires a few more iterations (maximum of 10 epochs) for the fine-tuning stage. FTNB-ID can apply online learning. A new online instance is predicted first, and then if misclassified, only its probability terms will be updated. This is not the case for other machine learning models, which need to be retrained again from scratch. Similarly, data-level approaches such as SMOTE oversampling will generate synthetic data that lack diversity and overlap with major classes, since they clone instances that are similar to the k nearest neighbors of the predicted instance and do not provide granularity updates for each instance attribute.

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 detail. In Section 5, we give our conclusions and suggestions for future research.

2. Background and Related work

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

The attempts to improve the classification accuracy of NB can be categorized into two main groups. The first focused on alleviating the conditional independence assumption, and the second tackled the problem of a lack of training data.

Bayesian networks (BN) [18] eliminate the naïve assumption of conditional independence; however, finding the optimal BN is NP-hard [19, 20]. Therefore, approximate methods that restrict the structure of the network [21, 22, 23, 24] have been proposed to make it more tractable. Other methods attempt to ease the independence assumption by selecting the relevant features [25, 26]. 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) [27], for instance, 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.

The methods of the second group were proposed to improve the estimation of the values of probability terms when there are not enough training data. In [28, 29], 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 [30]. The authors of [28] used a method called LNB (for Lazy Naïve Bayesian) to clone some instances based on their dissimilarity to a new instance, whereas the authors of [29] used a greedy search algorithm to determine the instances to clone. Although the cloning techniques in [28] and [29] were developed for improving the ranking performance of NB, their classification performance is significantly greater than NB's in the same way [29]. The Discriminatively Weighted Naïve Bayes (DWNB) [31] 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 weights of instances such that difficult instances get larger weights. Instances with large weights have a greater effect on the estimation of $P(c)$ and $P(a_j|c)$ than instances with small weights.

FTNB [9] and the selectively fine-tuning Bayesian network (SFTBN) [32] were also proposed to address the problem of scarcity of data for NB classification. 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 as in class $c_{predicted}$, instead of the true class c_{actual}, then the predicted class, $c_{predicted}$ has a higher probability than the actual class, c_{actual}, given the instance's other attribute values $< a_1, a_2, \cdots, a_m >$. During the fine-tuning stage, FTNB updates the relevant probability terms 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 fine-tuning process continues as long as the classification accuracy improves. However, the fine-tuning makes NB less tolerant to noise, therefor, a more noise tolerant FTNB was proposed in [10].

Increasing the variance of NB, however, makes it more suitable for building ensembles of classifiers using a method such as bagging [33]. In [34], the FTNB algorithm was compared and combined with the DWNB algorithm. In [19] and [35], the probability estimation problem was modeled as an optimization problem and metaheuristic approaches were used to find better probability estimation. In [36], the author proposed Class Imbalance Learning with Bayesian optimization (CILBO) to find the best hyper-parameter combination of the model, and it differs from commonly used hyper-parameter optimization by addressing the issue of class imbalance. In this paper, which is an extension of our previous work [37], we further investigate the following:

- We modified the original FTNB early termination condition to be based on the training F1 measure instead of accuracy and similar to the proposed FTNB-ID;

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

- We investigated different training size thresholds and applied our experiments on a relatively new benchmark dataset;
- Finally, we tested our proposed method on 57 datasets from UCI to evaluate the performance of FTNB-ID on both balanced and imbalanced datasets.

3. FTNB for Imbalanced Datasets (FTNB-ID)

In many applications, the available datasets are not balanced, as many instances belong to a single class and a small number of instances belong to the remaining classes. For example, in cybersecurity datasets, most instances represent benign behavior and very few represent an attack. In this work, we propose an FTNB algorithm to re-weighting imbalanced datasets attributes and classes probability terms by considering the rare classes prior distributions, and granularly and gradually update the classifier's probability terms to eliminate the diversity and overlap drawback. We tackle these two drawbacks in most synthetic sample generation such as SMOTE and its variants [15] since the update weight is disproportional to each probability term compared to its complement class term. Thus, the fine-tuning process will create a more granular and dynamic distribution for each rare class attribute. Nevertheless, FTNB-ID can be integrated with any data-level approaches for a class imbalance problem such as SMOTE.

When the training data is severely imbalanced, classifiers tend to predict the most common class correctly and the rare classes tend to be misclassified. In addition, the probability term values of a discriminative attribute of misclassified instance would be derived from either: 1) a small pair of actual and predicted classes probability term values, or 2) a small probability term of rare actual class and larger probability term of predicted class due to rare actual class instances, and weak association with the predicted major class.

In this case, we can safely augment the data for this probability term of misclassified instance. Thus, the size of the update step should be large; i.e., we need to substantially increase $p(a_i|c_{actual})$ and decrease $p(a_i|c_{predicted})$. In contrast, if both $p(a_i|c_{actual})$ and $p(a_i|c_{predicted})$ probability terms are relatively large, this implies there are already sufficient data (support) and this attribute is not discriminative and shouldn't be the cause for the misclassification. Thus, the update step size should be minimal.

In order to apply these disproportional probability term updates, we utilize the harmonic average, since it is dominated by the smaller values. Precisely, the complement harmonic average (1- harmonic average) would be small and the update size would be small if both the $p(a_i|c_{actual})$ and $p(a_i|c_{predicted})$ were to be large. Similarly, in the case of scarce or skewed data, the complement harmonic average would be large and the update size would be large if both or one of $p(a_i|c_{actual})$ and $p(a_i|c_{predicted})$ were to be small. Thus, in FTNB-ID, we calculate the update weights for $p(a_i|c_{actual})$ and $p(a_i|c_{predicted})$ using (2), (3) and (4), respectively.

$$W_i = \frac{\eta}{t^2} \cdot \left(1 - 2 / \left(\frac{1}{p_t(a_i|\, c_{actual})} + \frac{1}{p_t(a_i|\, c_{predicted})} \right) \right) \tag{2}$$

$$p_{t+1}(a_i|\, c_{actual}) = p_t(a_i|\, c_{actual}) + W_i \tag{3}$$

$$p_{t+1}(a_i|\, c_{predicted}) = p_t(a_i|\, c_{predicted}) - W_i \tag{4}$$

Here, (η) is a learning rate between zero and one, and (t) is the iteration (epochs) number used to decrease the update step in the fine-tuning process for each misclassified instance's probability terms.

Contrary to what was reported in [9], we found out that it is useful to update the priors for the misclassified instances as well in the fine-tuning process. This was probably

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

the case because we were dealing with imbalanced training data. To modify class probability $p(c_{actual})$ and $p(c_{predicted})$ for misclassified instances, we apply (5), (6) and (7), respectively. This will increase $p(c_{actual})$ and decrease $p(c_{predicted})$ proportionaly to the related attribute probability term updates frequency. Precisely, we are updating these priors every time we are updating attributes' probability terms for misclassified instance. Thus, and since we modify the probability terms, one can think of them as granular weights of the attributes and classes.

$$W_j = \frac{\eta}{t^2} \cdot \left(1 - 2 / \left(\frac{1}{p_t(c_{actual})} + \frac{1}{p_t(c_{predicted})} \right) \right) \quad (5)$$

$$p_{t+1}(c_{actual}) = p_t(c_{actual}) + W_j \quad (6)$$

$$p_{t+1}(c_{predicted}) = p_t(c_{predicted}) - W_j \quad (7)$$

Algorithm 1. FTNB-ID.

Input : a set of training instances D and the maximum number of iterations T
Output : Fine-Tuned Naïve Bayes

(1) Build initial naïve Bayes using D
(2) While training **F-score** improves and t < T do
  a. For each training instance, inst, from D do
    i. classify(inst)
    ii. if $c_{predicted}$ <> c_{actual} //misclassified
    iii. for each attribute value, a_i, of inst. Do
      1. $p_{t+1}(a_i | c_{actual}) = p_t(a_i | c_{actual}) + W_i$
      2. $p_{t+1}(a_i | c_{predicted}) = p_t(a_i | c_{predicted}) - W_i$
    iv. $p_{t+1}(c_{actual}) = p_t(c_{actual}) + W_j$
    v. $p_{t+1}(c_{predicted}) = p_t(c_{predicted}) - W_j$
    vi. Update the naïve Bayes structure with the current weights
  b. Let $t = t + 1$
(3) Final trained naïve Bayes

Figure 1: The FTNB-ID algorithm

4. Experimental Setup and Results

In this section, we evaluate the classification performance of FTNB-ID and compare it with the NB and FTNB algorithms and other relatively recent non-Bayesian classifiers developed to tackle the class imbalanced dataset problem. In addition, we modified the original early termination condition of FTNB to be based on F1 score, like FTNB-ID, instead of accuracy, and we experimented with different fine-tuning training thresholds for both FTNB and FTNB-ID. The proposed fine-tuning algorithm was implemented in Java by extending the Weka source code of (Multinomial Naïve Bayes) [38]. All continuous attributes were discretized using Fayyad et al.'s [9] supervised discretization method as implemented in Weka [38], and missing values were simply ignored.

4.1 Datasets

The first cybersecurity dataset [39] was created in 2017 by the Canadian Institute for Cybersecurity (CIC) to be used as a benchmark dataset to evaluate intrusion detection systems. The CIC-IDS'17 dataset contains both raw and aggregated netflow data generated by two different networks. The first network is the Attack-Network for launching most up-to-date common attacks, and the second network is a Victim-Network of an agent performing benign behaviors. The dataset contains five categorical features (source and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

destination IPs and ports, protocol and timestamp), 78 continuous features (flow statistical analysis) and a label class which represents benign data and 14 different attacks.

The second dataset was created and verified in [40]. The authors 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 11 different family names. The dataset contains (582) ransomware instances, (942) benign records and 30,967 features.

The third dataset is for intrusion detection in wireless sensor networks (WSNs) (named as WSN-DS) [41]. This dataset has been evaluated and published recently and contains (374,661 records and 19 numeric features) that represent four types of DoS attacks, blackhole, grayhole, flooding and scheduling (TDMA) attacks, in addition to the benign behavior (normal) records. WSN-DS is an imbalanced dataset. Table 1 shows a brief description of the three datasets, and Tables (2-4) shows Type of attack classes distributions with the results per class.

Table 1: Imbalanced datasets summary

Dataset	# instances	# features	Type of attack	Normal Class (%)
CIC-IDS 2017 *[39]*	2,830,743	83	14	80.3 %
Ransomware *[40]*	1,524	30,967	11	61.8 %
WSN *[41]*	374,661	19	4	90.8 %

Lastly, we benchmarked our experiments against 57 different datasets obtained from the UCI repository [42]. Missing values were simply ignored. All ordinal attributes were discretized using Fayyad et al.'s [9] supervised discretization method as implemented in WEKA. Dataset details are in Table 7.

4.2. Results

We evaluated the algorithms with respect to accuracy, precision, recall and F1 measure. The F1 measure is a more suitable evaluation criterion than accuracy for domains with imbalanced datasets. We also report accuracy and FP rate. We used 10-fold cross-validation and the t-test to see if the differences are significant.

Multi-class confusion matrices were built for each dataset in order to calculate the performance metrics. In a confusion matrix, the predicted classes are compared with the actual classes. Each row of the matrix represents the results of the prediction for the corresponding class in that row, and each column represents the actual class. The diagonal cells show the number and percentage of correct classifications by the trained classifier, true positives (TPs), and the off-diagonal cells represent the misclassified predictions—false positives (FPs) and false negatives (FNs). The averages of 10 folds for each class and of all classes are reported for each dataset.

4.2.1. Results for Imbalanced datasets

To evaluate the classification performance of proposed method, we conducted a corrected paired two-tailed t-test with 95% confidence for 10-fold cross validation. FTNB-ID significantly outperformed NB and FTNB in all three datasets with respect to accuracy, F1, precision, recall and FP rate. The results reveal that FTNB-ID consistently outperforms NB and FTNB with respect to all performance metrics. In this experiment, we use macro-average (unweighted) for the measures (F-measure, precision, recall, and FP Rate etc.) since it is more sensitive to class-imbalance dataset than (weighted) measures. Thus, major and minor classes with skewed data will contribute equally to the measurement metrics.

Figure 2 (a) shows the results of NB, FTNB and FTNB-ID for CIC-IDS'17, Ransomware and WSN datasets. For CIC-IDS'17, FTNB-ID outperformed NB and FTNB in terms

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

of the F-measure by 39% and 4%, respectively. With this dataset, FTNB-ID had better and more consistent performance for each class (see Table 2).

For the Ransomware dataset, FTNB-ID improved the F1 score by an average of 37% compared to NB and 34% compared to FTNB. In addition, FTNB-ID had better performance for each class compared to NB and six classes compared to FTNB (see Table 3). FTNB was better at handling three classes, but this at the expense of the most common (normal) class, which got its F1 score substantially decreased compared to FTNB-ID.

For the WSN dataset, there was a 7% improvement in the F1 score for FTNB-ID compared to NB, and 14% compared to FTNB. FTNB-ID had better performance for each class (see Table 4). For example, there were 18% and 48% improvements in F1 score for TDMA compared to NB and FTNB, respectively. Moreover, there were even improvements for the most common class (normal) by 1% and 10% compared to NB and FTNB, respectively.

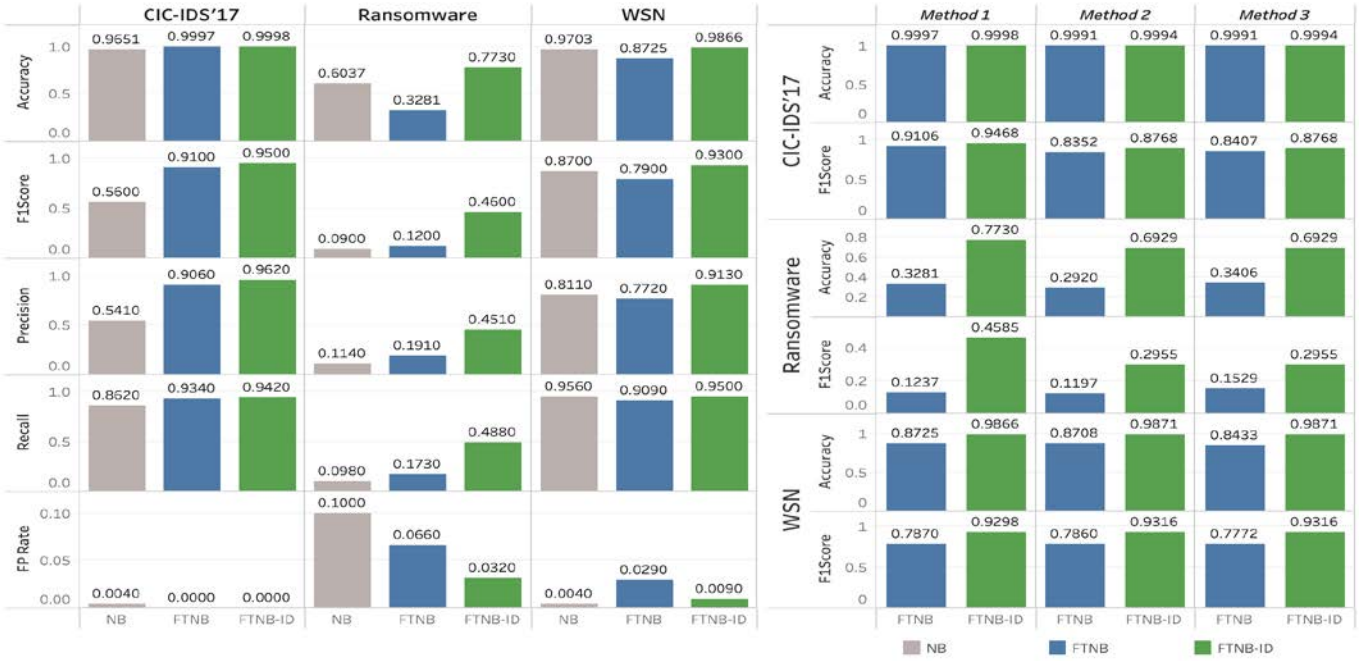

Figure 2: (a) Test results of NB, FTNB, and FTNB-ID (b) Different method results of FTNB, and FTNB-ID

In the next experiment, we changed the early termination condition in the original FTNB to be the F1 score instead of accuracy, as in FTNB-ID. We also tried different fine-tuning dataset threshold sizes. Figure 2 (b) shows the results for the three methods on all three datasets. In the first method, we used the same previous settings without any changes. In the second method, we used 10% of training data for the fine-tuning phase instead of all the training dataset. In the third method, we changed the early termination condition during the fine-tuning phase to be based on F1 score improvement for FTNB instead of accuracy and used 10% of the training data for the fine-tuning phase.

Figure 2 (b) shows that for the CIC-IDS17 and Ransomware datasets, fine tuning FTNB-ID on all training data resulted in a higher F1 score compared to fine tuning on 10% of the training data. This was due to very scarce data for almost half of the classes. However, the third dataset (WSN) does not have very scarce data for each class; therefore, there was no such improvement in F1 score when we fine-tuned with 100% of training data compared to 10%.

On the other hand, FTNB with fine tuning on all training data had a better F1 score for the CIC-IDS17 dataset by only 7% compared to fine tuning on 10% of training data.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

Changing the termination condition to be based on F1 score in (method 3) increased the F1 for the original FTNB by 5% for the same dataset, but it made only a slight difference for the other two datasets. We tested the obvious hypothesis in the next section using the UCI datasets and different fine-tuning training data sizes. Detailed results for the third method are shown in Tables 2–4.

Table 2: Test results (precision, recall, and F1 score) for the CIC-IDS'17 dataset.

		F1 Score			**Precision**			**Recall**		
Class	Support	NB	FTNB	FTNB_ID	NB	FTNB	FTNB_ID	NB	FTNB	FTNB_ID
Benign	2271320	0.9782	0.9995	**0.9997**	0.9998	0.9998	0.9998	0.9576	0.9992	**0.9996**
Infiltration	36	0.0190	0.4523	**0.5703**	0.0096	0.3431	**0.5636**	**0.7833**	0.7583	0.6750
Bot	1956	0.0905	0.9126	**0.9171**	0.0474	0.8717	**0.9083**	**0.9969**	0.9606	0.9269
PortScan	158804	0.9928	0.9985	**0.9990**	0.9906	0.9979	**0.9988**	0.9950	0.9992	0.9992
DDoS	128025	0.9988	0.9990	**0.9992**	0.9990	0.9988	**0.9992**	0.9986	0.9992	0.9992
FTP-Patator	7935	0.9750	0.9936	**0.9980**	0.9532	0.9891	**0.9980**	0.9977	0.9981	0.9981
SSH-Patator	5897	0.2859	0.9827	**0.9906**	0.1670	0.9722	**0.9876**	**0.9939**	0.9936	0.9937
Slowloris	5796	0.9865	0.9918	**0.9921**	0.9809	0.9900	**0.9921**	0.9921	**0.9936**	0.9921
Slowhttptest	5499	0.9170	0.9751	**0.9841**	0.8561	0.9620	**0.9795**	0.9875	0.9885	**0.9887**
DoS Hulk	230124	0.9981	0.9993	**0.9994**	0.9981	0.9991	**0.9992**	0.9982	0.9995	0.9995
GoldenEye	10293	0.9851	0.9923	**0.9937**	0.9797	0.9880	**0.9922**	0.9906	**0.9967**	0.9953
Heartbleed	11	0.0049	0.4577	**0.6800**	0.0025	0.3085	**0.4833**	1.0000	1.0000	1.0000
Brute Force	1507	0.1350	0.9104	**0.9361**	0.0729	0.9020	**0.9421**	0.9117	0.9203	**0.9309**
XSS	652	0.0419	0.7923	**0.8672**	0.0489	0.7147	**0.8459**	0.0369	**0.9019**	0.8912
Sql Injection	21	0.0193	0.1537	**0.2173**	0.0100	0.1136	**0.2041**	0.2833	0.2833	0.2833

Table 3: Test result (precision, recall, and F1 score) for the Ransomware dataset.

		F1 Score			**Precision**			**Recall**		
Class	Support	NB	FTNB	FTNB_ID	NB	FTNB	FTNB_ID	NB	FTNB	FTNB_ID
Goodware	942	0.7523	0.6695	**0.8902**	0.6156	0.6855	**0.9009**	0.9671	0.4946	**0.8822**
Citroni	50	0.0000	0.4932	**0.6772**	0.0000	0.5659	**0.8100**	0.0000	0.6200	**0.6000**
CryptLocker	107	0.0451	0.2605	**0.4692**	0.1333	0.1412	**0.4029**	0.0273	0.1400	**0.7218**
CryptoWall	46	0.0000	**0.0051**	0.0000	0.0000	**0.0026**	0.0000	0.0000	**0.1000**	0.0000
Kollah	25	0.0000	**0.0160**	0.0000	0.0000	**0.0082**	0.0000	0.0000	**0.4000**	0.0000
Kovter	64	0.0000	**0.0286**	0.0000	0.0000	0.0167	0.0000	0.0000	0.1000	0.0000
Locker	97	0.0548	0.1180	**0.5727**	0.2000	0.1389	**0.5896**	0.0322	0.1211	**0.5600**
Matsnu	59	0.0857	0.1655	**0.3073**	0.3000	0.3750	**0.7417**	0.0500	0.1200	**0.2033**
Pgpcoder	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Reveton	9	0.0000	0.0909	**0.5861**	0.0000	0.0962	**0.5440**	0.0000	0.2556	**0.6778**
TeslaCrypt	6	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Trojan-Ransom	34	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 4: Test result (precision, recall, and F1 score) for WSN dataset.

		F1 Score			**Precision**			**Recall**		
Class	Support	NB	FTNB	FTNB_ID	NB	FTNB	FTNB_ID	NB	FTNB	FTNB_ID
Normal	340066	0.9857	0.8980	**0.9943**	**0.9987**	0.9985	0.9963	0.9729	0.8296	**0.9923**
Flooding	3312	0.8797	**0.9338**	0.9312	0.7880	0.8925	**0.9242**	**0.9958**	0.9792	0.9387
Grayhole	14596	0.8539	0.8291	**0.9284**	0.7759	0.6927	**0.9164**	0.9494	**0.9899**	0.9416
Blackhole	10049	0.9421	0.8569	**0.9604**	0.9557	0.9914	**0.9651**	0.9289	0.7492	0.9558
TDMA	6638	0.6807	0.3682	**0.8526**	0.5359	0.2692	**0.8060**	0.9328	**0.9444**	0.9141

In Table 5, we report the time complexity in terms of number of epochs for the Fine-tuning process for both Original FTNB compared to FTNB-ID. We can see FTNB terminates the fine-tuning process earlier than FTNB-ID (faster) and this suggests the reason for FTNB-ID outperforming FTNB. This clearly seen in WSN dataset with the fewest iterations and the most outperformed result in the applied Method 3.

Table 5: Average number of epochs of the 10 folds

Dataset	NB	FTNB	FTNB-ID
CIC-IDS 2017	-	5.4	8.3
Ransomware	-	3.7	4.7
WSN	-	2	4.1

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

In order to verify the effectiveness of our proposed (FTNB-ID), we extend the experiment for imbalanced classification to compare our work with relatively recent four state-of-the-art ensemble classifiers for imbalanced datasets. Namely, BalancedBagging [43], BalancedRandomForest [44], RUSBoost [45], EasyEnsemble [46] which allows to bag Adaptive Boosting Classifier as learners trained on balanced bootstrap samples. Moreover, we integrate all classifiers with data-level approaches (11 different resampling methods) which tackles the class imbalance problem.

In this set of experiments, we use imbalanced-learn Python package [47] to implement 11 resampling methods and four Ensemble Imbalanced Classifier, and we directly use their default hyper-parameters. We conduct our experiments on 10% stratified sampling of WSN dataset described in previous section for efficiency. We chose WSN since Over-sampling approaches such as SMOTE and its variants require minimum number of instances (k=3) per class in order for KNN algorithm to generate synthesis data of minor classes. WSN dataset has sufficient instances for each minor class and therefore can be sampled for faster experiments of total of 7 classifiers and 11 different resampling approaches per classifier.

Table 6: comparison of FTNB-ID with other classifiers (weighted F1 score)

Category ●	**Method ○**	**FTNB-ID (OURS)**	**NB**	**Balanced Bagging** [43]	**Balanced R.Forest** [44]	**Easy Ensemble** [45]	**RUS Boost** [45]	**#Training Samples**
No resampling	-	**0.952**	0.882 ●	0.915 ●	0.912 ●	0.636 ●	0.659 ●	37,467
Under-sampling	RandomUS	0.952	0.954	0.964 ○	**0.968** ○	0.647 ●	0.702 ●	1,655
	NearMiss [48]	**0.943**	0.888 ●	0.930	0.939	0.680 ●	0.716 ●	1,655
Over-sampling	RandomOS	**0.999**	0.974 ●	0.999 ●	0.999 ●	0.651 ●	0.651 ●	170,035
	SMOTE [13]	0.994	0.979 ●	0.997 ○	**0.998** ○	0.780 ●	0.780 ●	170,035
	ADASYN [49]	0.994	0.977 ●	0.994	**0.996** ○	0.764 ●	0.739 ●	169,967
Cleaning-sampling	ENN [50]	**0.992**	0.846 ●	0.936 ●	0.942 ●	0.734 ●	0.791 ●	34,471
	TomekLinks [51]	**0.953**	0.884 ●	0.927 ●	0.929 ●	0.794 ●	0.777 ●	37,078
	AllKNN [52]	**0.984**	0.850 ●	0.928 ●	0.942 ●	0.890 ●	0.847 ●	35,445
	OOS [53]	**0.952**	0.884 ●	0.926 ●	0.925 ●	0.683 ●	0.703 ●	35,699
Over-sampling + Cleaning	SmoteENN [54]	0.996	0.984 ●	0.998 ○	**0.999** ○	0.781 ●	0.835 ●	164,019
	SmoteTomek [55]	0.994	0.979 ●	0.998 ○	**0.998** ○	0.856 ●	0.761 ●	169,449
Win/Loss/Tie			**11 / - / 1**	**6 / 6 / -**	**6/ 6 / -**	**12 / - / -**	**12 / - / -**	
Significant (95%) Win/Loss			**11 / -**	**6 / 4**	**6 / 5**	**12 / -**	**12 / -**	
Time (Minutes)		**30.3**	**0.5**	**4.3**	**11.2**	**42.0**	**27.3**	

● FTNB-ID is significantly better; ○ FTNB-ID is significantly worse.

Table 6 shows the (unweighted) Fscore average measures of 10-folds cross validation and the significant t.test (95%) results. The result reveals that our proposed (FTNB-ID) without resampling technique significantly outperforms all other classifiers. Moreover, FTNB-ID without resampling technique outperforms other classifiers with resampling techniques except two classifiers with oversampling techniques. The result shows that over- and cleaning-sampling increase the classifiers performance as opposed to under-sampling techniques.

In this experiment, only two classifiers have tight performance with FTNB-ID. Namely, Balanced Random Forest and Balanced Bagging significantly outperform FTNB-ID in five and four methods, respectively, while FTNB-ID significantly outperforming in six methods for each one. On the other hand, Easy Ensemble is the worst classifier and the slowest. Table 6 shows the time complexity for each classifier without the resampling time

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

since we generated in advance the resampling files and then generate 10 stratified sampling files to be used for each classifier 10-fold cross validation.

4.2.2. Result for Balanced (UCI) datasets

This set of experiments was conducted using 57 UCI datasets (Table 7) i) to test the proposed method on balanced and more diverse datasets, and ii) to test the effect of different amounts of training data being used for the fine-tuning phase. We compared the proposed FTNB-ID with NB and the modified FTNB, as described in Method 3, in the previous section. For fine-tuning training data size, we chose various thresholds, T = 10%, 20%, 50% and 100%, for stratified sampling of each training dataset.

Figure 3 shows that FTNB-ID, by average accuracy and F1 score, for all 57 datasets, performed better compared to NB, and slightly better than FTNB, for all different thresholds (T) sizes. The performance of FTNB-ID increased when (T) size increased. In contrast, FTNB did not have the same steady increase; it gave the worst result when (T) = 50%.

We used a paired two-tailed t-test with 95% confidence for the 10-fold cross validation and used T=100%. The results reveal that FTNB-ID significantly outperforms NB and FTNB with respect to accuracy and F score; see Table 7.

The results show that FTNB-ID significantly outperformed NB in 27 and 25 datasets and NB significantly outperformed FTNB-ID in 5 and 3 datasets, in terms of accuracy and F score, respectively. In addition, FTNB-ID significantly outperformed FTNB in 17 and 18 datasets and FTNB significantly outperformed FTNB-ID in 2 and 5 datasets in terms of accuracy and F score, respectively.

Figure 3: UCI datasets result with various fine-tuning thresholds (10%, 20%, 50% and 100%).

Table 7: Test results for UCI dataset of NB, FTNB and FTNB-ID.

#	Dataset	Inst.	class	Att.	Accuracy			F1Score		
					NB	FTNB	FTNB-ID	NB	FTNB	FTNB-ID
1	anneal	898	6	39	0.9644 ●	0.9800 ●	0.9922	0.9219 ●	0.9350 ●	0.9666
2	anneal.orig	898	6	39	0.9633 ●	0.9744	0.9722	0.9256 ●	0.942	0.9384
3	arrhythmia	452	16	280	0.7544 ●	0.4071 ●	0.7942	0.5204 ●	0.4808 ●	0.6018
4	audiology	226	24	70	0.7345 ●	0.7434	0.7611	0.4923 ●	0.6285 ○	0.5582
5	autos	205	7	26	0.7415 ●	0.8293	0.8049	0.7006 ●	0.8297	0.7789
6	balance.scale	625	3	5	0.7856 ●	0.8112	0.8096	0.5448 ●	0.5643 ●	0.6844
7	breast.cancer	286	2	10	0.7168 ○	0.6783	0.6748	0.6397 ○	0.6285	0.5874
8	breast.w	699	2	10	0.9728	0.9599 ●	0.9685	0.9703 ○	0.9557 ●	0.9653
9	bridges.version1	108	6	13	0.7143	0.6952	0.7238	0.5284	0.5446	0.5509

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 23 November 2022 doi:10.20944/preprints202211.0435.v1

10	bridges.version2	108	6	13	0.6667 ●	0.7143	0.7143	0.5146 ●	0.5891	0.5894
11	car	1728	4	7	0.8553 ●	0.9201	0.9236	0.6393 ●	0.843	0.8486
12	colic	368	2	23	0.7989 ●	0.8071	0.8397	0.7869 ●	0.796	0.8284
13	colic.orig	368	2	28	0.7255	0.7473	0.7609	0.7108	0.7278	0.7359
14	credit.a	1000	2	21	0.8565 ○	0.8478	0.8420	0.8531	0.8408	0.8395
15	credit.g	690	2	16	0.762	0.749	0.7440	0.7041 ○	0.6953 ○	0.6628
16	cylinder.bands	512	2	40	0.7741	0.8074	0.8148	0.768	0.797	0.8088
17	dermatology	366	6	34	0.9754	0.9781	0.9754	0.9721	0.9755	0.9721
18	diabetes	768	2	9	0.7695 ○	0.7044 ●	0.7396	0.7474	0.7005 ●	0.7295
19	ecoli	363	8	9	0.8512	0.8631	0.8542	0.7319	0.7671 ○	0.7489
20	flags	194	8	31	0.6186	0.6134	0.5979	0.4738	0.4996	0.4645
21	glass	214	7	10	0.7336	0.7523	0.7804	0.6577	0.7046	0.7117
22	haberman	306	2	4	0.7386	0.7092	0.7353	0.5795	0.5987	0.5756
23	heart.c	303	5	14	0.8482	0.8449	0.8416	0.8442	0.8435	0.8382
24	heart.h	294	5	14	0.8401	0.8367	0.8367	0.8234	0.8303	0.8227
25	hepatitis	155	2	20	0.871	0.8774	0.8581	0.8119	0.7873	0.7707
26	hypothyroid	3772	4	30	0.9836 ●	0.9907	0.9926	0.8809 ●	0.9208	0.9176
27	ionosphere	351	2	35	0.9003 ●	0.9316	0.9316	0.8886 ●	0.9265	0.9245
28	iris	150	3	5	0.9667	0.9733	0.9733	0.9665	0.9732	0.9732
29	kr.vs.kp	3196	2	37	0.8789 ●	0.9152 ●	0.9662	0.8785 ●	0.9151 ●	0.9661
30	labor	57	2	16	0.9123	0.9298	0.9298	0.912	0.9273	0.9273
31	letter	20000	26	17	0.7411 ●	0.7802 ●	0.8332	0.7430 ●	0.7786 ●	0.8325
32	lymph	148	4	19	0.8311	0.8514	0.8446	0.8523	0.8734	0.8670
33	mushroom	8124	2	23	0.9583 ●	0.9995	0.9995	0.9581 ●	0.9995	0.9995
34	nursery	12960	5	9	0.9032 ●	0.9116 ●	0.9191	0.6840 ●	0.8228 ●	0.8465
35	optdigits	5620	10	65	0.9253 ●	0.9452 ●	0.9553	0.9257 ●	0.9453 ●	0.9554
36	page.blocks	5473		10	0.9370 ●	0.9571 ●	0.9691	0.7459 ●	0.8065 ●	0.8542
37	pendigits	10992	10	17	0.8802 ●	0.9502 ●	0.9633	0.8781 ●	0.9501 ●	0.9633
38	postoperative.patient	90	3	8	0.6667 ○	0.4444	0.5111	0.3771	0.373	0.3581
39	primary.tumor	339	2	17	0.5015 ○	0.4366 ●	0.4867	0.3272	0.3224	0.3148
40	segment	2310	7	20	0.9143 ●	0.945	0.9459	0.9133 ●	0.9448	0.9457
41	shuttle.landing.control	15	2	6	0.9333	0.9333	0.9333	0.9333	0.9333	0.9333
42	sick	3772	2	30	0.973	0.9040 ●	0.9738	0.8872	0.7427 ●	0.8897
43	solar.flare1	1066	6	13	0.9319 ●	0.9505	0.9598	0.6971	0.6514	0.6580
44	solar.flare2	1066	6	13	0.9765 ●	0.9906	0.9916	0.8197	0.7972	0.7976
45	sonar	208	2	61	0.8558	0.8462	0.8702	0.8545	0.8422	0.8692
46	soybean	683	19	36	0.9297	0.9283	0.9253	0.9526	0.9542	0.9520
47	spambase	4601	2	58	0.9024 ●	0.8211 ●	0.9381	0.8962 ●	0.8207 ●	0.9351
48	spect.test	267	2	44	0.6898	0.6310 ●	0.6738	0.6787	0.6161 ●	0.6591
49	splice	3190	3	62	0.953	0.9455 ●	0.9520	0.9478	0.9408 ●	0.9468
50	sponge	76	12	45	0.9211	0.9342	0.9342	0.7823	0.7828	0.7828
51	tic-tac-toe	958	2	9	0.6962 ●	0.9823 ○	0.9530	0.6378 ●	0.9800 ○	0.9490
52	trains	10	2	33	0.7	0.7	0.7000	0.7	0.7	0.7000
53	vehicle	846	4	19	0.6324 ●	0.6986	0.7222	0.6127 ●	0.6929 ●	0.7204
54	vote	435	2	17	0.9011 ●	0.9471 ○	0.9287	0.8976 ●	0.9443 ○	0.9246

● FTNB-ID is significantly better. ○ FTNB-ID is significantly worse.

Continued: Table 7: Test result for UCI dataset of NB, FTNB and FTNB-ID.

#	Dataset	Inst.	class	Att.	Accuracy			F1Score		
					NB	FTNB	FTNB-ID	NB	FTNB	FTNB-ID
55	vowel	990	11	14	0.6707 ●	0.6990 ●	0.7970	0.6676 ●	0.6969 ●	0.7948
56	waveform5000	5000	3	41	0.8062 ●	0.7962 ●	0.8476	0.7949 ●	0.7840 ●	0.8476
57	zoo	101	7	18	0.9307	0.9307	0.9307	0.8762	0.8762	0.8762
	Win/Loss/Tie (FTNB-ID)				**38/15/4**	**32/15/10**		**36/17/4**	**29/21/7**	
	Significant (95%) Win/Loss				**27/5**	**17/2**		**25/3**	**18/5**	

● FTNB-ID is significantly better. ○ FTNB-ID is significantly worse.

5. Conclusions

 doi:10.20944/preprints202211.0435.v1

This work proposed a fine-tuning algorithm for NB that is more suitable for imbalanced datasets than the original FTNB algorithm. The proposed algorithm (FTNB-ID) determines the size of the granular update step based on the complement classes' harmonic average (predicted vs actual class) of the probability terms. This makes the update size large when rare and common classes have very skewed or scarce data and small otherwise. We evaluated the performance of the algorithm with respect to precision, recall and F1 score. Our empirical analysis revealed that with respect to the F1 score, FTNB-ID significantly outperforms NB (39%, 37% and 7%) and FTNB (4%, 34% and 14%) on imbalanced datasets. In addition, we tested the effect of the proposed method on balanced datasets, and the results showed slight improvements (3% and 1%) with respect to the F1 score for FTNB-ID compared to NB and FTNB. As future work, we intend to investigate using the harmonic average for feature weighting in Bayesian network classifiers.

6. Acknowledgement

The authors would like to thank the Deanship of Scientific Research at King Saud University for funding and supporting this research through the initiative of DSR Graduate Students Research Support (GSR).

7. Author Contributions

This research is part of Fahad Alenazi Ph.D. dissertation work under the supervision of Khalil El Hindi. Extensive discussions about the algorithms and techniques presented in this paper were made among the authors; Basil AsSadhan advised Fahad Alenazi in designing and conceiving the experiments; Fahad Alenazi performed the experiments; the manuscript was written by Fahad Alenazi; all authors reviewed and approved the final manuscript.

8. Conflicts of Interest

The authors declare no conflict of interest.

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, p. 75–306, 2010.

[2] G. Fatma, S. C. Okan, E. Zeki and K. Olcay, "Online naive bayes classification for network intrusion detection," *In Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '14)*, 2014.

[3] P. Alaei and N. Fakhroddin, "Incremental anomaly-based intrusion detection system using limited labeled data," *In 3th International Conference on Web Research (ICWR), IEEE*, pp. 178-184, 2017.

[4] S. Ren, L. Yangyang and Z. and Xiaojian, "Incremental Naïve Bayesian Learning Algorithm based on Classification Contribution Degree," *J. Comput.*, vol. 9, no. 8, pp. 1967-1974, 2014.

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

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

[7] 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*, p. 249–256, 2003.

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

[9] K. El Hindi, "Fine tuning the Naïve Bayesian learning algorithm," *AI Commun*, vol. 2, p. 133–141, 2014.

[10] 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, p. 237–246, 2014.

[11] S. Wang and X. Yao, "Multiclass imbalance problems: analysis and potential solutions.," *IEEE Trans Syst Man Cybern Part B Cybern*, vol. 42, no. 4, pp. 1119-1129 , 2012.

[12] J. Bia and C. Zhang, "An empirical comparison on state-of-the-art multi-class imbalance learning algorithms and a new diversified ensemble learning scheme.," *Knowl Based Syst*, no. 158, pp. 81-93, 2018.

[13] C. Nitesh V, B. Kevin W, H. Lawrence O and K. W Philip, "Smote: synthetic minority over-sampling technique.," *Journal of artificial intelligence research, ,* vol. 16, p. 321– 357, 2002.

[14] V. García, J. Sánchez, A. Marqués and e. al, "Understanding the apparent superiority of over-sampling through an analysis of local information for class-imbalanced data.," *Expert Syst Appl*, vol. 158, pp. 113-126, 2020.

[15] J. Mathew, C. Pang, M. Luo and e. al, "Classification of imbalanced data by oversampling in kernel space of support vector machines.," *IEEE Trans Neural Netw Learn Syst* , vol. 29, no. 9, pp. 4065-4076 , 2018.

[16] S. Raghuwanshi and S. Shukla, "SMOTE based class-specific extreme learning machine for imbalanced learning.," *Knowl Based Syst* , vol. 187, no. 104814 , 2020.

[17] W. Tzu-Tsung and T. Hsing-Chen, "Multinomial naïve Bayesian classifier with generalized Dirichlet priors for high-dimensional imbalanced data," *Knowledge-Based Systems*, vol. 228, no. 107288, 2021.

[18] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," *Publishers Inc., San Francisco, CA, USA: Morgan Kaufmann*, 1988.

[19] G. F. Cooper, "The computational complexity of probabilistic inference using bayesian belief networks,"," *Artif. Intell.*, vol. 42, no. 2-3, p. 393–405, 1990.

[20] D. M. Chickering, "Learning Bayesian Networks is NP-Complete," *in Learning from Data, Springer New York*, vol. 112, p. 121–130, 1996.

[21] L. Jiang, H. Zhang and Z. Cai, "A Novel Bayes Model: Hidden Naïve Bayes," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 10, p. 1361–1371, 2009.

[22] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees,"," *IEEE Trans. Inf. Theory*, vol. 14, no. 3, p. 462–467, 1968.

[23] L. Jiang, Z. Cai, D. Wang and H. Zhang, "Improving Tree Augmented Naïve Bayes for Class Probability Estimation," *Know-Based Syst*, vol. 26, p. 239–245, 2012.

[24] 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, p. 273–286, 2013.

[25] M. Hall, "A decision tree-based attribute weighting filter for naive bayes," *Knowledge- Based Systems*, vol. 20, p. 120–126, 2007.

[26] P. Langley and S. Sage, "Induction of selective Bayesian classifiers," *in Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, p. 339–406, 1994.

[27] L. Jiang, H. Zhang, Z. Cai and J. Su, "Evolutional naïve bayes," *In Proceedings of the International Symposium on Intelligent Computation and its Application, ISICA*, p. 344–350, 2005.

[28] L. Jiang and Y. Guo, "Learning lazy naïve Bayesian classifiers for ranking," *in 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'05),* p. 5– 16, 2005.

[29] L. Jiang and H. Zhang, "Learning instance greedily cloning naïve Bayes for ranking," *in Fifth IEEE International Conference on Data Mining (ICDM'05),* p. 8, 2005.

[30] L. Jiang, D. Wang, Z. Cai and X. Yan, "Survey of Improving Naïve Bayes for Classification," *in Advanced Data Mining and Applications,* p. 134–145, 2007.

[31] L. Jiang and D. a. C. Z. Wang, "Discriminatively weighted naïve bayes and its application in text classification," *Int. J. Artif. Intell. Tools,* vol. 21, no. 1, pp. 125-137, 2012.

[32] A. Alhussan and K. El Hindi, "Selectively Fine-Tuning Bayesian Network Learning Algorithm," *Int. J. Pattern Recognit. Artif. Intell.,* vol. 30, no. 6, pp. 165-175, 2016.

[33] L. Breiman, "Bagging Predictors," *Mach Learn,* vol. 24, no. 2, p. 123–140, 1996.

[34] K. El Hindi, "Combining Instance Weighting and Fine Tuning for Training Naïve Bayesian Classifiers with Scant data," *International Arab Journal of Information Technology (IAJIT),* vol. 15, no. 6, pp. 1099-1106, 2016.

[35] D. M. Diab and K. El Hindi, "Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification," *Appl. Soft Comput.,* vol. 54, p. 183–199, 2017.

[36] S. Guan and N. Fu, "Class imbalance learning with Bayesian optimization applied in drug discovery.," *Sci Rep,* vol. 12, no. 2069, 2022.

[37] F. S. Alenazi, K. El Hindi and B. AsSadhan, "Fine-Tuning Naïve Bayes for Imbalanced Datasets," *Int'l Conf. Data Science, ICDATA'19* , 2019.

[38] I. W. a. E. Frank, "Data Mining: Practical Machine Learning Tools and Techniques," *Morgan Kaufmann,* 2005.

[39] "UNP, Canadian Institute for Cybersecurity, "Intrusion Detection Evaluation Dataset (CICIDS2017)"," *Available, https://www.unb.ca/cic/datasets/ids-2017.html* , 2021.

[40] 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],* 2016.

[41] I. Almomani, B. Al-Kasasbeh and M. AL-Akhras, "WSN-DS: A Dataset for Intrusion Detection Systems in Wireless Sensor Networks," *Journal of Sensors* , p. 16, 2016.

[42] C. Blake and C. Merz, "UCI Repository of machine learning databases, Univ. Calif.," 1998.

[43] W. Shuo and Y. Xin, "Diversity analysis on imbalanced data sets by using ensemble models.," *Symposium on computational intelligence and data mining, IEEE,* p. 324–331, 2009.

[44] "Chao Chen, Andy Liaw, Leo Breiman, and others. Using random forest to learn imbalanced data. University of California, Berkeley," vol. 110, no. (1-12):24, p. 2004, 2004.

[45] S. Chris, K. Taghi M, H. Jason Van and N. Amri, "Rusboost: a hybrid approach to alleviating class imbalance.," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, ,* vol. 40, no. 1, p. 185–197, 2009.

[46] L. Xu-Ying, W. Jianxin and Z. Zhi-Hua, "Exploratory undersampling for class-imbalance learning.," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), ,* vol. 39, no. 2, p. 539–550, 2008.

[47] L. Guillaume, N. Fernando and A. Christos K., "Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning.," *Journal of Machine Learning Research,* vol. 18, no. 17, p. 1–5, 2017.

[48] M. Inderjeet and Z. I, "knn approach to unbalanced data distributions: a case study involving information extraction.," *In Proceedings of workshop on learning from imbalanced datasets,* vol. 126, 2003.

 doi:10.20944/preprints202211.0435.v1

[49] H. Y. B. Haibo, G. Edwardo A and L. Shutao, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning.," *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, p. 1322–1328, 2008.

[50] W. Dennis L, "Asymptotic properties of nearest neighbor rules using edited data.," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, p. 408–421, 1972.

[51] T. Ivan, "Two modifications of cnn," *IEEE Trans. Systems, Man and Cybernetics, ,* vol. 6, p. 769–772, 1976.

[52] T. Ivan, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions on systems, Man, and Cybernetics, ,* vol. 6, p. 448–452, 1976.

[53] K. Miroslav, M. Stan and e. al., "Addressing the curse of imbalanced training sets: one-sided selection.," *In Icml, Nashville, USA.*, vol. 97, p. 179–186, 1997.

[54] B. Gustavo EAPA, P. Ronaldo C and M. Maria Carolina, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, p. 20–29, 2004.

[55] B. Gustavo EAPA, B. Ana LC and M. Maria Carolina, "Balancing training data for automated annotation of keywords: a case study," *In WOB*, pp. 10-18, 2003.