Conference on
Information Technology and
Data Science

# Hybrid AdaBoost and Naïve Bayes Classifier for Supervised Learning

**Ahiya Ahammed, Balazs Harangi, Andras Hajdu**

University of Debrecen, Faculty of Informatics
ahiya.evan@gmail.com
harangi.balazs@inf.unideb.hu
hajdu.andras@inf.unideb.hu

*Proceedings of the 1st Conference on Information Technology and Data Science*
*Debrecen, Hungary, November 6–8, 2020*
*published at http://ceur-ws.org*

**Abstract**

Supervised learning is a task of machine learning that maps an input to an output based on available data. The data set contains the information from which we get to know the process of classifying at least some of the data correctly. Thus, the classified data is called Training set. In this learning method, the supervision comes from the instances having labels in the training set. Classification problems are mostly referred to come from the branch of supervised learning. Classification is a function of machine learning that finds out the correctly predicted class labels of instances for all the unlabelled instances. In this research, our main focus will be discussing about the most commonly used data classification methods especially Naïve Bayes (NB) and Boosting basically Adaptive Boosting (AdaBoost) classifiers. In preparation for the enhancement of the performances with regards to the accuracy rate of those classifiers we would like to introduce two newly hybrid approaches for classification when the data set is big, noisy and high dimensional. In real life, the data sets usually contain noise or outliers, contradictory instances or missing values. Mostly the data got affected by them during the time of data collection or generation. To resolve that issue, we proposed two new hybrid classifiers. Our first hybrid approach is the ADA+NB classifier, where we used Adaptive Boosting (AdaBoost) classifier to find comparatively more important attribute subsets before the assumption of class conditional independence using Naïve Bayes (NB) classifier [13]. Besides, NB classifier assumes class conditional independence, therefore, it's possible to multiply the probabilities when the events are independent. Because of that, NB classifier can be very effective in removing examples from the training set before the decision tree (DT) generation at the time of building AdaBoost model. We named this process our second proposed hybrid NB+ADA classifier [14]. This paper investigates the comparison between two classical machine learning approaches with two new hybrid classifiers in terms of accuracy rate, error rate, precision, f-score, sensitivity, specificity analysis on 4 real benchmark data sets who are high dimensional and noisy chosen from UCI (University of California, Irvine) machine learning repository. For instance, Adult data is one of the renowned noisy data set available in UCI. Therefore, we tested AdaBoost classifier which gave us accuracy of 87.65% and NB classifier provided 79.99%. On the other hand, our first proposed (ADA+NB) classifier showed 86.39% and our second proposed (NB+ADA) indicated 94.14% of the accuracy rate with the same data. Similarly, we used other data-sets and derived performance comparison between the classifiers to prove that our proposed classifier's performances are higher than the text-book classifier's.

*Keywords:* Supervised learning, classification, hybrid classifier, decision tree, adaptive boosting, Naive Bayes classifier and performance comparison

## 1. Introduction

This research intends to establish two new, strong and powerful hybrid classifiers with the following names ADA+NB and NB+ADA and compare them with two existing machine learning approaches such as AdaBoost and Naïve Bayes classifiers in order to gain the highest accuracy rate. Currently, increasing number of researchers are using different classical machine learning algorithms to solve different regression, classification and clustering problems. Classification problems are the most widely studied and filled with plenty of new research domains in the area of data science. Different applications of classification are being used in variety of problem domains such as multimedia, text, medical or biological data and social networking. These domains are using various methods of data classification such as neural networks, decision trees, probabilistic or rule-based classification, Bagging, Boosting, Support Vector Machine (SVM) and other instance-based techniques etc. This paper consists of 2-major portions: comparative analysis on the machine learning models, evidence of choosing our proposed classifiers over 2-classical machine learning classifiers with respect to higher accuracy rates, especially when the training data is too noisy and large in dimension.

The first part "comparative analysis of machine learning models", contains the comparison between the classical machine learning algorithms such as Adaptive Boosting (AdaBoost) and Naïve Bayes (NB) with our two proposed hybrid algorithms, first hybrid classifier denoted as ADA+NB, because it starts with Adaboost induction and then it classifies with the NB classifier and second one is NB+ADA, because it starts with NB induction and then it classifies according to the AdaBoost classifier model. In the second part, the "evidence of choosing our proposed algorithms" makes the comparison between our proposed algorithms with the classical Naïve Bayes (NB) classifier and Adaptive Boosting (AdaBoost) classifier, and states that when the data sets are noisy or there are presence of outliers and respectively high in dimensions the proposed hybrid classifiers perform well, when it comes to the accuracy rate, error rate, precision, f1-score, sensitivity/recall or specificity records analysis. Generally, the classifier's performance in machine learning depends on the data quality. Sometimes data with the lower quality are used as the training set to build the model that might lead to creating the over-fitting problems for that particular model. For instance, though AdaBoost is a strong classifier, due to the presence of noise or outliers in the training data, if anyone tries to build AdaBoost model from that data, decision trees (DT) might suffer from the over fitting problem, during the time of tree generation. As a result, being a classifier with high calculations AdaBoost fails to provide the expected accuracy rate [3].

Therefore, it is really important to pre-process the data set before it can be used for the training. Data pre-processing can improve the data quality. As a result, it can take tremendous part to enhance the classifier model's efficiency. Therefore, accuracy might also be improved. In paper [14] several pre-processing techniques are available such as:

1. Data cleaning: Removing noise or outliers or missing values.
   (a) Outliers or anomaly: Using method of binning, process of regression or clustering.
   (b) Missing value: Ignoring the instance or filling the missing values (considering the mean value or manually).
2. Integration: Data are merged from different resources.
3. Transformations: Normalization of data, proper selection of attributes, discretization of the data.
4. Reduction: Cube aggregation of the data, subset selection of attributes, shorten the dimensions.

## 2. Related Work

### 2.1. Adaptive Boosting (AdaBoost) Classifier

During the process of Boosting in paper [1] a weight is assigned for each training example. It also adjusts the weight efficiently at completion of individual cycle. Not correctly classified examples will have more increased weight, whereas correctly classified ones will have more decreased weights. In consecutive iterations, AdaBoost is forced to concentrate on the hardly identified examples.

The paper [13] discussed about a AdaBoost classifier for biological data which have multiple classes. It takes into consideration noise reduction, over-fitting and class-imbalance problems. To stop over-fitting, the classifier considers a random subspace solution. By giving it a flavor of boosting, it pays more emphasis on the incorrectly classified instances in the next round or iteration.

Adaptive boosting was first applied in 1997 by Freund and Schapire [16]. Over-fitting is the main disadvantage of AdaBoost. When the final classifier gets too complicated, the test error increases [4, 10, 22]. The AdaBoost classifier's test error also starts to reduce after the training error becomes 0 and never rise back; when lots of rounds are passed through. Besides, after incredibly huge rounds, the error on the test set was found to be increased marginally [17] Boosting, along with the more developed classifiers, such as the Decision Tree (DT), Neural Network (NN), Support Vector Machine (SVM) and Naïve Bayes (NB) classifier have become one of the alternative mechanisms for classification [6].

AdaBoost classifier's popularity can be due to its potential to stretch the margin according to the research, [23] which may increase the classifier's generalization capabilities. There have been reports of several experiments using Decision Trees (DTs) [9] or Neural Networks [24] or Support Vector Machine (SVM) [19] as element classifiers or learning schemes in Adaptive Boosting (AdaBoost) classifier. Such experiments indicate strong success of these AdaBoost classifier's generalization, with some difficulties still remaining. What should the appropriate size of the tree? Especially when Decision Trees (DTs) are used as learning scheme. And how should manage computational difficulties? Especially when Neural Networks (NNs) are used as learning scheme. Before using AdaBoost classifier, both of these questions need to handled carefully to produce better results.

## 2.2. Decision Tree (DT) Classifier

Construction of decision tree (DT) is very simple and useful approach with a wide number of variables for classifying instances in large data sets. During the process of DT generation, 2-widely known steps are followed:

- Decision tree evolution has to be made so that the examples available on the training can be classified correctly.
- With regard to getting higher classification rate; precision or accuracy rate at time of pruning some redundant nodes and branches needed to be eliminated.

In paper [15] they proposed a new algorithm named Decision Tree Fast Splitting (DTFS), is used for large data sets to construct decision trees (DTs). To extend tree nodes as well as training examples are processed in an incremental way, DTFS used this attribute collection technique. DTFS has stored a limit of N instances in a leaf node in order to avoid saving all the examples available in DT. Until the number of instances available in leaf node that have been stored, met it's maximum, the leaf node was modified accordingly.

A new approach of constructing Decision Tree (DT) while solving classification problems using cluster based analysis method; named Classification based clustering (CbC), was presented in the research paper [3] Similarities were observed between instances while using the clustering algorithms and specified some target attributes. Afterwards, it evaluated the distribution of each cluster's target attributes. When the number of instances contained in a cluster is met for a specific threshold value, all instances in each cluster were listed according to the target attribute's acceptable value.

They recommended a new classification scheme on a classifier for the Decision Tree (**C4.5** algorithm) and to enlarge the accuracy rate for solving multi-class problems in [21] and they also introduced an unique approach where, **M** numbers of classifiers were developed by their system, each of which distinguished one class from the other. A procedure is suggested in [2] in the analysis to overcome the possible exceptions in the algorithms of simple decision tree (DT) induction. In their work, the influential element of attributes was discovered, which provided the reliability of the attribute value on the target value. Based on this aspect, the DT was pruned.

In the research [8] an algorithm is made especially for the Fuzzification of decision boundaries without translating attributes into fuzzy linguistic terminology. In order to select suitable attribute for individual node that needed to be split, Gini Index is used by the G-FDT tree. According to the average-points of feature values in which the details of the class altered, the split-points were selected.

## 2.3. Naïve Bayes (NB) Classifier

In the area of data science or machine learning Naïve Bayes (NB) Classifier is most popular especially for classification problems. A new classifier named Hidden Naïve Bayesian (HNB) was added by authors in paper [5] through a system that can be used for the detection of any network's invasion. This greatly increased the attack detection accuracy in Denial of Services (DoS). Main reason of having another layer, representing a individual feature's secret or hidden parent was basic idea in the process of HNB. In terms of higher accuracy rate or lower error rate, the HNB multi-class classifier demonstrated comparatively better in performance [20, 25].

During the research of [7] they suggested a Robust Naïve Bayesian Classifier (RNBC) to address the original limitations, one is over-fitting and another one is underflow, for gene expression data sets classification. Instead of multiplying probabilities, RNBC used an approximate method to provide solutions to over-fitting issues and probability logarithms to deal with the underflow problems.

For the Naïve Bayesian classification of micro-array data analysis, "Partition Conditional Independent Component Analysis (PC-ICA)"; a new approach was suggested by the authors of [11]. They used Class Conditional Independent Variable Analysis (CCICA) approach. In order to conduct the "Independent Component Analysis (ICA)" within individual partition, "PC-ICA" splashed the small sized data samples into various partitions. Inside of each partition, which can consist of many classes, "PC-ICA" also tried to do "ICA" based function uprooting.

A newly proposed classification system for multi classes data classification called Extended Naïve Bayesian (ENB) was suggested in [18]. Categorical and numeric data were used in the mixed data categories. To compute the probability of categorical features, ENB used a classical Naïve Bayes (NB) classifier approach. By observing the average and variation of numeric values, the mathematical principle was given for discretizing numeric features into symbols while treating numeric attributes.

## 3. Classification

In machine learning, classification derives a function from the training set. Training data-set consists of set of input vector or scalar and target output vector or scalar (class-label). The result of a function may be a constant value, or the target values of input vectors can be expected. Sometimes, it is also referred to supervised learning since, before analyzing the data, the class labels are known.

### 3.1. Adaptive Boosting (AdaBoost) Classifier

AdaBoost is a commonly used boosting algorithm that takes into account a number of classifiers and combines each classifier's votes to classify an unknown or known instance.

- Weights are allocated to each training example at the time of boosting
- There is an iterative trained sequence of $k$ classifiers
- $M_i$ is learned after using a classification model, the weights are modified to help the following classifier, $M_{i+1}$ to give more emphasis on the instances incorrectly classified by $M_i$

In Adaptive Boosting (AdaBoost) after following the above mentioned steps, a final boosted classifier, M* is found. It blends individual classifier's votes and these weights of the votes of every classifier is accuracy rate's function. If the original training dataset D is given with a set of d class-labeled instances available, $(x_1, y_1), (x_2, y_2), \ldots, (x_d, y_d)$, where $y_i$ is considered as the class label of instance $x_i$. Assume, assigned weights which were allocated to each training example initially is $\frac{1}{d}$. For the ensemble, generating $k$ classifiers requires $k$ rounds over the rest of the algorithm. In order to shape some size training set, we should consider samples, not needed to be the size d. Sampling with replacement is used so that the same example can be chosen more than once. The probability of being chosen for each instance is dependent on its weight. M is a classifier model, can be generated from $D_i$ training instances. Then it's error can be measured as a test set using $D_i$. The weights of the instances can be modified if needed according to their performances.

When any instance is misclassified the weight of its increased and if that instance is properly classified then its weight is reduced. Judging from instance's weights, classification difficulties can be noticed. Instances with higher weights are hard to classify. Weights are important to select instances that will be participating in the next iterations for generating next classifiers.

**Error Rate:** We measured sum of weights for individual instances in $D_i$ that were not correctly classified by $M_i$ to measure the error rate of $M_i$ model.

$$\text{Error}(M_i) = \sum_{j=1}^{d} \text{error}(x_j) * w_j$$

Here, the error of not correctly classified instance $x_i$ is denoted as error $(x_j)$. If the instance $x_i$ is correctly classified by the classifier, then error $(x_i)$ is 0. In contrast, it is 1. A threshold value 0.5 is used to measure the classifier's performance. If it is more than the threshold, then it cannot be considered. Thus, repetition of the same process is necessary.

**Normalizing Weight:** If an instance was correctly classified by the classifier in $i^{th}$ iteration, then the weight of its multiplied by,

$$\text{Error}\left(\frac{\text{error}(M_i)}{1 - \text{error}(M_i)}\right).$$

According to [12] the weights for all instances are normalized until properly classified instance's weights are changed. We considered the product value of a weight and sum of old weights; divided product value by the new weights, to normalize it.

```
Algorithm: 1: AdaBoost Classifier
    Input: Training data, $D$, number of iterations, $k$ and a learning scheme.
    Output: Ensemble model, $M^*$.
    Method:
     1.  Initialize weight, $x_i \in D$ to $1/d$.
     2.  for $i = 1$ to $k$ do
     3.    Sample $D$ with replacement according to instance weight to obtain $D_i$.
     4.    Use $D_i$ and learning scheme to derive a model, $M_i$,
     5.    Compute error($M_i$).
     6.    if error($M_i$) >= 0.5 then,
     7.      Go back to step 3 and try again.
     8.    end if
     9.    for each correctly classifier $x_i \in D$ do
    10.    Multiply weight of $x_i$ by error($M_i$)/(1 − error($M_i$))
    11.    end for
    12.    Normalize weight of instances.
    13.  end for
         To use $M^*$ to classify new instances, $x_{\text{new}}$:
           • Initialize weight of each class to zero
           • for $i = 1$ to $n$ do
           • $w_i$ = log((1 − error($M_i$))/error($M_i$)) //weight of classifier's vote.
           • $c = M_i(x_{\text{new}})$ //class prediction by $M_i$,
           • add $w_i$, to weight for class $c$.
           • end for
           • return class with largest weight.
```

**Figure 1.** Adaptive Boosting (AdaBoost) classifier.

## 3.2. Naïve Bayesian (NB) Classifier

A fundamental probabilistic classifier contingent on Bayes hypothesis with solid predictions of (Naïve) independence and models of independent characteristics; is called Naïve Bayesian (NB) classifier. While determining the probability of participation for each class, attributes with missing values are perfectly handled by merely omitting the corresponding probabilities by the NB classifier itself. Class conditional independence is also necessary. For instance, the attribute's effect on a defined class can be distinct from other attributes.

NB classifier assumes that an training example X belongs to a class $C_i$ if and only if $P(C_i|X) > P(C_j|X)$ belongs to $j \neq i$ and $1 \leq j \leq m$. Therefore, highest posterior analysis shows $C_i$ class due to that $P(C_i|X)$ is maximized.

$$P(C_i|X) = \frac{P(X|C_i) \times P(C_i)}{P(X)} \tag{3.1}$$

A constant $P(X)$ is for all classes in equation of (3.1), then $P(X|C_i) \times P(C_i)$ has to be increased. If the prior probabilities of the class are not specified, then the classes are generally believed to be identical, i.e. $P(C_1) = P(C_2) = \cdots = P(C_m)$. Thus, maximize $P(X|C_i)$ accordingly or $P(X|C_i) \times P(C_i)$ needed to be increased. The prior probabilities of the class are determined by $P(C_i) = |C_i, D|/|D|$, here, $|C_i, D|$ is the number of available instances that are ready to be trained in $D$ belonging to the $C_i$ class.

It is highly computationally costly to calculate $P(X|C_i)$ in a dataset which contains lots of attributes. Provided the class labels of the events, the attributes are conditionally independent of each other. The equation showed in (3.2) is used for generating $P(X|C_i)$

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \tag{3.2}$$

In equation (3.2), $x_k$ corresponds to the value of the $A_k$ feature, for any $X$ instance. However, datasets that are usually used for the training can be numeric, nominal or continuous or categorical. Attributes are usually believed to have a Gaussian distribution having a mean $\mu$ and standard deviation $\sigma$ which can be represented from the following equations:

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) \tag{3.3}$$

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-p)^2}{2\sigma^2}}$$

In equation (3.3), mean value and standard deviation are denoted by $\mu_{C_i}$ and $\sigma_{C_i}$ respectively, for the attribute value $A_k$ of all training examples that belong to the $C_i$ class. Now, $P(x_k|C_i)$ can be calculated by using the method of substitution using the $g(x, \mu, \sigma)$ value.

**Algorithm: 2: Naïve Bayesian Classifier**

```
Input: $D = \{x_1, x_2, \ldots, x_n\}$ //Training data.
Output: A naïve Bayes model.
Method:
  1. for each class, $C_i \in D$ do
  2.     Find the prior probabilities $P(C_i)$.
  3. end for
  4. for each attribute $A_i \in D$ do
  5.     for each attribute value $A_{ij} \in A_i$, do
  6.     Find the class conditional probabilities $P(A_{ij}|C_i)$.
  7.     end for
  8. end for
  9. for each instance, $x_i \in D$ do
 10. Find the posterior probabilities $P(C_i|x_i)$.
 11. end for
```

**Figure 2.** Naïve Bayesian (NB) classifier.

## 3.3. Proposed Algorithm1 (ADA+NB) Classifier

Our proposed algorithm1 (ADA+NB) classifier is a newly proposed, hybrid and very powerful classifier to solve the problems in classification domains, it made up with two strong classifiers, one is Boosting algorithm basically Adaptive Boosting (AdaBoost), with a learning scheme of decision tree and another one is Naïve Bayes (NB) classifier. It performs better when the dataset is too noisy and high dimensional. We named this classifier ADA+NB because it uses the AdaBoost classifier to detect the noise or missing value from the data. Because AdaBoost has an amazing ability to find the subset of attributes which are comparatively more important. At the beginning, equal weights are assigned to all the training instances and those who are misclassified by AdaBoost gains higher weights. Afterwards, we deleted the noisy instances from the original training data which have higher weights and prepares the data to go for the training and uses Naïve Bayes (NB) classifier to make the predictions. And if we used AdaBoost and try to build model using the training set which is noisy; there is high probabilities that AdaBoost might fall into the over-fitting during the time of Decision tree (DT) generation.

**Algorithm: 3: ADA+NB Classifier**

```
Input: $D = \{x_1, x_2, \ldots, x_n\}$ // Training data.
Output: An ADA+NB Model.
Method:
  1. Assign weights $1/d$ initially to all the instances on $D$.
  2. Split the dataset D into 70% $D_{\text{Train}}$ for training and 30% $D_{\text{Test}}$ for testing.
  3. Use $D_{\text{Train}}$ to build AdaBoost model using algorithm 1 and test each of the instances $x_{\text{Test}}$
     from $D_{\text{Test}}$.
  4. Increase the weights of the misclassified instances at the original data $D$ by the
     followings:
                    $\text{weight} = \text{weight} * 1/2 * (\text{error}/\text{accuracy})$
  5. Delete the data/instances, which have higher weights from the original dataset $D$.
  6. Use the new cleaned dataset $D_{\text{TrainNew}}$ to build a new model NB using the Algorithm 2
     and calculate the performance.
```

**Figure 3.** ADA+NB Classifier.

## 3.4. Proposed Algorithm2 (NB+ADA) Classifier

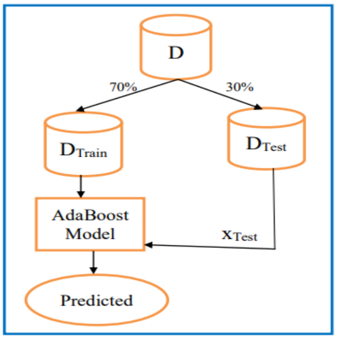
Our proposed algorithm2 (NB+ADA) classifier is another newly proposed, hybrid and powerful classifier to solve the classification problems, it made up with two strong classifiers, one is Naïve Bayes (NB) classifier and another one is powerful Boosting algorithm to be more specific Adaptive Boosting (AdaBoost) classifier. Here, decision trees were used as a weak learning scheme during the AdaBoost induction. It performs better when the data-set is too noisy and larger in dimensions. We named it NB+ADA, the reason behind it is we used NB classifier to find the noise or missing values from the data. As, NB has a ability to handle the missing value by simply ignoring the it or use the mean value. Initially, all the instances were given weights and those instances who were misclassified by the NB classifier have the higher weights than before. Afterwards, we delete the instances, given the priority of higher weights, which can not be handled by NB classifier from the original data assuming that they are anomaly in the data. Then, we proceeded comparatively cleaned data used as the training data-set for AdaBoost to build the model and make the predictions. In that way due to the less presence of anomaly in the recently created training set, AdaBoost would have lower chances of falling into over-fittings.

```
Algorithm: 4: NB+ADA Classifier
  Input: $D = \{x_1, x_2, \ldots, x_n\}$ // Training data.
  Output: A NB+ADA Model.
  Method:
    1. Assign weights $1/d$ initially to all the instances on $D$.
    2. Split the dataset $D$ into 70% $D_{\text{Train}}$ for training and 30% $D_{\text{Test}}$ for testing.
    3. Use $D_{\text{Train}}$ to build NB model using algorithm 2 and test each of the instances $x_{\text{Test}}$
       from $D_{\text{Test}}$.
    4. Increase the weights of the misclassified instances at the original data $D$ by the
       followings:
                 $\text{weight} = \text{weight} * 1/2 * (\text{error}/\text{accuracy})$
    5. Delete the data/instances, which have higher weight from the original dataset $D$.
    6. Use the new cleaned datasct $D_{\text{TrainNew}}$ to build a new model AdaBoost using the
       Algorithm 1 and calculate the performance.
```

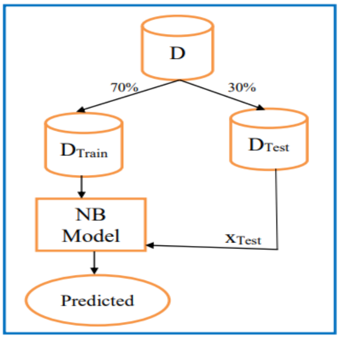
**Figure 4.** NB+ADA Classifier.

## 3.5. Architecture Comparison

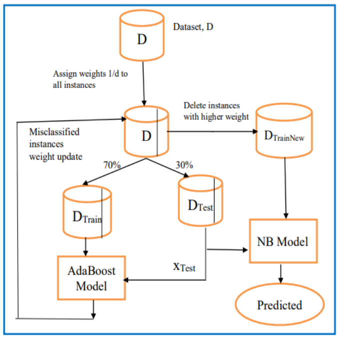
In this section, we can see the plots of the architectural view of the 4-classifiers shown in Figure 5 and compare them. Here, Figure 5(a) shows the architecture of Adaptive Boosting (AdaBoost) classifier. It uses the Algorithm 1 to generate the AdaBoost model. Hence, Figure 5(b) stands for the Naïve Bayesian (NB) classifier architecture using the Algorithm 2 to build a NB model and Figure 5(c) and 5(d) illustrates the architectural model of our proposed classifier1 (ADA+NB) and proposed classifier2 (NB+ADA) which uses the Algorithm 3 and Algorithm 4 to build it's model accordingly. It is clearly seen that all the classifiers that were used in this research, we used the same split-validation process, which is 70% for training and 30% for testing the model. It is a matter of great concern that the amount of data are used to train the classifier's model should be more; if better classification performances are expected from that model.



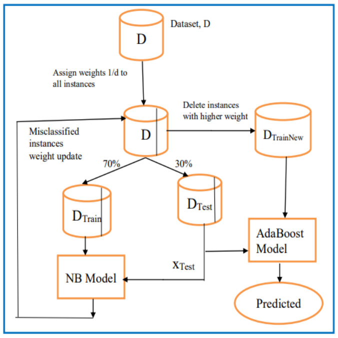
(a)



(b)


D
Dataset, D
Assign weights 1/d to all instances
Delete instances with higher weight
Misclassified instances weight update
D
DTrainNew
70%
30%
DTrain
DTest
NB Model
XTest
AdaBoost Model
Predicted

(c)


D
Dataset, D
Assign weights 1/d to all instances
Delete instances with higher weight
Misclassified instances weight update
D
DTrainNew
70%
30%
DTrain
DTest
AdaBoost Model
XTest
NB Model
Predicted

(d)

**Figure 5.** Classifier's architectural comparisons.

## 3.6. Experiment

This section of the paper describes the data sets that are used for the experiment. It also represents the performance evaluation of Adaptive Boosting (AdaBoost), Naïve Bayesian (NB), our proposed (ADA+NB) and our proposed (NB+ADA) classifiers.

**Datasets:** We used our chosen 4-real benchmark data sets collected from the online free data resource named UCI machine learning repository and also evaluated the classification performances for the above mentioned classifiers. Each of the data sets consists of a two dimensional (2D) Excel spreadsheet. The data sets are following:

- Adult Database
- Covertype Data
- Dota2 Games Results Data

- Isolet Database

**Table 1.** Data-set Description.

| **Data-sets** | **No. of Att.** | **Att. Types** | **No. of Instances** | **No. of Classes** |
|---|---|---|---|---|
| Adult | 14 | Categorical, Integer | 48842 | 2 |
| Covertype | 54 | Categorical, Integer | 581012 | 7 |
| Dota2 Games Results | 116 | Integer, Real | 102944 | 2 |
| Isolet | 617 | Real | 7797 | 26 |

**Experimental Setup:** We conducted all the experiments using a 2.30 GHz; Intel Core i3 Processor and RAM of 4 GB machine. We implemented both AdaBoost and Naïve Bayesian (NB) classifiers and our proposed hybrid algorithms (Algorithms 1 and 2) using Python 3.6. Several Python packages were used: Scikit Learn, Numpy, Pandas, Seaborn, Math, Itertools, Catplot and Matplotlib were used for plotting graphs. Besides, the soft-wares were used to run the program are Anaconda/Spyder and Microsoft Excel for the performance records.

Different symbols were used in measuring the classifier's performances such as follows:

**Table 2.** Meanings of performance evaluation symbols.

| **Symbols** | **Symbolic Meanings** |
|---|---|
| Accuracy | Percentage of assumptions which are correct |
| Precision | Percentage of "+" assumptions which are correct |
| F1-score | Harmonic mean of precision and sensitivity/recall |
| Sensitivity | Percentage of "+" instances (predicted as "+") |
| Specificity | Percentage of "-" instances (predicted as "-") |

**Results Evaluation:** At first, the performance of the proposed classifiers (Algorithms 1 and 2) against existing AdaBoost and NB classifiers was tested using the classification accuracy of the training sets of the 4 benchmark data sets shown in Table 3–6, summarizing the classification accuracy rates of the simple AdaBoost and NB classifiers, and our proposed hybrid algorithms for each of the 4 test data sets. We have used the weighted average to record the precision, f1-score, sensitivity/recall, specificity.

The results of the Table 3 indicates that despite all the algorithms are performing well in terms of accuracy rate but if we check the Specificity and Sensitivity for this Adult data-set we can find that both of them are almost 94% and 99% for the proposed algorithm2(NB+ADA) respectively.

**Table 3.** Classifier's performances on Adult Data-set.

| Algorithms | Accuracy (%) | Error Rate (%) | Precision | F1-score | Sensitivity | Specificity |
|---|---|---|---|---|---|---|
| AdaBoost | 87.65 | 12.35 | 0.87 | 0.88 | 0.68 | 0.94 |
| Naïve Bayes | 79.99 | 20.00 | 0.78 | 0.77 | 0.31 | 0.95 |
| Proposed Algorithm1 | 86.39 | 13.60 | 0.85 | 0.85 | 0.45 | 0.96 |
| Proposed Algorithm2 | 94.14 | 5.86 | 0.96 | 0.95 | 0.99 | 0.94 |

Hence, Table 4 illustrates all the four classifier's performance on the data-set named Covertype. It is visible that accuracy rates are poor due to the data-set imbalanced, still our proposed algorithm1(ADA+NB) is able to provide the 45.76% accuracy with 95% Specificity.

**Table 4.** Classifier's performances on Covertype Data-set.

| Algorithms | Accuracy (%) | Error Rate (%) | Precision | F1-score | Sensitivity | Specificity |
|---|---|---|---|---|---|---|
| AdaBoost | 44.29 | 55.70 | 0.59 | 0.48 | 0.52 | 0.79 |
| Naïve Bayes | 45.31 | 54.69 | 0.64 | 0.41 | 0.21 | 0.96 |
| Proposed Algorithm1 | 45.76 | 54.24 | 0.65 | 0.42 | 0.22 | 0.95 |
| Proposed Algorithm2 | 44.29 | 55.70 | 0.59 | 0.48 | 0.52 | 0.79 |

**Table 5.** Classifier's performances on Dota2 Games Result data-set.

| Algorithms | Accuracy (%) | Error Rate (%) | Precision | F1-score | Sensitivity | Specificity |
|---|---|---|---|---|---|---|
| AdaBoost | 58.82 | 41.18 | 0.59 | 0.59 | 0.66 | 0.51 |
| Naïve Bayes | 55.61 | 44.39 | 0.56 | 0.56 | 0.59 | 0.52 |
| Proposed Algorithm1 | 73.41 | 26.59 | 0.73 | 0.73 | 0.81 | 0.63 |
| Proposed Algorithm2 | 76.19 | 23.81 | 0.76 | 0.76 | 0.81 | 0.69 |

The performance of the Dota2 Games Results data-set from Table 5 states that our proposed algorithm2(NB+ADA) performs outstanding with less number of error rate of around 23% which proves our proposed algorithms strength over AdaBoost and NB classifiers for this data-set.

**Table 6.** Classifier's performances on Isolet Data-set.

| Algorithms | Accuracy (%) | Error Rate (%) | Precision | F1-score | Sensitivity | Specificity |
|---|---|---|---|---|---|---|
| AdaBoost | 28.89 | 71.10 | 0.30 | 0.22 | 1.00 | 0.00 |
| Naïve Bayes | 81.46 | 18.54 | 0.85 | 0.81 | 0.88 | 1.00 |
| Proposed Algorithm1 | 83.55 | 16.45 | 0.87 | 0.83 | 0.88 | 1.00 |
| Proposed Algorithm2 | 31.19 | 68.81 | 0.33 | 0.24 | 1.00 | 0.00 |

The data of the Table 6 indicates that if we consider higher Accuracy rate for the Isolet data-set Naïve Bayes (NB) classifier and our proposed algorithm1 (ADA+NB) performs well with the Accuracy rate of 81.46% and 83.54%. Even, if we compare the Specificity which is 100% for both of them but Sensitivity is much higher for NB. Now, if we compare the Error rate between the two classifiers then Proposed Algorithm1(ADA+NB) works better than classical NB classifier.

## 4. Graphical Comparison

In this section, we plotted the performance graph in Figure 6 for AdaBoost, Naïve Bayes (NB), Proposed Algorithm1 (ADA+NB) and Proposed Algorithm2 (NB+ ADA) respectively and compared their performances based on the Accuracy rate on the 4-real benchmark data-set that are discussed in the above sections.

### 4.1. Classifier's Performance Graphs

If we analyze the graph shown in Figure 6(a) the performance of AdaBoost classifier, we can see that despite of being such a strong classifier AdaBoost's performance for the Dota2 Games Result data-set and Isolet data-set denotes as orange and red color dots in the graph are not satisfactory. Performance for the Covertype data-set denotes as green dot is average due to the data-set imbalanced. But the Accuracy rate that we got for the Adult data-set represented as blue dot is around 90%. NB classifier's performance shown in Figure 6(b). The NB classifier induction shows that for the Adult and Isolet data-set it performs extraordinary as the Accuracy rate is almost 90% for the both data-sets, denotes as blue and red color dots accordingly and Covertype orange colored dot shows the worst performance accuracy rate due to imbalanced data-set.

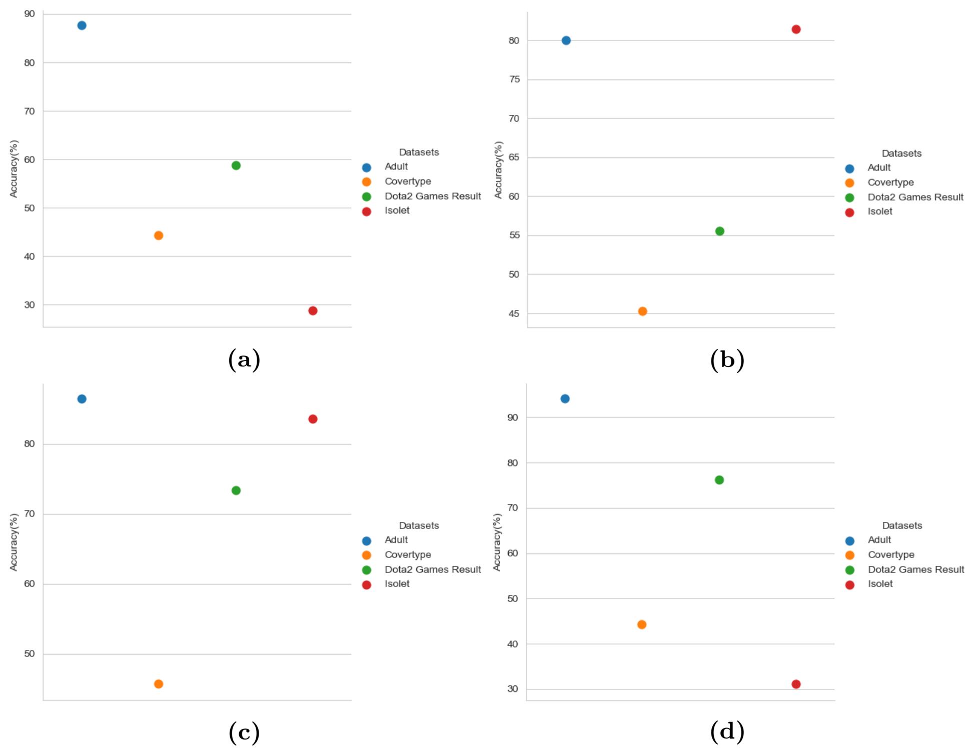


**Figure 6.** Classifier's Performance Comparison Graphs.

From the Figure 6(c) which illustrates proposed algorithm1(ADA+NB) classifier's performance, it's clearly visible that for Adult and Isolet data-set our proposed algorithm1 (ADA+NB) classifier provides Accuracy rate over 80%. But unfortunately for the Covertype data-set (orange color dot) it's unable to provide better classification Accuracy due to the data-set is very imbalance with higher in dimension too and AdaBoost trees are facing over-fitting problems. Figure 6(d) stands for the performance of proposed algorithm2(NB+ADA) classifier, visualizes the performance of our proposed algorithm2 (NB+ADA) classifier over our chosen data-sets. And it can be seen from the graph that except the Isolet data-set which is denoted by red dot, this classifier's performance is out ranging the other classifier's performances for all the chosen data-sets. Because, this classifier uses NB to reduce the noise and then uses the AdaBoost method to generate decision trees so that this decision trees DT do not fall into over-fitting.

## 4.2. Datasets Vs. Classifier's Performances

Earlier mentioned classifier's performances over chosen data-sets are shown in Figure 7. Based on Accuracy rate, it is proved that for the Adult & Dota2 Games Result data-set, our proposed classifier2 (NB+ADA) can be chosen, denoted as red.

Though the accuracy rate is almost the same for the Covertype data-set, therefore it is hard to define any specific classifier for that particular data-set but still our first hybrid classifier1 (ADA+NB) denoted as green, is better. Also, for the Isolet data-set we can use that classifier too.

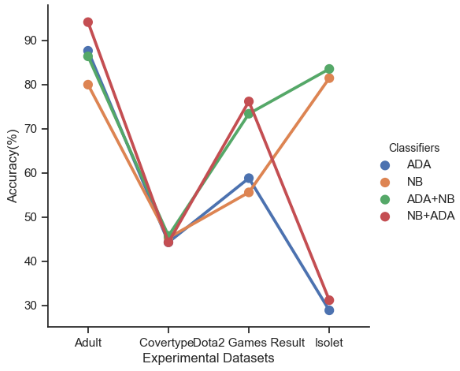


**Figure 7.** Classifier's Performance Comparison.

## 5. Conclusion

This research focuses on the establishment of our two newly proposed hybrid classifiers with regard to the improvements of the root classifier; Naïve Bayesian (NB) and Adaptive Boosting (AdaBoost). It is proved that our proposed classifiers have successfully increased the classification performance of AdaBoost and NB consequently when it comes to the accuracy rates. Technically, AdaBoost is one of the really strong Boosting algorithms. But it's a fact that in real life the data sets are not well organised. Most of the time it is full with noise or missing values or different behavioural instances. Due to these obstacles in the data, it is often seen that AdaBoost fails to provide its own capability when it comes to the accuracy rate. Hence, both of our proposed hybrid methods used NB classifier to help AdaBoost; so that we can save AdaBoost from over-fitting. We used NB classifier here because of no complexity and just one scan is enough for the training data-set as a result the computation gets fast. Another reason for choosing NB is that it has an amazing way of handling missing value when it calculates each class's similarities. In our first hybrid (ADA+NB) classifier, AdaBoost is used to remove the outliers from the training set and then proceeded comparatively clean data set for the Naive prediction. On the other hand, our second hybrid (NB+ADA) we used NB classifier to eradicate anomaly from the original data-set and passed the training examples to generate the decision trees (DTs) for AdaBoost model through preventing over-fitting. The performances of 2-classical approaches in machine learning, AdaBoost and NB and our two proposed hybrid classifiers were tested. Afterwards, we compared the experimental results of the four classifiers. The experiment proved that our newly proposed classifiers exceed the performances of other two textual approaches due to higher accuracy rates or lower error rates. We strongly believe that our proposed algorithms can be used in solving different classification problems in real life problem-domains. Forthcoming, we definitely want to extend our work by using some other classification techniques to remove the noise from the data such as Naive Bayes Trees or K-means clustering before the AdaBoost starts to generate its decision trees (DT).

**Acknowledgements.** The entire research was supported by the Janos Bolyai Research Scholarship of the Hungarian Academy of Sciences and the GINOP-2.2.1-18-2018-00012 supported by the European Union, co-financed by the European Social Fund. Therefore, we really feel privileged to get that intense support.

## References

[1] A. A. Afza, D. M. Farid, C. M. Rahman: *A Hybrid Classifier using Boosting, Clustering, and Naïve Bayesian Classifier*, World of Computer Science and Information Technology Journal (WCSIT) 1.3 (2011), pp. 105–109.

[2] S. Appavu alias Balamurugan, R. Rajaram: *Effective solution for unhandled exception in decision tree induction algorithms*, Expert Systems with Applications 36.10 (2009), pp. 12113–12119,
doi: `https://doi.org/10.1016/j.eswa.2009.03.072`.

[3] B. Aviad, G. Roy: *Classification by clustering decision tree-like classifier based on adjusted clusters*, Expert Systems With Applications 38 (2011), pp. 8220–8228.

[4] L. Breiman: *Arcing classifier (with discussion and a rejoinder by the author)*, Ann. Statist. 26.3 (June 1998), pp. 801–849,
doi: `10.1214/aos/1024691079`.

[5] T. Bujlow, M. Riaz, J. Pedersen: *A method for classification of network traffic based on C5.0 Machine Learning Algorithm*, in: 2012 International Conference on Computing, Networking and Communications (ICNC), Feb. 2012, pp. 237–241,
doi: `10.1109/ICCNC.2012.6167418`.

[6] K. M. A. Chai, H. L. Chieu, H. T. Ng: *Bayesian Online Classifiers for Text Classification and Filtering*, in: SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval, 2002, pp. 97–104,
doi: `10.1145/564376.564395`.

[7] B. Chandra, M. Gupta: *Robust approach for estimating probabilities in Naïve–Bayes Classifier for gene expression data*, Expert Systems with Applications 38.3 (2011), pp. 1293–1298,
doi: `https://doi.org/10.1016/j.eswa.2010.06.076`.

[8] B. Chandra, P. Paul Varghese: *Fuzzifying Gini Index based decision trees*, Expert Systems with Applications 36.4 (2009), pp. 8549–8559,
doi: `https://doi.org/10.1016/j.eswa.2008.10.053`.

[9] T. G. Dietterich: *An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization*, Machine Learning 40.2 (2000), pp. 139–157,
doi: `10.1023/a:1007607513941`.

[10] H. Drucker, C. Cortes: *Boosting Decision Trees.* In: Advances in Neural Information Processing Systems, vol. 8, Jan. 1995, pp. 479–485.

[11] L. Fan, K.-L. Poh, P. Zhou: *Partition-conditional ICA for Bayesian classification of microarray data*, Expert Systems with Applications 37.12 (2010), pp. 8188–8192,
doi: https://doi.org/10.1016/j.eswa.2010.05.068.

[12] D. M. Farid, G. M. Maruf, C. M. Rahman: *A new approach of Boosting using decision tree classifier for classifying noisy data*, in: 2013 International Conference on Informatics, Electronics and Vision (ICIEV), 2013, pp. 1–4,
doi: 10.1109/ICIEV.2013.6572718.

[13] D. Farid, M. Al-Mamun, B. Manderick, A. Nowe: *An adaptive rule-based classifier for mining big biological data*, Expert Systems with Applications 64 (2016), pp. 305–316,
doi: https://doi.org/10.1016/j.eswa.2016.08.008.

[14] D. M. Farid, L. Zhang, C. M. Rahman, M. Hossain, R. Strachan: *Hybrid decision tree and naïve Bayes classifiers for multi-class classification tasks*, Expert Systems with Applications 41.4, Part 2 (2014), pp. 1937–1946,
doi: https://doi.org/10.1016/j.eswa.2013.08.089.

[15] A. Franco-Arcega, J. Carrasco-Ochoa, G. Sánchez-Díaz, J. Martínez-Trinidad: *Decision tree induction using a fast splitting attribute selection for large datasets*, Expert Systems with Applications 38.11 (2011), pp. 14290–14300,
doi: https://doi.org/10.1016/j.eswa.2011.05.087.

[16] Y. Freund, R. E. Schapire: *A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*, Journal of Computer and System Sciences 55.1 (1997), pp. 119–139,
doi: https://doi.org/10.1006/jcss.1997.1504.

[17] A. J. Grove, D. Schuurmans: *Boosting in the Limit: Maximizing the Margin of Learned Ensembles*, in: AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, 1998, pp. 692–699.

[18] C.-C. Hsu, Y.-P. Huang, K.-W. Chang: *Extended Naive Bayes classifier for mixed data*, Expert Systems with Applications 35.3 (2008), pp. 1080–1083,
doi: https://doi.org/10.1016/j.eswa.2007.08.031.

[19] X. Li, L. Wang, E. Sung: *AdaBoost with SVM-based component classifiers*, Engineering Applications of Artificial Intelligence 21.5 (2008), pp. 785–795,
doi: https://doi.org/10.1016/j.engappai.2007.07.001.

[20] J. McHugh: *Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory*, Association for Computing Machinery 3.4 (2000), pp. 262–294,
doi: 10.1145/382912.382923.

[21] K. Polat, S. Güneş: *A novel hybrid intelligent method based on C4.5 decision tree classifier and one-against-all approach for multi-class classification problems*, Expert Systems with Applications 36.2, Part 1 (2009), pp. 1587–1592,
doi: https://doi.org/10.1016/j.eswa.2007.11.051.

[22] J. R. Quinlan: *Bagging, Boosting, and C4.S*, in: AAAI Press, 1996, pp. 725–730.

[23] R. E. Schapire: *The Boosting Approach to Machine Learning: An Overview*, in: Denison D.D., Hansen M.H., Holmes C.C., Mallick B., Yu B. (eds) Nonlinear Estimation and Classification. Lecture Notes in Statistics, vol 171. Springer, New York, NY. 2003, pp. 149–171,
doi: https://doi.org/10.1007/978-0-387-21579-2_9.

[24] H. Schwenk, Y. Bengio: *Boosting Neural Networks*, Neural Computation 12.8 (2000), pp. 1869–1887,
doi: 10.1162/089976600300015178.

[25] M. Tavallaee, E. Bagheri, W. Lu, A. Ghorbani: *A detailed analysis of the KDD CUP 99 data set*, 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications (2009), pp. 1–6.