Improvement of the Fine tuning algorithm

Joseph Mietkiewicz[1,2] and Anders L. Madsen[2,3]

[1] Technological University of Dublin, Ireland
[2] Hugin Expert A/S, Denmark
[3] Aalborg University, Denmark
D21127042@mytudublin.ie anders@hugin.com

Abstract. Khalil El Hindi has developed a fine-tuning algorithm to improve the classification accuracy of the Naive Bayes. His algorithm optimizes the conditional probability tables of the Naive Bayes after the training phase. The values of the probabilities of a variable are modified if it causes misclassification of a training instance. The algorithm outperforms in many cases the Naive Bayes. We analyze the performance of the algorithm, discussed its issues, and compare it to a modified algorithm. The new algorithm simplifies the formula used in the fine-tuning algorithm and uses a more efficient scoring metric, the Brier score, to fine-tune the probabilities. The new algorithm shows an improvement in terms of classification accuracy on benchmark data sets compared to the Naive Bayes and fine tuned Naive Bayes.

Keywords: Fine tuning · Naives Bayes · Classification

1 Introduction

Khalil El Hindi has developed a fine-tuning algorithm (FTNB)[1] to improve the Naive Bayes (NB) classifier. His algorithm adjusts the probability of the conditional probability table (CPT) to increase the accuracy of the model in the training set. This optimization outperforms Naive Bayes in many cases. He has extended his algorithm in different ways. He has built fine-tuning for Tree augmented Naive Bayes (FTTAN)[2]. It extends the algorithm for TAN and works in the same way as fine tuning. To increase the performance of FTTAN he developed, selective fine-tuning for NB (SFTNB) and TAN (SFTTAN)[3]. He concludes that SFTTAN works better for TAN but NB outperforms SFTNB. He has also used a differential evolution algorithm to fine-tune Naive Bayes [4]. He has more recently developed lazy fine-tuning algorithms GLFTNB and LFTNB[5]. In this paper, he concluded that the LFTNB is the best algorithm. He also developed other algorithms more specific for text classification that are beyond the scope of this paper [5]. Attribute weights are also used to improve the classification accuracy of NB. Zhang combined attribute weight and FTNB to build a more accurate model [6]. We focus here on the FTNB algorithm. Our goal here is to optimize the fine-tuning algorithm. The FTBN is studied and

we introduce a modification to it to improve and simplifie the algorithm. The remainder of the paper is organized as follows: Section 2 introduces the background necessary and the FTNB algorithm, section 3 introduces its limitations of the FTNB algorithm, and section 4 introduces our novel algorithm.

2 Background

2.1 Bayesian Network

A Bayesian network is a direct acyclic graph where the nodes represent variables and the arcs represent causal interactions between these variables. It is a powerful tool for modeling causal interaction and predicting events [7]. A Bayesian network is defined as follows:

Definition (Jensen and Nielsen) [8]: A discrete Bayesian network $\mathcal{N} = (\mathcal{A}, \mathcal{G}, \mathcal{P})$ consist of

- A DAG $\mathcal{G} = (V, E)$ with nodes $V = \{v_1, ..., v_n\}$ and directed links E.
- A set of discrete random variables, $\mathcal{X}$, represented by the nodes of $\mathcal{G}$.
- A set of conditional probability distributions, $\mathcal{P}$, containing one distribution $P(X_v|X_{pa(v)})$, for each random variable $X_v \in \mathcal{X}$.

A Bayesian network encodes a joint probability distribution over a set of random variables, $\mathcal{X}$, of a problem domain. The set of a conditional probability distributions, $\mathcal{P}$, specifies a multiplicative factorization of the joint probability distribution over $\mathcal{X}$.

$$P(\mathcal{X}) = \prod_{(v \in V)} P(X_v|X_{pa(v)}) \tag{1}$$

2.2 Naives Bayes

The Naive Bayes classifier is here studied. It has a strong assumption of conditional independence between all variables given the class variable. Let C be the class variable to predict and $X_1, ..., X_n$ the feature variables. Given the Naive Bayes assumption the joint probability distribution over the variables can be written as:

$$P(\mathcal{X}) = P(C) \prod_{i=1}^{n} P(X_i|C) \tag{2}$$

Let $(a_1, ..., a_n)$ be an instance to classify and C the possible classes. Using the Independence assumption and Bayes formula, the Naive Bayes predicts the class of this instance according to:

$$\begin{aligned} c_{\text{predicted}} &= \underset{c \in C}{\operatorname{argmax}} P(c|a_1, ..., a_n) \\ &= \underset{c \in C}{\operatorname{argmax}} \frac{P(c)\prod_{i=1}^{n} P(a_i|c)}{P(a_1, ..., a_n)} \\ &= \underset{c \in C}{\operatorname{argmax}} \frac{P(c)\prod_{i=1}^{n} P(a_i|c)}{\sum_{c \in C}\prod_{i=1}^{n} P(a_i|c)P(c)} \end{aligned} \tag{3}$$

The performance of Naive Bayes relies on a good estimate of the conditional probabilities $P(a_i|c)$ and $P(c)$. Those probabilities are estimated on the data during the training phase of the model. The idea of fine-tuning is to modify those probabilities to increase the accuracy of the model after training. The fine-tuning algorithm is defined in the next section.

2.3 Fine tuning algorithm

The idea of fine-tuning is to increase the accuracy of the model on the training set as described in [1]. For each misclassified instance, meaning that the model predicted the class $c_{predicted}$ instead of c_{actual}, the terms involved in the prediction of $c_{predicted}$ are reduced and those involved in c_{actual} are increased (with $c_{predicted} \neq c_{actual}$). For each misclassified instance the procedure described below is applied.

Let $a = (a_1, ..., a_n)$ be a misclassified instance classified as $c_{predict}$ instead of c_{actual}. The update of the probabilities for each a_i is defined as:

$$P(a_i|c_{actual}) = P(a_i|c_{actual}) + \delta(a_i, c_{actual})$$
$$P(a_i|c_{predicted}) = P(a_i|c_{predicted}) - \delta(a_i, c_{predicted})$$

with

$$\delta(a_i, c_{actual}) = \eta(\alpha \times \max_{ai} P(a_i|c_{actual}) - P(a_i|c_{actual}))\epsilon$$
$$\delta(a_i, c_{predicted}) = \eta(\beta \times P(a_i|c_{predicted}) - \min_{a_i} P(a_i|c_{predicted}))\epsilon$$

The error rate, ϵ, is defined as:

$$\epsilon = |P(c_{\text{actual}}|a) - P(c_{\text{predicted}}|a)|$$

The coefficients α and β are constant larger or equal to 1 which controls the size of the update. Those constants are set at 2 according to [1] (this value gives a better result than the value 1 as advised in [5]). η is the learning rate lower than 1 set at 0.01 according to [1]. The missing values are ignored. The procedure is applied for each misclassified instance. If the training accuracy increase after a modification of the conditional probability distributions, the fine-tuning continues. Algorithm 1 describes the steps of the FTNB algorithm.

Algorithm 1 FTNB

Inputs:Naive Bayes, Training data
Output: Fine Tune Naive Bayes
Train the model on the training data
while training classification accuracy improves **do**
  Save the model
  for each training instance, inst, **do**
    if $c_{predicted} \neq c_{actual}$ **then**
      for for each attribute value, a_i, of inst **do**
        $p(a_i|c_{actual})$=$p(a_i|c_{actual})$+$\delta(a_i, c_{actual})$
        $p(a_i|c_{predicted})$=$p(a_i|c_{predicted})$-$\delta(a_i, c_{predicted})$
      end for
    end if
  end for
  Compute accuracy
end while
return Model

3 Issue with the fine tuning algorithm

The major problem with FTNB is that it uses accuracy to decide whether or not to continue fine-tuning. In case the accuracy stays the same after a fine-tuning epoch, it is not clear if it is wise to continue to fine-tune or not. Moreover small drops of the training accuracy can occur causing the termination of the algorithm before reaching the global maximum. A more precise scoring metric is needed to measure the impact of the fine-tuning after each epoch. The above consideration is illustrated with the typical following example.

To illustrate the issue with the FTNB algorithm we have run the algorithm on the Parkison data set [10]. The algorithm is run with 50 epochs and the accuracy of the test set is calculated for each number of epochs. Figure 1 shows the training accuracy and testing accuracy for 50 epochs on the Parkinson data. The figure shows when FTNB stops and what is the actual best number of epochs. We can observe that fine-tuning more can increase the accuracy even if it decreases or keeps at the same value the training accuracy for a couple of epochs. Moreover, the training accuracy curve is very sharp. Small decreases are very frequent which causes the early termination of the algorithm.

The Brier score can be used to overcome this problem. The Brier score calculates the performance of the model regarding the different probabilities of the different classes for each prediction. Dealing with the value of the probabilities allow more fine tuning steps. The use of the Brier score is described in the next section.

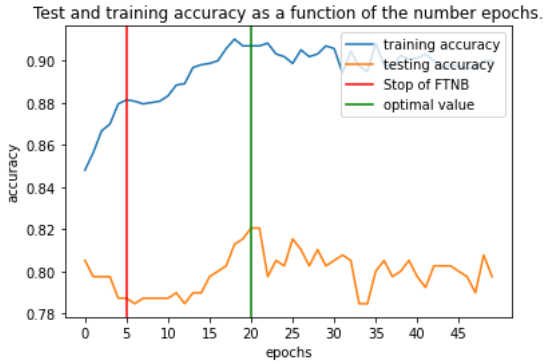


Fig. 1: Testing and training accuracy of the FTNB on the UCI Parkinson data set as a function of the number of epochs. The red line indicats the termination of the FTNB algorithm according to the treditional stopping criterion and the green line indicats the best value possible.

4 Fine tuning using Brier score

4.1 Brier score

The Brier score is a proper scoring function that asses the classification performance of a model. Let n be the number of instances to classify and r the number of classes. The Brier score is defined as follows [9]:

$$P = \frac{1}{n}\sum_{k=1}^{n}\sum_{l=1}^{r}(P(c_l|a_k) - E_{kl})^2 \tag{4}$$

Where E_{kl} has the value 1 or 0 according to whether the class of the instance k is the class l or not. Reducing the Brier score will tend to more sharp predictions. In optimizing the Brier score, for each instance, the probability of the predicted class will tend to be one and the others 0. The score ranges from 0 to 2, 0 being the optimal score. Below is an example of the calculation of the Brier score:

Example:

Table 1: Example of 2 instances x_1 and x_2 to be classified in 3 classes with there respective probabilities

	C_1	C_2	C_3	True class
x_1	0.7	0.2	0.1	C_1
x_2	0.2	0.5	0.4	C_3

The Brier score of Table 1 is calculated as:

$$B = \frac{1}{2}\Big(\big[(1-0.7)^2 + 0.2^2 + 0.1^2\big] + \big[0.2^2 + 0.5^2 + (1-0.4)^2\big]\Big) = 0.1975 \quad (5)$$

This score can be a good candidate for fine-tuning because it will take into account any increase in the performance of the model even if the increase is very small. The application of the Brier score for fine-tuning is explained in the next section.

4.2 Fine tuning with Brier score

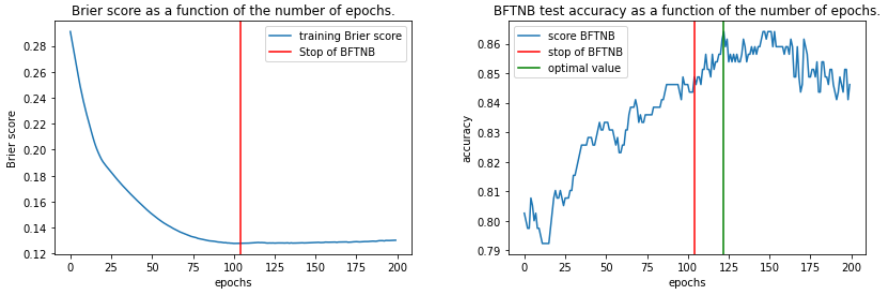
The advantage of the Brier score for fine-tuning is that even in case of a small increase in the classification of the model the algorithm will continue. In the original FTNB if the accuracy does not increase the algorithm stops. Using Brier allows small steps that don't increase the accuracy immediately but increase the Brier score. The model can therefore be fine tuned more slowly and accurately. This comes at the cost of a greater number of iterations compared to FTNB.

The idea of the FTNB algorithm is kept but the formula is reduced to only the learning rate η. Each node is equally fine-tuned by only the η parameter. It results in a simpler algorithm that more accurately classifies benchmark data sets than the original fine-tuning algorithm. In case of the probabilities go below 0 after subtracting η we don't modify the probabilities value.

Example: We illustrate the use of the Brier score with the typical same example as in the previous section on the Parkison dataset. Figure 2 shows the Brier score on the training data for 200 epochs. The Brier score curve is smooth. The global minima of the curve is found by the algorithm. On the other hand, the FTBN algorithm finds only the local maxima. We can observe in Figure 2 the number of epochs where the algorithm stops and the optimal value of the number of epochs. The algorithm still stops earlier but the testing accuracy can reach a much higher maximal value than with the FTBN algorithm.

5 Experimental results and analysis

The experiment was performed using datasets similar to the one used in the paper by El Kahindi. Those datasets come from the UCI repository [10]. The datasets are discretized if needed using the minimization entropy algorithm. Eta=0.001 was used for BFTNB. This value provides the best result. Stratified shuffle split cross-validation was used with a test set of 20% and 100 folds. A two-tailed-t-test with a 95% confidence interval was used. Random priors were used for each fold. Laplace smoothing was used for the estimation of the conditional probability distributions. The code of the algorithm and the data used are available at the git repository `https://github.com/Jomietk/Brier_FTNB`. The code is developed using HUGIN software API [11]. The results of the algorithms are shown in the tables below.



(a) (b)

Fig. 2: Training Brier score of the BFTNB (a) and testing accuracy (b) on the UCI Parkinson data set as a function of the number of epochs. The red line indicates the stop of the BFTNB algorithm according to the stopping criterion. The green line indicates the best value.

Algorithm 2 BFTNB

Inputs:Naive Bayes, training data
Output: Fine Tune Naive Bayes
Train the model on the training data
while Brier score improves **do**
  Save the model
  for each training instance, inst, **do**
    if $c_{predicted} \neq c_{actual}$ **then**
      for for each attribute value, a_i, of inst **do**
        $p(a_i|c_{actual})$=$p_t(a_i|c_{actual})$+η
        $p(a_i|c_{predicted})$=$p_t(a_i|c_{predicted})$-η
      end for
    end if
  end for
  Compute Brier score
end while
return Model

The average accuracies across all BFTNB, FTNB, and NB data sets are 81.1, 80.6, and 79.7 respectively. It shows that BFTNB outperforms on average both algorithms. BFTNB works better and on more data sets than FTNB. But when the algorithm does not work, it tends to give lower accuracy than NB, while FTNB tends to give the same result as NB. An interesting consideration that is not shown in the tables is the cases where the fine-tuning algorithms work better than the other algorithm and that it works better than the NB algorithm. This avoids considering the case where NB is better, in which case both algorithms are not useful and it is less meaningful to compare them. BFTNB performs significantly better than FTNB and NB in 16 datasets. FTNB performs significantly better than BFTNB and NB only in 5 datasets. They are also equally significantly better than NB in 3 datasets. However, the better performance of FTNB comes at the price of more fine-tuning epochs. The average number of epochs for BFTNB is 44.8 compared to FTNB 1.2. This is due to the smaller steps allowed by the use of the Brier score.

We compared the algorithms using the AUC (one versus one with weighted averages) in Table 4. The results are even better. If we consider only the cases when the fine-tuning algorithms are better than Naive Bayes. BFTNB works better than FTNB and NB in 15 cases. FTNB works better than BFTNB and NB only in one case. This is due to the Brier score improving the probabilities of the prediction and the AUC not being dependent on the threshold value for classification. The average AUC of BFTNB, FTNB, and NB is 87.0, 86.3,86.4 respectively. We use the Brier score with the original formula but it didn't give successful outcomes. But the global minima with the training data was easier to find as in Figure 2(a). It was tested for eta=0.01,0.001 beta=alpha=1,2. For each combination, it gave the same result as NB or worst.

6 Conclusion

FTNB augments the training phase with fine-tuning to increase the accuracy of the training data. This algorithm outperforms the Naïve Bayes algorithm in most cases. We have analyzed this algorithm and discussed its problems. It can stop too early and the terms used in the formula are relatively arbitrary. We have developed a simpler algorithm using the Brier score. We empirically evaluate the performance of both algorithms and find that our algorithm outperforms FTNB in many cases. The Future research will be on finding a more performing formula to fine tune the probabilities distribution.

Acknowledgements This work has been done within the collaborative intelligence for safety Critical systems project (CISC). The CISC project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie grant agreement no. 955901.

Table 2: Comparison of accuracy between NB, FTBN and BFTBN

	NB VS FTNB		NB VS BFTNB		FTNB VS BFTNB	
Interval	NB	FTNB	NB	BFTNB	FTNB	BFTNB
anneal	95.8	**96.5**	95.8	**96.9**	96.5	**96.9**
arrhythmia	70.3	**70.9**	**70.3**	69.2	**70.9**	69.2
balance	90.8	**91.1**	90.8	**92.9**	91.1	**92.9**
breast cancer	77	77.5	77	77	77.5	77
bridges V2	59.7	59.7	**59.7**	58.8	**59.7**	58.8
bridges V1	62	62	62	62.2	62	62.2
car	85.7	**89.3**	85.7	85.7	**89.3**	85.7
cmc	50.7	51.1	50.7	**52.1**	51.1	**52.1**
contact lenses	72.6	72.6	72.6	72.8	72.6	72.8
credit g	74.3	74.3	74.3	**74.9**	74.3	**74.9**
dermatology	97.8	97.8	97.8	97.7	97.8	97.7
diabetes	75.2	75.3	75.2	75.3	75.3	75.3
flare	85.3	**87.2**	85.3	**87.1**	87.2	87.1
glass	70.9	**71.1**	70.9	70.8	**71.1**	70.8
haberman	78.6	78.6	78.6	78.5	78.6	78.5
hayes roth	79.9	79.9	79.9	**82.2**	79.9	**82.2**
hepatitis	82.7	82.7	82.7	82.8	82.7	82.8
horse colic	85.6	85.6	85.6	85.6	85.6	85.6
hypothyroid	98.7	**98.8**	98.7	**99.1**	98.8	**99.1**
ionosphere	86	85.9	86	86.2	85.9	86.2
krvskp	87.8	**96.2**	87.8	**95**	**96.2**	95
labor	87.3	87.4	87.3	**88.2**	87.4	**88.2**
letter recognition	75	75.1	75	**81.8**	75.1	**81.8**
lymphography	84.3	84.2	84.2	84.4	84.2	84.4
monks 1	74.9	74.9	74.9	74.8	74.9	74.8
monks 2	62.6	**65.4**	62.6	62.7	**65.4**	62.7
monks 3	96.5	96.5	96.5	96.5	96.5	96.5
mushrooms	99	**99.9**	99	**99.8**	99.9	99.8
nursery	90.4	90.4	90.4	90.4	90.4	90.4
parkinsons	79.3	**79.9**	79.3	**82.4**	79.9	**82.4**
primary tumor	48.3	48.4	**48.3**	47	**48.4**	47
segment	92.5	92.5	92.5	**94.2**	92.5	**94.2**
seismic bumps	82.1	**92.9**	82.1	**92.8**	92.9	92.8
solar flare 1	69.8	69.8	**69.8**	67.8	**69.8**	67.9
solar flare 2	66.9	67	66.9	66.4	66.9	66.4
sonar	74.9	74.9	**74.9**	73.9	**74.9**	73.9
soybean	91.6	91.6	**91.6**	89.7	**91.6**	89.7
spambase	92.8	**94.8**	92.8	**95.4**	94.8	**95.4**
splice	**95.5**	95.2	95.5	95.4	95.2	**95.4**
tae	56.7	56.7	56.7	**62.5**	56.7	**62.5**
tic-tac-toe	70.2	**73.6**	70.2	**79.4**	73.6	**79.4**
transfusion	74.6	**75.2**	74.6	**75.7**	75.2	**75.7**
Vehicle	51.5	51.5	51.5	**59.6**	51.5	**59.6**
vote	90.1	90.1	90.1	**94.1**	90.1	**94.1**
waveform500	80.2	**81**	80.2	**84**	81	**84**
wine	98.1	98.1	**98.1**	97.7	**98.1**	97.7
zoo	95.5	95.5	95.5	95.5	95.5	95.5
Average	79.7	80.6	79.7	81.1	80.6	81.1
#Better	19	28	19	28	23	24
#Sig Better	1	16	7	21	10	18

Table 3: Comparison of the AUC between NB, FTBN and BFTBN

	NB VS FTNB		NB VS BFTNB		FTNB VS BFTNB	
Interval	NB	FTNB	NB	BFTNB	FTNB	BFTNB
anneal	98.3	**98.4**	98.3	**98.5**	98.4	**98.5**
balance	86.6	**86.8**	86.6	**86.8**	86.8	**93**
breast cancer	**69.7**	65.5	**69.7**	65	**65.5**	65
bridges V2	83.9	83.9	83.9	83.8	83.9	83.8
Bridge V1	84.7	84.7	84.7	84.6	84.7	84.6
car	97.3	**98.2**	97.3	97.3	**98.2**	97.3
cmc	**70.4**	68.9	70.4	**70.7**	68.9	**70.7**
contact lenses	91.6	91.6	91.6	91.3	91.6	91.3
credit g	**77.7**	76.9	77.7	**77.8**	76.9	**77.8**
dermatology	99.9	99.9	99.9	99.9	99.9	99.9
diabetes	80.7	80.7	80.7	80.7	80.7	80.7
flare	**67.2**	65.5	**67.2**	65.1	65.5	65.1
glass	88.7	88.7	**88.7**	88.4	88.7	88.7
haberman	64.9	64.9	64.9	64.9	64.9	64.9
hayes roth	92.9	92.9	92.9	**95.2**	92.9	**95.2**
Hepatitis	83.8	83.7	**83.8**	82.4	**83.7**	82.4
horse colic	76.3	76.3	76.3	76.3	76.3	76.3
ionosphere	91.6	91.7	91.6	91.3	91.7	91.3
krvskp	95.2	**99.4**	95.2	**99.1**	**99.4**	99.1
labor	96.4	96.4	**96.4**	95.7	**96.4**	95.7
letter recognition	98.1	**98.1**	98.1	**98.9**	98.1	**98.9**
monk 1	71.6	71.6	71.6	71.6	71.6	71.6
monk 2	53.6	53.8	53.6	53.6	53.8	53.6
monk 3	98.4	98.4	98.4	98.4	98.4	98.4
mushrooms	99.9	**1**	99.9	**1**	1	1
segment	99.2	99.2	99.2	**99.5**	99.2	**99.5**
seismic bumps	**77.8**	74.1	**77.8**	77.4	74.	**77.4**
solar flar 1	87.4	87.4	87.4	**87.8**	87.4	**87.8**
solar flar 2	87.5	87.4	87.5	**87.8**	87.4	**87.8**
sonar	**85**	83.8	**85**	83.8	**85**	83.8
soybean	99.6	99.6	99.6	99.6	99.6	99.6
spambase	96.6	**97.5**	96.6	**97.9**	97.5	**97.9**
splice	**99.4**	99.4	**99.4**	99.3	**99.4**	99.3
tae	77.6	77.6	77.6	**78.5**	77.6	**78.5**
tic-tac-toe	74.6	**75.8**	74.6	**86.8**	75.8	**86.8**
transfusion	69.4	**70.3**	69.4	**72**	70.3	**72**
vehicle	76.5	76.5	76.5	**80**	76.5	**80**
vote	96.8	**96.9**	96.8	**98.4**	96.9	**98.4**
waveform 5000	95.4	**95.8**	95.4	**96.2**	95.8	**96.2**
wine	99.9	99.9	99.9	99.9	99.9	99.9
zoo	99.8	99.8	99.8	99.8	99.8	99.8
Average	79.7	80.6	79.7	81.1	80.6	81.1
#Better	11	15	18	22	21	20
#Sig Better	7	11	8	18	7	17

Table 4: Number of epoch of FTBN and BFTNB

data	FTNB	BFTNB	data	FTNB	BFTNB
anneal	2.8	25.1	monks 1	0	0.0
arrhythmia	1.6	35	monks 2	1.7	0.3
balance	0.8	31.0	monks 3	0	0
breast cancer	1.2	8.3	mushrooms	2.7	12.3
bridges V2	0.1	79.3	nursery	0	0
bridges V1	0.1	104.9	parkinson	2.4	79.5
car	2.0	0	primary tumor	0.8	41.61
cmc	2.0	1	segment	0.0	21.6
contact lenses	0.1	34.8	seismic bumps	1.2	2.0
credit g	0.6	1.7	solar flare 1	0.6	6.7
dermatology	0.05	508.23	solar flare 2	0.6	7.7
diabetes	0.7	2.2	sonar	0.3	49.4
flare	1.6	11.6	soybean	0.0	103.1
glass	0.2	12.9	spambase	2.0	10.2
haberman	0.0	0.0	splice	2.0	11.9
hayes roth	0.0	3.4	tae	0.0	416.8
hepatitis	0.7	49.5	tic-tac-toe	1	6.0
horse colic	0	1.03	transfusion	0.7	3.7
hypothyroid	1.1	5.0	vehicle	0.0	51.3
ionosphere	0.3	101.5	vote	0.2	43.0
krvskp	10.8	2.0	waveform 5000	9.2	6.1
labor	0.0	121.6	wine	0.0	40.3
letter recognition	5.2	6.0	zoo	0	0
lymphography	0.6	48.4			
Average	1.2	44.8	Median	0.57	11.58

References

1. EL HINDI, Khalil. Fine tuning the Naïve Bayesian learning algorithm. AI Communications, 2014, vol. 27, no 2, p. 133-141.
2. ALHUSSAN, Amel et EL HINDI, Khalil. Fine tuning the tree augmented Naïve Bayes (FTTAN) learning algorithm. In : 2015 SAI Intelligent Systems Conference (IntelliSys). IEEE, 2015. p. 72-79.
3. ALHUSSAN, Amel et EL HINDI, Khalil. Selectively fine-tuning Bayesian network learning algorithm. International Journal of Pattern Recognition and Artificial Intelligence, 2016, vol. 30, no 08, p. 1651005.
4. DIAB, Diab M. et EL HINDI, Khalil M. Using differential evolution for fine tuning naïve Bayesian classifiers and its application for text classification. Applied Soft Computing, 2017, vol. 54, p. 183-199.
5. EL HINDI, Khalil M., ALJULAIDAN, Reem R., et ALSALMAN, Hussien. Lazy fine-tuning algorithms for naïve Bayesian text classification. Applied Soft Computing, 2020, vol. 96, p. 106652.
6. ZHANG, Huan; JIANG, Liangxiao. Fine tuning attribute weighted Naive Bayes. Neurocomputing, 2022, 488: 402-411.
7. KJAERULFF, Uffe B. et MADSEN, Anders L. Bayesian networks and influence diagrams. Springer Science+ Business Media, 2008, vol. 200, p. 114.
8. JENSEN, Finn V. et NIELSEN, Thomas Dyhre. Bayesian networks and decision graphs. New York : Springer, 2007.
9. Brier, Glenn W., et al. Verification of forecasts expressed in terms of probability. Monthly weather review, 1950, vol. 78, no 1, p. 1-3.
10. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
11. HUGIN EXPERT A/S, HUGIN sofware [https://www.hugin.com]