OPEN ACCESS
Saudi Journal of Engineering and Technology
Abbreviated Key Title: Saudi J Eng Technol
ISSN 2415-6272 (Print) |ISSN 2415-6264 (Online)
Scholars Middle East Publishers, Dubai, United Arab Emirates
Journal homepage: https://saudijournals.com

Review Article

The Philosophy of Naive Bayes and its Comparison with the Tree Augmented Naive Bayes (TAN) in Making Predictions (Case Study Using Course Student Data)

Aslan Alwi[1,2*], Isdaryanto Iskandar[1], Djoko Setyanto[1]

[1]Faculty of Engineering, Atma Jaya Chatolic University of Indonesia, Indonesia
[2]Faculty of Engineering, Universitas Muhammadiyah Ponorogo, Indonesia

DOI: 10.36348/sjet.2022.v07i07.005 | **Received:** 06.07.2022 | **Accepted:** 14.08.2022 | **Published:** 17.08.2022

***Corresponding author:** Aslan Alwi
Faculty of Engineering, Atma Jaya Chatolic University of Indonesia, Indonesia

Abstract

This paper aims to present a comparison between the Naive Bayes model and the Tree Augmented Naive Bayes (TAN). This comparison is preceded by a philosophical explanation of the two in the author's way of understanding. Then this comparison was carried out in the case of making a predictive model for the final exam scores of students at the Muhammadiyah University of Ponorogo. In this case, we apply two different models, namely Naive Bayes and Tree Augmented Naive Bayes (TAN) to predict learning outcomes before ending at the end of the semester in terms of lecturers' assessments of students. When the course is in progress, the lecturer needs to continuously evaluate students' understanding of the subject matter being taught. This is so that lecturers can immediately anticipate learning problems in class. Calculations with these two models use only R language with jupyter notebook interface. Validation and testing of the two models used a case dataset in 4 classes of language theory and automata course students even semester 2017-2018 at the Department of Informatics Engineering, Faculty of Engineering, and University of Muhammadiyah Ponorogo with a dataset size of 99 notes (99 students). For the validation and model testing methods, k- fold and hold-out cross-validation are used. Each model is validated and tested with the same k-fold method scheme.

Keywords: Naive Bayes, Bayes Classifier, Tree Augmented Naive Bayes, R Language, Student final exams, Prediction, k-fold cross validation**,** hold-out cross validation.

1. INTRODUCTION

Bayesian reasoning is a tool that is one of the pillars in artificial intelligence and machine learning construction. This concept introduces one of the most widely used probabilistic machine learning methods. Therefore, a deep understanding of this concept is a must, so that broad enlightenment and imagination can be obtained to apply this in various engineering fields, especially artificial intelligent engineering or machine learning. This paper presents a philosophical exploration of the origin of the idea of naive nayes and tree augmented naive bayes as a form of approximation of Bayesian reasoning. Although there have been many writings that present the derivation of this idea from Bayes' theorem, this paper tries to provide a comprehensive and detailed explanation of how this idea is constructed by people and then applied to various life problems. This paper also presents a comparison of the application of the naive bayes method and the augmented naive bayes tree method using the k-fold testing model verification method and hold- out cross validation. Deployment using the R language interface. This application is carried out in a case study to make a prediction model for end-of-semester exam scores for automata courses at the Muhammadiyah University of Ponorogo.

This paper only explores the philosophy of TAN and NB as an idea that simplifies the computation of Bayesian reasoning, making it a fast tool to use in creating or engineering artificial intelligence machines or machine learning that performs probabilytic reasoning. Then compare the two in modeling.

1.1. Naive Bayes Philosophy

Naive Bayes is used by people to filter emails whether they are spam or not. Like research (Marsono *et al.*, 2006) which implements naive Bayes classifier

on hardware that filters incoming emails. Naive Bayes is also done by (Das & Kolya, 2017) which uses the algorithm for text mining and sentiment analysis of twitter statuses. Naive Bayes is also used by (Hsu *et al.*, 2016) for image classification, or by (Nirmala *et al.*, 2017) to detect glaucoma in patients.

Naive Bayes method is one of the forms of the Bayes classifier. Bayes classifier is a method of classifying using the Bayes theorem. The Bayes theorem has the following basic forms as stated by formula (1).

$$P(Y|X) = \frac{P(X|Y).P(Y)}{P(X)} \quad (1)$$

The probability of P (Y | X) is the posterior probability, P(X|Y) is the conditional probability, P (Y) is the prior probability and P (X) evidence probability.

The formula (1) can be written more broadly and specifically by expanding the variable X to X_1, X_2, X_3, ..., X_n, where each *i*-variable, X_i is an explanatory variable or variable that is intended to be used for classification or prediction. The Y variable is called the class variable or target variable because the value of the Y variable becomes the classes that classify the dataset.

In general, the formula (1) can be written in the form X_1, X_2, X_3, ..., X_n, and Y as follows: $P(Y|X_1,X_2,X_3,...,X_n) = P(X_1,X_2,X_3,...,X_n|\ Y).P(Y) / P(X_1,X_2,X_3,...,X_n)$. By taking the maximum probability value from the posterior, the classification is done. For example, a record with attribute values (X_1, X_2, X_3, ..., X_n) then the record can be classified into a Y value that has a maximum posterior probability. That is the maximum P (Y | X_1, X_2, X_3, ..., X_n).

This classification process can be written in the following formula:

$$\text{argmax } P(Y|X_1,X_2,X_3,...,X_n) = \text{argmax } P(X_1,X_2,X_3,...,X_n|Y).P(Y)/P(X_1,X_2,X_3,...,X_n) \quad (2)$$

But to calculate the probability P (Y | X1, X2, X3, ..., Xn) requires a combination of counts for P (X_1, X_2, X_3, ..., X_n | Y). In general, the number of possible probabilities for calculating P(X_1, X_2, X_3, ..., X_n|Y) if the value of each variable there are only two possible values, we get $2.2^{(n-1)}$ calculation. If there are *k* possible values, the number of computational possibilities is $k.k^{(n-1)}$. But if we use the assumption that each variable in X_1, X_2, X_3, ..., X_n, Y is independent of each other (independent relations or naive relationships, so-called naive), then the number of computational possibilities for calculating the probability of P(X_1, X_2, X_3, ..., X_n |Y) is just 2*n*. If there are *k* possible values for each X_i then the number of possible computational combinations is *k.n*.

The following is an argument about why the amount of computation is so. Suppose that for each X_i it has the possibility of u_i and non-u_i values (including Y, i.e. u_y and non-u_y), ie there are only two possible values for each variable. If so, the number of combinations P(X_1, X_2, X_3, ..., X_n|Y) is the number of binary combinations (binary because 2 possibilities are u_i and non-u_i) of X_1, X_2, X_3, ..., X_n in *n* positions plus the possibility of a general Y value overall in *n* + 1 possible positions, namely 2^{n+1} or 2.2^n.

But we are only interested in calculating the probability of P(X_1, X_2, X_3, ..., X_n|Y) for a given Y value (fixed Y) so that the probability of the Y value at that time is only one. This means the number of possible computations is only 2.2^{n-1}.

In general, if there are k_i the number of possible values for each X_i and k_{max} is the maximum number of k_i, then the maximum number of computations of $k_{max}.k_{max}^{n-1}$.

Next, suppose a reasonable assumption is made that each variable is independent of each other. This independent assumption causes P (X_1, X_2, X_3, ..., X_n|Y) = P(X_1, X_2, X_3, ..., X_n) that is that X_1, X_2, X_3, ..., X_n does not depend Y, and also between X_1, X_2, X_3, ..., X_n itself is independent. It means:

P (X_1, X_2, X_3, ..., X_n|Y) = P(X_1, X_2, X_3, ..., X_n) based on the principle of independent, x independent y if only if P(x|y) = P(x). But P(X_1, X_2, X_3, ..., X_n) = P(X_1, X_2, X_3, ..., X_{n-1}|X*n*). P(X*n*) based on the definition of conditional probability. But because each variable is independent, it is returned

$P(X_1,X_2,X_3,...,X_{n-1}|X_n).P(X_n) = P(X_1,X_2,X_3,...,X_{n-1}).P(X_n) = P(X_1,X_2,X_3,...,X_{n-1}|X_{n-1}).P(X_{n-1}).P(X_n)$

and so on so that it is obtained $P(X_1,X_2,X_3,...,X_n|Y) = P(X_1).P(X_2).P(X_3)...P(X_{n-1}).P(X_n)$.

From the derivation above, it is found that if independent assumptions are applied, the calculation of probability P(X_1, X_2, X_3, ..., X_n|Y) becomes multiplication of $P(X_1)P(X_2)P(X_3)...P(X_{n-1}).P(X_n)$.

Suppose that for each variable there are only 2 possible values namely u_i and non-u_i, then the probable number for each count of P (X_i) is 2 possibilities namely P(Xi = u_i) and P (Xi = non-u_i). Because there are *n* possible multiplications, namely $P(X_1)P(X_2)P(X_3)...P(X_{n-1}).P(X_n)$, the sum of all possible computations is 2*n*. In general, if there are k_i possible values for each X_i and k_{max} is the largest, the maximum number of all possible computations is $2k_{max}$.

From the calculation above, the probability equation (2) can be expressed in the form of an assumption of naive Bayes as follows:

$$\text{argmax } P(Y|X_1,X_2,X_3,...,X_n) = \text{argmax} P(X_1)P(X_2)P(X_3)...P(X_{n-1})P(X_n).P(Y)/P(X_1,X_2,X_3,...,X_n) \quad (3)$$

Equation (3) is easier to calculate because it only has 2n the number of computational probabilities

compared to equation (2) with the computational amount of 2.2^{n-1} possible computing.

But we can make a new naive assumption that even though X_1, $X2$, X_3,..., X_n are independent of each other, but X_1, $X2$, X_3,..., X_n is not independent of Y. This means $P(X_1,X_2,X_3,...,X_n|Y) \neq P(X_1,X_2,X_3,...,X_n)$. But this assumption is written as follows $P(X_1,X_2,X_3,...,X_n|Y) = P(X_1|Y)P(X_2|Y)P(X_3|Y) \ldots P(X_{n-1}|Y) P(X_n|Y)$. To understand how these assumptions are derived are as follows:

From the independent assumption that X is independent of Z if only if P (X|Z) = P(X). This means P(X, Z) = P(X|Z).P(Z) = P(Z|X).P(X) = P(X).P(Z) of the definition of probability condition. So that the independence assumption can be expanded to:
X and Z are mutually independent if only if P(X, Z)=P(X).P(Z) (4)

But if we bring (4) into the set viewpoint, then (4) can be written as P (X, Z) = P(X∩Z).

But P (X∩Z) = P (X∩Z∩U), U is a universal set where P(U) = 1.

However $P(X\cap Z\cap U) = P(X,Z,U) = P(X,Z|U).P(U)$ based on the definition of conditional probability.

Thus becoming:
$P(X,Z)=P(X\cap Z)=P(X\cap Z\cap U)=P(X,Z,U)= P(X,Z|U).P(U)=P(X,Z|U)$ (5)

Then $P(X,Z)=P(X,Z,U)=P(X\cap Z\cap U)= (X\cap U\cap Z\cap U) =P((X,U), (Z,U))$

because X, Z are independent then $X\cap U$ independent $Z\cap U$ then (X,U) independent (Z,U).
then $P((X,U), (Z,U))=P(X,U).P(Z,U)$ based on independent assumptions.

But

$P(X,U)=P(X|U).P(U)=P(X|U)$ and
$P(Z,U)=P(Z|U).P(U)=P(Z|U)$ (6)

Based on the conditional probability definition and P (U) = 1. From (5) and (6) obtained:

$P(X,Z|U)=P(X|U).P(Z|U)$ (7)

We can expand Equation (7) for each Y⊆U. Based on the view of the set, each set of X_i that is in Y can see relatively that Y is the universal set. In general, every X_i slice in Y can see that Y is the universal set, or that each X_i∩Y set can see Y relatively as its universal set. So that in relative terms, we can specify P(Y) = 1 (for all Y values) for all relative probabilities of X_i towards Y. Therefore, equation (7) can be rewritten in the perspective of relative probability as:

$P(X_1,X_2|Y)=P(X_1|Y).P(X_2|Y)$ (8)

Next if X_1, X2, X3,..., Xn mutually independent but not Y, then using induction, in general. (7) can be expanded to:

$P(X_1,X_2,X_3,...,X_n|Y)=P(X_1|Y).P(X_2|Y).P(X_3|Y)....P(X_n|Y)$ (10)

that is
$P(X_1,X_2,X_3,...,X_n|Y)= \prod P(X_i|Y)$, i=1,2,3,..., n (11)

Furthermore, the Bayes classifier equation using the assumption of naive bayes in equation (3) can be written as:

$argmax\ P(Y|X_1,X_2,X_3,...,X_n) = argmax\ P(Y).\prod P(X_i|Y) /P(X_1,X_2,X_3,...,X_n)$ (12)

With the same count as in the previous discussion, the number of computational possibilities for (12) is $2n$ or $2k_{max}$.

1.2. Tree Augmented Naive Bayes (TAN) Philosophy

The Naive Bayes model discussed in the previous chapter, encodes incorrect independence assumptions that, given the class label, the attributes are independent of each other. But in the real world, the attributes of any system are mostly correlated and the case as in Naive Bayes rarely happens. In spite of such incorrect independent assumptions, the Naive Bayes model seems to perform fairly well. So, if the model also takes into account the correlations between the attributes, then the classification accuracy can be improved (Padmanaban, 2014).

Compared to Naive Bayes, TAN has a more complicated graph structure. If in a Naive Bayes graph each attributes $X_1,X_2,X_3,...,X_n$ assumed to be independent of each other, there is no mutual influence, as in Figure 1, then in the TAN model, each attributes may be assumed to be affected by one or more other attributes as can be seen in Figure 2.

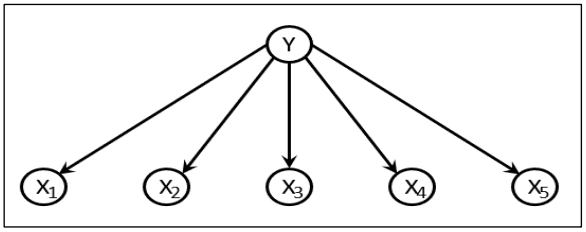

Fig. 1: An example of NB

TAN is an extended tree-like naive Bayes (Jiang *et al.,* 2005) in which the class node directly points to all attribute nodes and an attribute node can have only one parent from another attribute node (in addition to the class node). Figure 2 shows an example

of TAN. In TAN, each node has at most two parents (one is the class node). TAN outperforms naive Bayes in terms of accuracy (Jiang *et al.,* 2005) and still maintains a considerably simple structure.

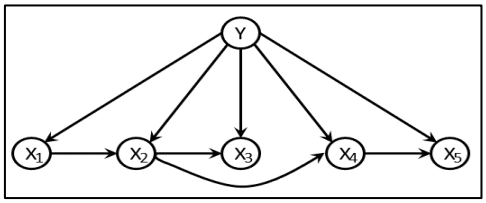

Fig. 2: An example of TAN

2. DATA SET CONSTRUCTION

Following is the construction of the dataset that you want to use to apply the Bayes classifier using the Naive Bayes calculation. The data collected is data obtained from the results of lectures in the even semester of 2017-2018 with a total of 99 records from 4 study groups.

Details of metadata from student datasets which are omitted in part in table 1 are as follows:

Table 2:. Student dataset metadata

Id	Column	Type	description
1.	NIM	Text	Student ID Number
2.	quis1	Numerical	Quiz score
3.	quis2	Numerical	Quiz score
4.	quis3	Numerical	Quiz score
5.	quis4	Numerical	Quiz score
6.	quis5	Numerical	Quiz score
7.	quis6	Numerical	Quiz score
8.	quis7	Numerical	Quiz score
9.	quis8	Numerical	Quiz score
10.	jumlah_quis	Numerical	Number of quiz followed
11.	total_quis	Numerical	The total number of quizzes for all classes
12.	Absen	Numerical	Attendance percentage
13.	UTS	Numerical	midterm exam scores
14.	UAS	Numerical	final exams scores

The following are some of the dataset pieces that you want to use in modeling Naive Bayes. In this data model, UAS is defined as a class variable, namely the Y variable in the model (12). The model (12) is used to predict the UAS value of students based on activeness values in the class (quiz values).

Next, we can write the naive bayes model as follows:

$$argmax\text{P}(UAS|quis_1,quis_2,quis_3,...,quis_8,Jumlah\_quis,Total\_quis, Absen,UTS)$$
$$=argmax\text{P}(quis_1|UAS).\text{P}(quis_2|UAS).\text{P}(quis_3|UAS)...,\text{P}(quis_8|UAS).\text{P}(Jumlah\_quis|UAS).\text{P}(Total\_quis|UAS).$$
$$\text{P}(Absen|UAS).\text{P}(UTS|UAS).\text{P}(UAS)/\text{P}(quis_1,quis_2,quis_3,...,quis_8,Jumlah\_quis,Total\_quis,Absen,UTS) \quad (13)$$

This model can be calculated on a spreadsheet sheet such as an excel sheet, but it requires a lot of accuracies to calculate all probability combinations. Therefore, in this study, the literature used in the R language has a Naive Bayes algorithm to automatically calculate the model equation (13).

3. NAIVE-BAYES's MODEL

Making a naive bayes model using the R language which is run using jupyter notebook. The library used is bnlearn. Making the model is according to the following steps:

a. Install the bnlearn library package from the cran site

In [3]: install.packages("bnlearn", repos="http://cran.us.r-project.org")

b. Loading bnlearn library into workspace

In [4]: library("bnlearn")

c. Import the provided dataset

In [5]: dataframe_nb=read.csv("Dataset nilai TBO.csv")

In [6]: dataframe_nb

A data.frame: 98 × 16

NIM	NAMA	quis1	quis2	quis3	quis4	quis5	quis6	quis7	quis8	jumlah_quis	total_quis	Absen	UTS	UAS
<int>	<chr>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
16532695	FREDY SETYO BUDI UTOMO	100	0	0	0	0	0	0	0	10	80	30	90	100
16532566	AHMAD ZAKWAN	100	80	100	90	0	0	0	0	40	80	90	70	90
16532565	FERESZA ADITIA CAHYONO	100	100	95	0	0	0	0	0	30	80	90	90	80
16532564	REYHAN PUTRA PRIMANDA	100	100	85	100	0	0	0	0	40	80	90	90	100
16532563	MUHAMMAD GILANG PRASETYO BUDI	80	100	95	0	0	0	0	0	30	80	60	80	100
16532562	LISTYA RIZKI ANDRIANI	100	100	100	0	0	0	0	0	30	80	90	90	100
16532561	DWI KRISTIANA	100	95	75	100	80	100	0	0	60	80	90	90	100
16532560	HANDIKA KUMALA SENA	100	100	90	0	0	0	0	0	30	80	60	70	100

Fig. 3: An screenshoot example of dataset

d. Clean the dataset by eliminating the "NIM" column and "NAMA" column so that a pure dataset is obtained with values that can be calculated to form a Naive Bayes model

In [7]:

```
dataframe_nb2<-
dataframe_nb[,c("quis1","quis2","quis3","quis4","q
uis5","quis6",
"quis7","quis8","jumlah_quis","total_quis","Absen
","UTS","UAS")]
```

e. Convert each value to factor data type

In [9]:

```
quis1=factor(dataframe_nb2[,"quis1"])
quis2=factor(dataframe_nb2[,"quis2"])
quis3=factor(dataframe_nb2[,"quis3"])
quis4=factor(dataframe_nb2[,"quis4"])
quis5=factor(dataframe_nb2[,"quis5"])
quis6=factor(dataframe_nb2[,"quis6"])
quis7=factor(dataframe_nb2[,"quis7"])
quis8=factor(dataframe_nb2[,"quis8"])
jumlah_quis=factor(dataframe_nb2[,"jumlah_quis"
])
total_quis=factor(dataframe_nb2[,"total_quis"])
Absen=factor(dataframe_nb2[,"Absen"])
UTS=factor(dataframe_nb2[,"UTS"])
UAS=factor(dataframe_nb2[,"UAS"])
```

f. Then arrange them all into a new dataset in the data frame format in the R language

In [10]:

```
dataframe_nb3<-
data.frame(quis1,quis2,quis3,quis4,quis5,quis6,quis7,qu
is8, jumlah_quis,tot al_quis,Absen,UTS,UAS)
```

In [11]: dataframe_nb3

```
dataframe_nb3
```

A data.frame: 98 × 13

quis1	quis2	quis3	quis4	quis5	quis6	quis7	quis8	jumlah_quis	total_quis	Absen	UTS	UAS
<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>	<fct>
100	0	0	0	0	0	0	0	10	80	30	90	100
100	80	100	90	0	0	0	0	40	80	90	70	90
100	100	95	0	0	0	0	0	30	80	90	90	80
100	100	85	100	0	0	0	0	40	80	90	90	100
80	100	95	0	0	0	0	0	30	80	60	80	100
100	100	100	0	0	0	0	0	30	80	90	90	100
100	95	75	100	80	100	0	0	60	80	90	90	100
100	100	90	0	0	0	0	0	30	80	60	70	100

Fig. 4: An screenshoot example of dataset after conversion to data frame format

g. Naive Bayes model construction
In [12]: bn = naive.bayes(dataframe_nb3, "UAS")
bn is a naive Bayes model that has been created based on previously imported datasets.

h. Show the naive Bayes graph of the created model
In [15]: plot(bn)

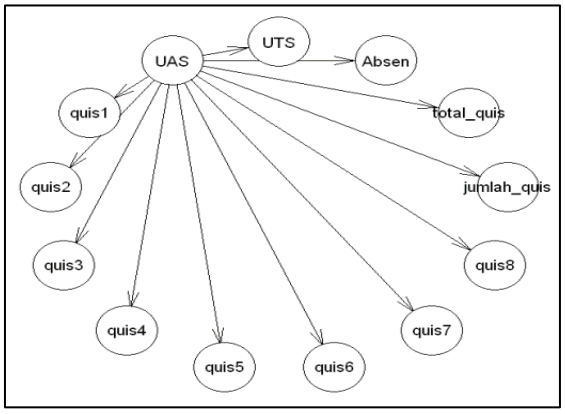

Fig. 5: A Naive Bayes graph for model

4. TAN's MODEL

The making of the TAN model as a comparison for the Naive Bayes model is as follows:

a. Create a model using the tree.bayes function in the bnlearn library
In [17]: tan = tree.bayes(dataframe_nb3, "UAS")

b. Show results
In [18]: Tan

```
tan

  Bayesian network Classifier

  model:
   [UAS][quis1|UAS][Absen|UAS:quis1][jumlah_quis|UAS:Absen]
   [total_quis|UAS:Absen][quis2|UAS:jumlah_quis][quis3|UAS:jumlah_quis]
   [quis4|UAS:jumlah_quis][quis5|UAS:jumlah_quis][quis6|UAS:jumlah_quis]
   [UTS|UAS:jumlah_quis][quis7|UAS:quis6][quis8|UAS:quis6]
  nodes:                                13
  arcs:                                 23
    undirected arcs:                    0
    directed arcs:                      23
  average markov blanket size:          3.54
  average neighbourhood size:           3.54
  average branching factor:             1.77

  learning algorithm:                   TAN Bayes Classifier
  mutual information estimator:         Maximum Likelihood (disc.)
  training node:                        UAS
  tests used in the learning procedure: 66
```

Fig. 6: A TAN model description

c. Show the graph of the TAN model
In [19]: Plot(Tan)

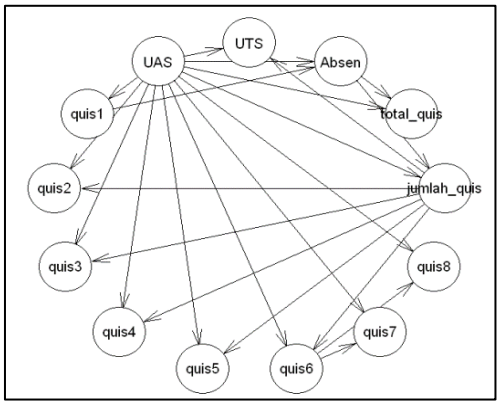

Fig. 7: A TAN model graph

5. COMPARISON USING CROSS VALIDATION METHODS

Comparison of the two models generally uses cross validation methods, namely the default k-fold method (10- fold cross validation), hold-out cross validation, 5-fold cross validation. Perform a k-fold or hold-out cross-validation for a learning algorithm or a fixed network structure (Scutari & Ness, 2018). The comparison is as follows:

a. Default k-fold (k = 10)

Naive Bayes Model Profile

In [23]: bn.cv(dataframe_nb3, bn, loss = "pred", loss.args = list(target = "UAS")

```
In [23]:
bn.cv(dataframe_nb3, bn, loss = "pred", loss.args = list(target = "UAS"))

  k-fold cross-validation for Bayesian networks

  target network structure:
   [Naive Bayes Classifier]
  number of folds:                    10
  loss function:                      Classification Error
  training node:                      UAS
  expected loss:                      0.4897959
```

Fig. 8: A screenshoot for Naive Bayes profil by k-fold default

TAN Model Profile

In [22]: bn.cv(dataframe_nb3, tan, loss = "pred", loss.args = list(target = "UAS"))

```
In [22]:
bn.cv(dataframe_nb3, tan, loss = "pred", loss.args = list(target = "UAS"))

  k-fold cross-validation for Bayesian networks

  target network structure:
   [UAS][quis1|UAS][Absen|UAS:quis1][jumlah_quis|UAS:Absen]
   [total_quis|UAS:Absen][quis2|UAS:jumlah_quis][quis3|UAS:jumlah_quis]
   [quis4|UAS:jumlah_quis][quis5|UAS:jumlah_quis][quis6|UAS:jumlah_quis]
   [UTS|UAS:jumlah_quis][quis7|UAS:quis6][quis8|UAS:quis6]
  number of folds:                    10
  loss function:                      Classification Error
  training node:                      UAS
  expected loss:                      0.4897959
```

Fig. 9: A screenshoot for TAN profil by k-fold default

b. Hold-out cross validation

Naive Bayes Model Profile

In [25]: bn.cv(dataframe_nb3, bn, loss = "pred", loss.args = list(target = "UAS"), method="hold-o ut")

```
In [25]:
bn.cv(dataframe_nb3, bn, loss = "pred", loss.args = list(target = "UAS"), method="hold-o
ut")

  hold-out cross-validation for Bayesian networks

  target network structure:
   [Naive Bayes Classifier]
  number of splits:                   10
  size of the test subset:            10
  loss function:                      Classification Error
  training node:                      UAS
  expected loss:                      0.48
```

Fig. 10: A screenshoot for Naive Bayes profil by Hold-out cross validation

TAN Model Profile

In [26]: bn.cv(dataframe_nb3, tan, loss = "pred", loss.args = list(target = "UAS"), method="holdout")

```
In [26]:
bn.cv(dataframe_nb3, tan, loss = "pred", loss.args = list(target = "UAS"), method="hold-
out")

  hold-out cross-validation for Bayesian networks

  target network structure:
   [UAS][quis1|UAS][Absen|UAS:quis1][jumlah_quis|UAS:Absen]
   [total_quis|UAS:Absen][quis2|UAS:jumlah_quis][quis3|UAS:jumlah_quis]
   [quis4|UAS:jumlah_quis][quis5|UAS:jumlah_quis][quis6|UAS:jumlah_quis]
   [UTS|UAS:jumlah_quis][quis7|UAS:quis6][quis8|UAS:quis6]
  number of splits:                  10
  size of the test subset:           10
  loss function:                     Classification Error
  training node:                     UAS
  expected loss:                     0.46
```

Fig. 11: A screenshoot for TAN profil by Hold-out cross validation

c. 5-fold cross validation

Naive Bayes Model Profile

In [29]: bn.cv(dataframe_nb3, tan, loss = "pred", loss.args = list(target = "UAS"), method="k-fol d", k=5)

```
In [29]:
bn.cv(dataframe_nb3, tan, loss = "pred", loss.args = list(target = "UAS"), method="k-fol
d", k=5)

  k-fold cross-validation for Bayesian networks

  target network structure:
   [UAS][quis1|UAS][Absen|UAS:quis1][jumlah_quis|UAS:Absen]
   [total_quis|UAS:Absen][quis2|UAS:jumlah_quis][quis3|UAS:jumlah_quis]
   [quis4|UAS:jumlah_quis][quis5|UAS:jumlah_quis][quis6|UAS:jumlah_quis]
   [UTS|UAS:jumlah_quis][quis7|UAS:quis6][quis8|UAS:quis6]
  number of folds:                   5
  loss function:                     Classification Error
  training node:                     UAS
  expected loss:                     0.4897959
```

Fig. 12: A screenshoot for Naive Bayes profil by 5-fold cross validation

TAN Model Profile

In [30]: bn.cv(dataframe_nb3, bn, loss = "pred", loss.args = list(target = "UAS"), method="k-fold ", k=5)

```
In [30]:
bn.cv(dataframe_nb3, bn, loss = "pred", loss.args = list(target = "UAS"), method="k-fold
", k=5)

  k-fold cross-validation for Bayesian networks

  target network structure:
   [Naive Bayes Classifier]
  number of folds:                   5
  loss function:                     Classification Error
  training node:                     UAS
  expected loss:                     0.4897959
```

Fig. 13: A screenshoot for TAN profil by 5-fold cross validation

6. CONCLUSIONS AND RECOMMENDATIONS

From the comparison results of the k-fold method using the bn.cv function in the bnlearn library, it is found that for Hold-out cross validation, the TAN method is better than the Naive Bayes method, especially for the case study dataset given. This is indicated by the value of expected loss (minimization of error) which is smaller in the TAN method, which is 0.46, but slightly larger in the Naive Bayes method, which is 0.48. Only a difference of 0.02. As for the 5-fold cross validation and 10-fold cross validation (k-fold default), both of them reached the same expected loss value, which is 0.4897959. namely that both are equally good.

This gives the conclusion that for data cases such as case studies from datasets, with relatively small data sizes, the TAN and Naive Bayes methods are not significantly different, although TAN looks a little better.

REFERENCES

- Das, S., & Kolya, A. K. (2017). Sense GST: Text mining & sentiment analysis of GST tweets by

Naive Bayes algorithm. *2017 Third International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, 239–244. https://doi.org/10.1109/ICRCICN.2017.8234513

- Hsu, S.-C., Chen, I.-C., & Huang, C.-L. (2016). Image classification using pairwise local observations based Naive Bayes classifier. *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA ASC 2015*, *December*, 444–452. https://doi.org/10.1109/APSIPA.2015.7415311
- Jiang, L., Zhang, H., Cai, Z., & Su, J. (2005). Learning tree augmented naive Bayes for ranking. *Lecture Notes in Computer Science*, *3453*, 688–698. https://doi.org/10.1007/11408079_63
- Marsono, M. N., El-Kharashi, M. W., & Gebali, F. (2006). Binary LNS-based na?ve Bayes hardware classifier for spam control. *Proceedings - IEEE International Symposium on Circuits and Systems*, 3674–3677.
- Nirmala, K., Venkateswaran, N., & C, V. K. (2017). *HoG Based Naive Bayes Classifier for Glaucoma Detection*. 2331–2336.
- Padmanaban, H. (2014). *Comparative Analysis of Naive Bayes and Tree Augmented Naive Bayes Models Presented to The Faculty of the Department of Computer Science San José State University In Partial Fulfillment of the Requirements of the Degree Master of Science by Harini Padmana*. San Jose State University.
- Scutari, M., & Ness, R. (2018). *Package ‘ bnlearn .’* Repository CRAN.