What explains the difference between naive Bayesian classifiers and tree-augmented Bayesian network classifiers.

Author

Thomas Wojcik
5915864

Master Thesis
Computing Science

Supervisor
Dr. Silja Renooij
Dr. Thijs van Ommen

Department of Information and Computing Sciences
Utrecht University

The Netherlands

April 2023

Utrecht University

Contents

Abstract

Naïve Bayesian Networks (NB) have been proven to be decently accurate classifiers, even in cases where their independency assumption does not hold. An approach to relax the independency assumption is to search through the possible single dependencies that can be added to the network, creating a so called Tree Augmented Bayesian Network (TAN), with the intention to improve the performance of the network. However, these TAN classifiers often perform about as good as a NB classifier, while increasing computational cost. In this research we will show what causes the difference in performances between these classifiers. This will be done by comparing NB and TAN classifiers learned from theoretical data-sets to test different theories. Afterwards, these theories are tested on real world data-sets, to see if these theories also hold in practice. The comparison between NB and TAN will not only be performed on their accuracy, but also on metrics pertaining to uncertainty, such as their Brier scores.

1 Introduction

Classification is the task of identifying the class label for instances, based on their features, and is one of the most important tasks in data mining. Some examples of classification problems are: *what kind of object is seen in an image?* Or *what disease does a patient suffer from?*. In these cases the color value of each pixel and symptoms of the patient are the features describing the instance. The instances are the image and the patient and their classes are the kind of object in the image and the disease the patient has.

As the examples show, classifiers can be used in a very wide field of practices, which makes learning accurate classifiers from pre-classified data a very active research topic in computer science and has been researched for decades. When deciding on what classification technique to use for any given problem, multiple aspects of the technique should be considered: how much training data does the technique need? How long does training the classification model take? How accurate will the classification model be? How fast can the model classify instances?

Depending on the task at hand, certain questions become more important than others, and having a technique outperform another based on a single of these metrics can decide if it is used or not. One simple yet effective classifier is the Naive Bayes classifier(NB). This classifier can be represented using a Bayesian Network ([20], [21]): a probabilistic graphical model with a directed graph, to establish probabilistic independencies between variables in the network, combined with probability functions for each node in the graph, see Figure 1 for an example. Each node in the network represents a variable and the associated probability function takes in the node's parent variables and outputs the probability distributions of the node's possible values. The naive part of Naive Bayes is that it is a specific type of Bayesian Network: it makes the assumption that all feature variables are independent given the class. This means that Figure 1 is not a Naive Bayesian Network, More is explained about Naive Bayes in section 2.2.

Some advantages that Naive Bayes has are that it requires a small training data-set to be accurate in its predictions ([15]), can be learned fast from any data-set, and is fast with classifying once learned ([8]). One of the big advantages in terms of learning speed compared to other classification techniques, is that there is no need to search for the model structure, as the Naive Bayesian Network always has the same structure adapted to the amount of variables, whereas other classification techniques often have to learn a structure before learning the parameters.

This has lead to making NB a popular research topic within the field of classification. Researchers have shown many useful cases where Naive Bayes performed good([12]), found explana-

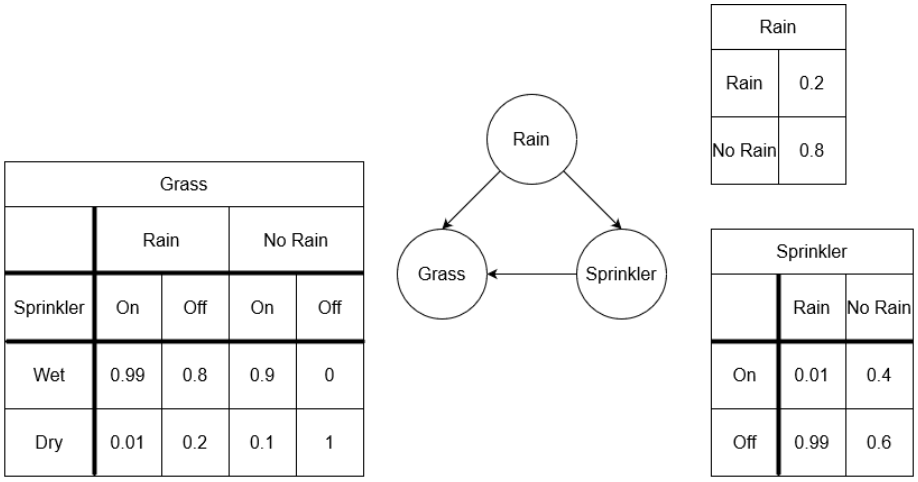


Figure 1: a representation of a Bayesian Network as a directed graph with probability tables. The edges in the graph show the dependency structure and the probability tables indicate the probabilities of each value of each node and show how these probabilities are influenced by their parent nodes.

tions for why it performed so good([15], [16], [22], [25]), and found variations of the model that improved performance even further. One of these variations of Naive Bayes is called the Tree Augmented Bayesian Network (TAN)([2], [7], [8], [13], [17]). TAN tries to improve Naive Bayes by relaxing the independency assumption, but it is not a clear winner over NB, as can we show in Table 1. More about TAN and how it compares to NB is explained in section 2.

The goal of this research is to find insight as to why the more simple technique Naive Bayes sometimes performs better than the more elaborate technique Tree Augmented Naive Bayes. This leads to the following research question:

Research question: *What explains the difference between naive Bayesian classifiers and tree-augmented Bayesian network classifiers?*

Answering this question would allows researchers or data analysts to have an understanding of which technique will perform better than the other before trying both and comparing their results. To answer this research question, we will perform experiments to answer the following sub-questions, which combined will answer our research question.

Research sub-question 1: *How do the two classifiers compare?*

We first looked for insight on this question by looking at the results of related works, which we summarized in Table 1. In this table we see that TAN performs better than NB in most cases. However, most experiments do not have a very clear winning technique, as the difference in accuracy are rather small. Further insights from previous work is given in Section 2. We will discuss what insights we have gained from each experiment on how the two classifiers compare in Section 3.1.

Research sub-question 2: *Can performance differences be explained by dependencies in the data?*

Data-set	NB	TAN
adult	82.82	83.78
anneal	94.32	97.44
anneal.ORIG	87.53	89.08
audiology	71.23	71.66
australian	86.23	84.2
autos	64.83	75.5
balance-scale	91.36	91.36
breast	96.72	90.24
breast-cancer	72.06	71.01
breast-w	97.28	97.42
car	86.15	90
chess	87.23	92.07
cleve	81.79	74.60
colic	78.81	79.63
colic.ORIG	75.26	75.8
corral	85.88	96.06
credit-a	84.78	84.78
credit-g	76.3	75.3
crx	85.54	78.23
diabetes	74.94	75.66
DNA	94.27	93.59
flare	78.13	80.01
german	74.35	73.1
glass	70.49	61.92
glass2	79.17	77.92
heart	82.74	83.33
heart-c	84.14	82.45
heart-h	84.05	84.05
heart-statlog	83.7	82.96

Data-set	NB	TAN
hep	77.38	66.36
herpatitis	87.52	87.52
hypothyroid	92.79	93.35
ionosphere	83.95	90.32
iris	89.52	86.19
kr-vs-kp	87.89	95.5
labor	93.33	93.33
letter	78.14	80.25
lymph	85.67	86.38
lymphography	79.72	85.03
mofn-3-7-10	86.43	91.11
mushroom	95.57	99.83
nursery	90.31	92.23
pima	73.08	72.94
primary-tumor	46.89	47.5
satimage	75.38	87.2
segment	90.05	94.98
shuttle-small	98.34	99.53
sick	96.74	97.61
sonar	75.75	76.07
soybean	92.96	85.94
soybean-large	91.29	92.17
splice	95.36	95.45
vehicle	61.7	69.87
vote	90.12	92.14
vowel	79.04	92.42
waveform-21	77.89	78.38
waveform-5000	79.96	82.18
zoo	96.09	94.18

Table 1: Average accuracy between NB and TAN on different data-sets ([6], [8], [11], [14]). Out of these 67 data-sets, TAN has a higher accuracy than NB on 42 data-sets, NB has a higher accuracy on 10 of the data-sets, and both models tie on 15 of the data-sets when taking into account the standard error given by the researches.

As will be explained in Section 3, these research questions will be answered by performing experiments using different data-sets. These data-sets will be generated from either self-created Bayesian networks with specific dependency properties or from real-world Bayesian networks. The experiments using self-created Bayesian networks will help give insight into how dependency properties in the data influences the performance of NB and TAN. Afterwards, these insights are tested on real-world data to see if they hold in the real-world.

2 Preliminaries

This section gives further explanation about how Bayesian networks work for readers without complete knowledge on this topic. The section will not explain all there is to know about Bayesian networks, but should allow readers with a background in computer science to understand the rest of this thesis. We will give specific detail on how Naive Bayesian networks and Tree Augmented Bayesian networks work.

2.1 Bayesian Networks

As previously mentioned and showed in Figure 1, a Bayesian network is a compact representation of a joint distribution by using a set of nodes which are connected by directed edges. Each node represents a variable and contains the probability distribution for each combination of states of the variables from which the node has an incoming edge. We call the node with an outgoing edge the *parent* of the node to which the edge points, which is then called the *child* node. In Figure 1 we show the probabilities of the *Grass* variable being either *Wet* or *Dry* depending on the state of *Rain* and *Sprinkler*.

The direction of the edges in a Bayesian network can represent a causal relationship between the nodes; however, this is not necessary. Knowing the value of a child node influences the probability distribution of a parent node, without changing how the two variables depend on each other, whereas this is not the case in a causal relationship.

The edges in a Bayesian network represent possible dependencies, which does not mean that nodes that are not connected have to be independent. Two nodes are independent in a Bayesian network when they are *d-separated*. We will not go into detail into how nodes can be d-separated from each other, but in the case of Naive Bayes all feature variables are d-separated from each other given the class variable, and in the case of Tree Augmented naive Bayesian networks the features are d-separated from each other given the class variable and its parent variable. This allows for easier computation of the probabilities in the network, which will be explained further later on, while still making the variables dependent on each other when the class is not known, which is the case in classification tasks. The edges in a Bayesian network are not allowed to form a cycle by following the direction of the edges.

A Bayesian network calculates the probabilities of all unknown variables given the known variables. The known variables are often called *evidence*. The probabilities of each variable without any evidence in the network are called the prior probabilities and the probabilities given the evidence are called the posterior probabilities. Calculating the posterior probabilities is done by using the probability distribution in each node, given its parent variables, and applying Bayes' rule for the child nodes, see Equation 1. The evidence does not have to be the known state of a variable's direct parents or children. The previously mentioned d-separation is used to know which probabilities change when there is evidence for a variable in the Bayesian network, as a variable that is d-separated from another variable has no influence on that variable.

$$P(Parent|Child) = \frac{(Child|Parent)P(Parent)}{P(Child)} \tag{1}$$

Next we will explain in more detail how Naive Bayes works and is used for classification. Afterwards we will also explain how tree augmented Bayesian networks work and what is known about its performance.

2.2 Naive Bayes

The Naive Bayes classifier ([12], [15], [16], [22], [23], [25]) is a classification model that uses a Bayesian Network to calculate the probability that an instance has for every possible class it can have, under the naive assumption that all features of the instance are independent given the class. This leads to the dependency graph structure in Figure 2 when looking at the solid edges. It is used for classification by classifying an instance as the class with the largest probability. The independence assumption on the features of each instance gives the technique its name and is its strength, because it makes the math for the conditional probabilities in the Bayesian network a lot simpler.

To calculate the probability of an instance being a certain class w_i given the combinations of values x for the feature values it would be possible to learn the conditional probability distribution of the class given the features $P(w_i|x)$, such as the table for *Grass* in Figure 1. However, the joint distribution of the class and feature variables increases exponentially in size with the amount of feature variables, which would make this calculation exponentially long. Therefore, we instead use the Bayes' rule, see equation 2, which allows us to calculate $P(w|x)$ from the likelihood of the features given the parent node $P(x|w)$, and the probabilities of the classes $P(w)$.

$$P(w_i|x) = \frac{P(x|w_i)P(w_i)}{\sum_{j=1}^{c} P(w_j)P(x|w_j)} \qquad i = 1, ..., c \tag{2}$$

$$P(x|w_i) = \prod_{j=1}^{n} P(x_j|w_i) \qquad i = 1, ..., c \tag{3}$$

This reduces the computation time to learn the probabilities in the model from data to $O(n + c \times |x|)$, where n is the number of data points in the data-set, c is the number of different classes and $|x|$ the number of feature variables. This is the case because the frequencies of each class, feature and likelihood of features can be found by going through the data-set once ($O(n)$), after which the posterior probability $P(w|x)$ can be calculated by filling in equation 2 for every class ($O(c \times |x|)$). To classify an instance, the model only has to calculate the probability of the class given the data, which can be done in $O(c \times |x|)$ as previously mentioned.

Another advantage of dividing the large joint distribution is that the marginal probability distributions of each variable can be more accurately estimated from smaller amounts of data ([15]). Because of this, Naive Bayes has proven itself to be a competitive classifier in terms of accuracy with other state-of-the-art classifier techniques ([12], [16], [22]), even though the independency assumption on which it is based often does not hold on real data-sets.

To explain what causes the NB classifier to perform good, despite its strong independence assumption, researchers looked into experimental and theoretical approaches. It was found through experimentation that NB performs good in both of the cases where the independency assumption holds and where the features are fully dependent on each other ([22]), which means that when one feature is assigned a value, all other features are determined in a deterministic way.

Further research showed that NB also performs good with inaccurate prior probability estimates ([23]). This research also mentions that entropy and mutual information of the features are often used for estimating how well NB will perform and shows that entropy is the better predictor for the probability of a wrong classification.

The understanding that Naive Bayes performs worse when the feature variables have strong dependencies has lead to Kuncheva et al ([15]) and de Wachter ([25]) to do research on the bounds

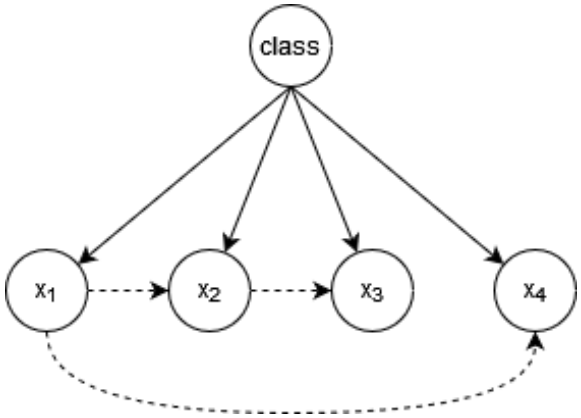


Figure 2: a representation of a Bayesian Network as a directed graph. The solid lines form a Naive Bayesian Network where all feature variables are not directly connected and independent given the class, whereas by including the dotted lines a Tree Augmented Bayesian Network is created with single dependencies among the feature variables.

on the error of the feature variable probabilities for Naive Bayesian networks with 2 and 3 feature variables. Both show that their found bounds are largest when the features are fully dependent. Other research that came from this understanding are researches into variants of NB that can handle dependent features better than NB can. One such variant is called Tree Augmented Naive Bayes.

2.3 Tree Augmented Naive Bayes

The idea for Tree Augmented Naive Bayes (TAN) is to use Naive Bayes, but allow some dependencies among feature variables in the network, such that the posterior class probabilities are more accurate when the features have dependencies on each other ([2], [7], [8], [13], [17]). The *tree* in Tree Augmented Bayesian Network is the tree-like structure of dependencies that is added to the feature variables. The tree structure of the original TAN classifier by Friedman et al ([11]) allowed for only a single parent for each feature variable in the tree structure. Some later variations for TAN, such as ETAN by Long et al [17]), relaxed this restriction by allowing any predefined amount of parents for each feature variable in the tree structure, as long as the structure still satisfies the Bayesian network restrictions.

This tree is found through a short structure searching algorithm ([11], [13]), see Algorithm 1. In this algorithm the class conditional mutual information between two features is calculated using Equation 4. Mutual information is an indication of how much the uncertainty of a feature is reduced by knowing the state of another feature ([9]), see Equation 5. The class conditional mutual information is an indication of how much the uncertainty of a feature is reduced by knowing the state of another feature and the value of the class variable. Friedman et al ([11]) prove in their research that the tree structure that maximizes the log likelihood of the feature variables is found by using the class conditional mutual information to determine the tree structure. An example of such a TAN model can be seen in Figure 2 by looking at both the solid and dotted lines.

Algorithm 1 TAN

input: $TrainingSet$
$CondMutInfo \leftarrow Dictionary$
for Feature X in $TrainingSet$ **do**
  for Feature Y in $TrainingSet$ and $Y! = X$ **do**
    $CondMutInfo[(X, Y)] \leftarrow CMI(X, Y|W)$
  end for
end for
$Graph \leftarrow FullConnectedGraph$
$Graph.Weights \leftarrow CondMutInfo$
$Tree \leftarrow MaxWeightSpanningTree(Graph)$
$DirectedTree \leftarrow OutwardDirectionsFrom(TrainingSet.Features[0], Tree)$
$TAN \leftarrow NaiveBayesStructure$
for Edge E in DirectedTree **do**
  $TAN.Edges.Append(E)$
end for
Return: TAN

$$CMI(X, Y|W) = \sum_{w} P(w) \sum_{y} \sum_{x} P(x, y|w) \log_2(\frac{P(x, y|w)}{P(x|w)P(y|w)}) \tag{4}$$

$$MI(X, Y) = \sum_{y} \sum_{x} P(x, y) \log_2(\frac{P(x, y)}{P(x)P(y)}) \tag{5}$$

Besides adding computing time for finding the additional tree structure, TAN also increases computational workload in equations 2 and 3 ([17]). Because of the additional parent node that all of the feature nodes have, besides the root of the tree, the probability functions get another dimension, which increases the computation time of classification by a factor of the maximum amount of possible values of a feature and increases the computational space of the model in memory by an equal amount.

Although this additional setup time and the increased calculations when classifying are not trivial, this relaxation is still computationally cheap compared to other classification models ([2], [8], [11]), allowing TAN to be a competitive model to state-of-the-art classification techniques.

Not much is known on the exact behavior of TAN, such as how good or bad it can perform in extreme cases, or what causes it to perform good or bad. Instead many of the research focus on new variants of TAN, in order to improve the model's performance even further ([2], [8], [13], [17]). Most of these variants of TAN have the same idea supporting their success theoretically: extending the model to catch more dependencies in the data, which leads to the model's performance improving.

One such variant is called Average TAN (ATAN)([13]). It expands the structure searching step by calculating a maximum weighted spanning tree for each feature variable, using that feature variable as the root node. This results in multiple TAN models. When predicting, the average probability of the class variable of all TAN models is used. Because the tree structure differs between each model, the models simulate different dependencies, allowing the model to classify instances as accurate or better than TAN.

Another variant is named Extended TAN (ETAN) ([17]). This variant expands the structure searching step by building the tree structure in iterations, where each iteration adds a single arc to the network, but instead of adding only one arc per node, two arcs are added per node. This results in one model with many arcs between the feature variables. These added arcs also capture different dependencies and result in a significant better performance than TAN in most cases.

2.4 Comparing NB and TAN

Since the inception of TAN it has been compared to NB, as it is an extension of the technique. Most often this is done on classification accuracy, but sometimes other metrics are used, such as zero-one loss or log-likelihood. Unfortunately, none of the researches on TAN dive as deep into what causes TAN to perform good and what causes it to perform worse, as is done for NB.

Although some researches only denote the win/tie/loss ratios between NB, TAN and their classification technique, many of them do provide their achieved accuracy on different data-sets. We have made table 1 to show the average accuracy of different works for NB and TAN. In it we can see that on average TAN has a higher accuracy on 36 out of 56 data-sets and NB has the higher accuracy on 17 data-sets and their accuracy are equal on the 3 other data-sets. This shows that although TAN performs better than NB on most of these data-sets, it is not a clear victory.

As mentioned in section 2.2, there exist proofs for the bounds on the error on NB, which are largest for fully dependent variables. However, no such proofs exists as of now for TAN, as it is mathematically harder to write a proof for all scenario's, as the scenario's scale exponentially with the amount of dependencies within the network. This amount increases at an exponential rate for TAN, due to the additional dependency between the variables.

3 Experiments

First I will give a general overview of how I worked to answer my research questions, before I go into detail of every experiment, including their implementation, in the next chapter.

To answer sub-questions 1 and 2, experiments on generated data-sets are performed. The data-sets are each time created from a Bayesian network (the truth). This way the predicted probabilities of NB and TAN can be compared with the use of the Brier score.

The first of these experiments uses fully independent variables, see Section 3.4 for the detailed implementation. This experiment can be referenced as a base case for results on completely random data. My expectation was for NB and TAN to perform about equally bad, as both assume there is a structure to the data, which is not present. I expected NB to perform slightly better than TAN, because it has no dependencies between the feature variables, which makes it closer to the non-existent structure of the data.

To also have a base case for when both NB and TAN perfectly represent the structure in the data, experiments with data generated from NB and TAN networks were performed. See Section 3.3 for the details of this experiment. The expectation was for both models to perform very good on these experiments, because their network structures either are exactly as the structure of the data, or are very close to the structure of the data, because NB and TAN have very similar structures. I expect that NB will perform slightly better on the data generated by NB networks and TAN to perform slightly better on the data generated by TAN networks.

To answer sub-question 1, experiments with different structures needed to be performed in order to find a pattern in the structures that cause either NB or TAN to outperform the other. To gain

insight into this, experiments with random Bayesian network structures were performed, see Section 3.4. The experiments also give insight in how NB and TAN perform on data generated from sparse and dense networks. A pattern was found in the Bayesian networks that generated data-sets on which TAN would outperform NB. To confirm this belief, three follow up experiments are used.

First, an extreme example of the pattern was tested using Bayesian networks with only 3 variables and two dependencies, see Section 3.5.1. Second, randomly generated Bayesian networks with the found pattern are used to further test if the pattern is the cause for TAN to outperform NB.

Finally, to answer sub-question 2 and test the found pattern, an experiment was performed on data-sets generated from real world Bayesian networks that have the desired property. Although the previously mentioned experiments help understand what causes TAN to perform better or worse, this experiment gives some insight in how realistic these theoretical experiments are.

3.1 Methodology

As mentioned before, I will answer the research questions through insights gained from experiments. Each experiment will be comparing the performance of NB and TAN, trained on the same data-set used for that experiment, with each other on accuracy, Brier score (Equation 6) ([5]), root mean square error (RMSE, Equation 7) and average error of the posterior class probability of the data-generating Bayesian network (Bayes error, Equation 8). Each experiment is repeated 100 times, resulting in 100 values of these four performance metrics.

Each experiment will have different structure of the data-generating Bayesian network. The details of the structure used will be described in the chapters of the experiments themselves, accompanied by an image showcasing the structure. Some experiments will use some randomness in the structure of the data-generating network. In such cases the structure of the data-generating network will be randomly generated each time the experiment is repeated.

The data-set used in each experiment consists of 2000 sampled data-points from the data-generating Bayesian network. The data-set will be split 90%-10% into training- and test-set. The training-set will be used to find the tree structure of the TAN model and estimating the likelihood of each individual variable in both the NB and TAN models. The test-set will be used for prediction and the performance of this prediction is measured and shown as the result of the experiment.

The data-generating Bayesian networks will have 10 feature variables and 1 class variable and all variables will be binary. The *High Mutual Information* experiments in Section 3.5 are an exception to this. How many feature variables are used and how many different values the class and feature variables have are discussed in the details of these *High Mutual Information* experiments.

The structure for these Bayesian networks will differ between each experiment. All probabilities in the data-generating Bayesian networks will be determined randomly using a uniform distribution. Each experiment is repeated for 100 different data-sets, each generated from a different Bayesian network with different probability distributions. NB and TAN are both trained and tested on the same 100 data-sets, resulting in 100 data-points of each performance metric.

The experiments are coded in *Python* ([10]). The *pgmpy* ([1]) library is used for all probability calculations using its implementation of Bayesian networks. The *fit* function is used to train the Bayesian networks and the *simulate* function for generating data. The *NumPy* ([19]) library is used for all complex mathematical functions, such as the square root and logarithm. It is also used for the global random number generator, which uses the same seed for each experiment, but a different state each time an experiment is repeated. The *pandas* ([18]) library is used for the data structures used for the data-sets with all of its variables.

The Brier score ([5]) is calculated using equation 6, where $P_{Pred}(W = 1|x_i)$ is the probability of the class being 1 given a combination of feature values x, c is the true value of the class in the data-set and $n = 200$, as it is averaged over each prediction on the test-set. This results in the Brier score being equal to the mean squared error of the prediction. A larger Brier score is worse than a low Brier score, with a score of 1 being the worst case, which will happen if the model predicts a probability of 1 for the incorrect class in the data. In contrast to accuracy, the Brier score can be used to ascertain how uncertain a model is in its prediction, as it compares how close its prediction is to the real occurrence. For example, a model that correctly classifies a full data-set with 100% accuracy, but predicts every time a 51% chance for the correct class, will have a $(0.51-1)^2 = 0.2401$ Brier score.

$$AverageBrierScore = \frac{1}{n}\sum_{i=1}^{n}(P_{Pred}(W = 1|x_i) - c)^2 \tag{6}$$

Because each data set will be generated by a Bayesian network, we will have access to the "real" probability of the class variable from the data-generating network given the evidence. To see how well NB and TAN will be at predicting this probability, the Root Mean Square Error (RMSE) of the class probability will be used for comparison, see equation 7. Where $P_{Pred}(W = 1|x_i)$ is the predicted probability of the binary class being 1, $P_{Gen}(W = 1|x_i)$ is the probability of the data-generating model of the binary class being 1 and $n = 200$.

$$RMSE = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(P_{Pred}(W = 1|x_i) - P_{Gen}(W = 1|x_i))^2} \tag{7}$$

Just as with the Brier score, a lower RMSE is better than a higher RMSE, with 1 being the worst case scenario and 0 being the best case scenario. An example of when the RMSE might give us insight that accuracy and the Brier score cannot give is when both the data-generating and the predicting network have a $P_{Pred}(W = 1|x_i)$ of 50%, but in prediction each time the wrong class is allocated. In this case the accuracy of the model would be 0%, its Brier score would be $(1 - 0.5)^2 = 0.25$, but its RMSE would be 0, showing that the model has accurately estimated the class probability.

To give insight in how uncertain the data-generating model is for each data point, the average Bayes Error is measured, see equation 8. Here c_i is the value of class which is 0 or 1, $P_{Gen}(W = 1|x_i)$ is the probability of the class being 1, $n = 200$, and $ABS()$ is a function which returns the positive number of the value it is called on.

$$AverageBayesError = \frac{1}{n}\sum_{i=1}^{n} ABS(c - P_{Gen}(W = 1|x_i)) \tag{8}$$

This makes the Bayes Error the average probability of the data-generating model for the incorrect class of the data-points in the data-set. An interpretation of the Bayes Error of a model would be *how well does the generative class probability reflect the class values in the data-set.* A high Bayes Error shows that the data-generating Bayesian network has a high error in the probability of the class. This is an indication that a Bayesian network with a good estimation of the "true" class probability, indicated by a low RMSE, will likely have a low accuracy and a high Brier score, because the true class probability was unlikely to generate the class values in the data-set.

If the class probability is well represented in the data-set, then the largest the Bayes Error can be is 0.5, which is the case if the class probability is a 50/50 split. If the class probability is

any other distribution, then the Bayes Error should be lower. The Bayes Error being larger than 0.5 would indicate that the data-generating model is unlikely to generate the class values that are present in the data-set. This means that the probabilities in the data-generating model are not well represented in the data-set. Given our goal to approximate the "true" model of the data using our Bayesian network, this is problematic. This problem should not occur in large enough data-sets.

3.2 Experiment 1: Full independent variables

The first experiment uses a data-set generated by sampling from 11 different independent random probability distributions. As previously mentioned, each probability in the distributions is a random sample from a uniform distribution between 0 and 1. This experiment can help answer what dependency structures cause NB or TAN to outperform each other, by showing how adding dependencies in the predicting network that are not present in the data generation can influence performance. This could be the worst case scenario for TAN, as TAN introduces more dependencies than NB. However, it is also possible for the TAN networks to have all of these introduced dependencies be very weak.

This experiment was repeated on 50 data-sets using 10-fold cross validation instead of the 100 data-sets without cross validation, resulting in 500 total training- and test-sets which were evaluated on their performance metrics. The reason for this was the great variance that was noticed during the testing of the experiment. After repeating the tests using cross validation the variance was greatly reduced, allowing for a more clear analysis of the results.

3.2.1 Results

In Figure 3 the performance of NB and TAN are plotted against each other. We can see that the differences in accuracy and Brier score between the models are small, showing that neither model is much better than the other in being confident on their predictions. The two largest differences in accuracy were split in favor of NB and TAN, both with a difference of 8%.

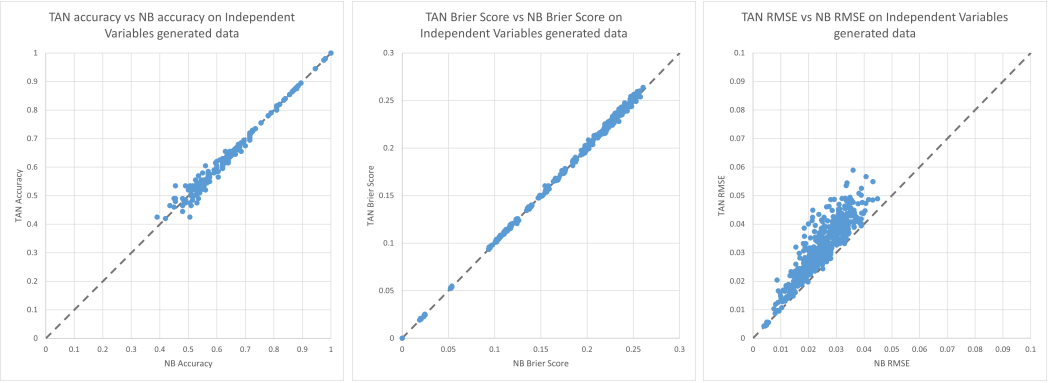
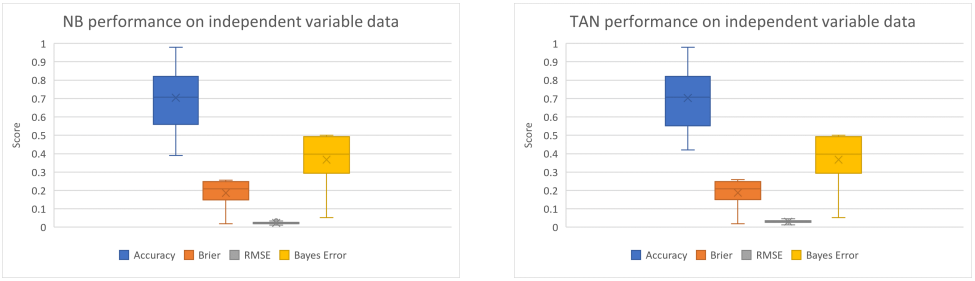


Figure 3: Difference per performance metric per generated data-set between Naive Bayes and TAN. Each point shows the performance of a NB model and a TAN model trained and tested on the same data-set. The grey line indicates NB and TAN to have equal performance, points above the line indicate TAN scoring higher and points below the line indicate NB scoring higher.

Figure 4 shows the distribution of the performance of NB and TAN on each performance metric as a box-plot where the line indicates the median, the coloured boxes show the first and third quartiles, and the whiskers show the variability outside of the upper and lower quartiles. The graphs for NB and TAN look almost identical, with the largest difference being TAN's slightly higher RMSE.



(a) (b)

Figure 4: Performance of NB and TAN on data-sets generated by Bayesian networks with only independent variables. Average metrics NB: 0.725 Accuracy, 0.182 Brier score, 0.025 RMSE. Average metrics TAN: 0.724 Accuracy, 0.183 Brier score, 0.032 RMSE.

3.2.2 Discussion

The most clear difference that can be seen is when looking at the RMSE plot. Here we see that only almost all RMSE scores of the TAN models were higher than those of the NB models, although very slightly. Only 4 out of the 500 results were in favor of TAN. As the only difference between the NB and TAN models are the added dependencies between the feature variables, these added dependencies would have to be the cause of this notable trend.

Another interesting trend to observe is how different the RMSE behaves from the accuracy and Brier score. The latter two show that on some data-sets both models perform very well, while the models perform very bad on others. Whereas the RMSE would make us believe that all models perform relatively well. This can be explained by looking at the Bayes Error in Figure 4. The Bayes Error of the generating class probabilities are very high for these data-sets. This means that the class probabilities are not well represented by the class values in the data-set. This leads to incorrect classifications with estimated probabilities close to the real probabilities for the class. This is also the explanation for both models to also have accuracies below 0.5. As even if all conditional probabilities in the NB and TAN models would be identical to the independent probabilities of the variables, if the class probability is close to 0.5, then the chance of making incorrect classifications is still high due to variance.

The reason for the Bayes Error to be this high has to do with what the class probabilities look like. All variables in the data-generating model are independent with random probabilities, thus the probability of the most likely class is in the range $[0.5, 1]$. When this probability is 0.5, then the Bayes Error will also be 0.5, whereas the Bayes Error would be 0 if this probability is 1. Because this probability is uniformly distributed, the Bayes Error for this experiment is very high.

3.3 Experiment 2: Naive Bayes and TAN

The next two experiments can be used as base-cases to help answer what dependency structures cause NB or TAN to outperform one another, as a NB and a TAN will be used to generate the data-sets. These experiments will show how good these networks perform when they completely match the generating network, which will be used as a base-case to compare to.

3.3.1 Results

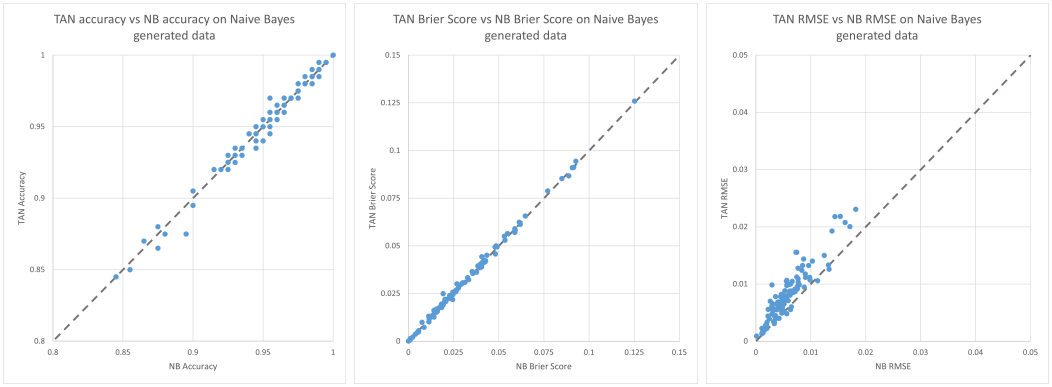


Figure 5: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets.

In Figure 5 the performance of NB and TAN are plotted against each other on data generated by a NB model. Note that the accuracy graph starts at 0.8, as the measured accuracies are all rather high, the graph is more zoomed in than others. Figure 6 shows the performance of NB and TAN on data generated by a TAN model.

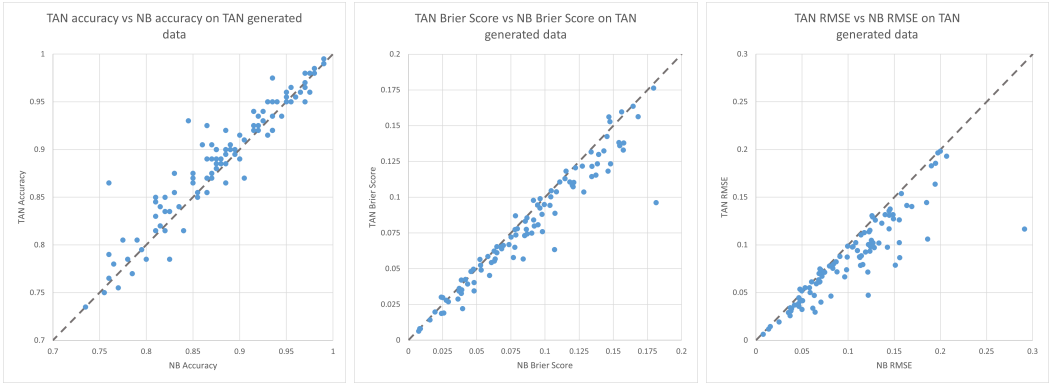
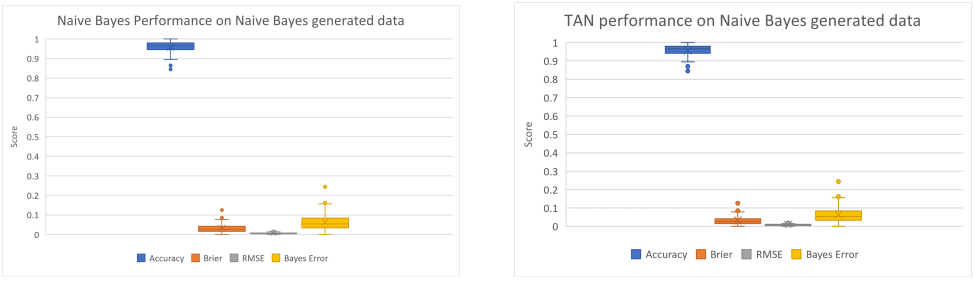


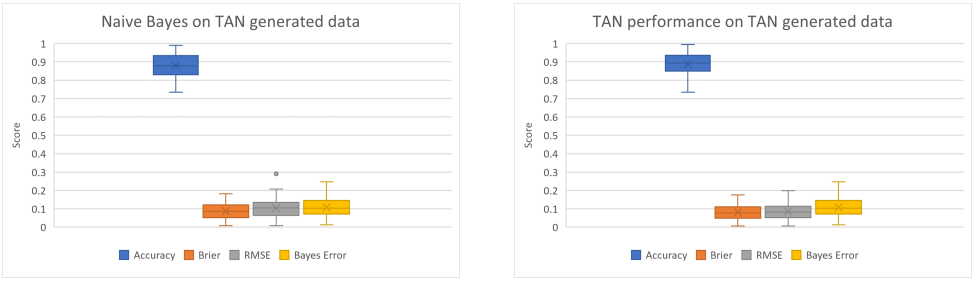
Figure 6: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets. Average metrics NB: 0.957 Accuracy, 0.032 Brier score, 0.006 RMSE. Average metrics TAN: 0.956 Accuracy, 0.032 Brier score, 0.008 RMSE.



(a) (b)

Figure 7: Performance of Naive Bayes and TAN on data-sets generated by Naive Bayesian networks.

Figure 7 shows the performance of NB and TAN on each performance metric for the NB-generated data-set. In Figures 8 we show the performance of NB and TAN on the TAN-generated data-set. By comparing the difference between the two figures, we see that the performance of both NB and TAN is more varied on the TAN generated data-sets.



(a) (b)

Figure 8: Performance of Naive Bayes and TAN on data-sets generated by TAN networks. Average metrics NB: 0.879 Accuracy, 0.088 Brier score, 0.103 RMSE. Average metrics TAN: 0.888 Accuracy, 0.080 Brier score, 0.086 RMSE.

3.3.2 Discussion

We see in Figure 5 NB and TAN perform very similarly, and when looking at the difference in Accuracy, Brier score, or RMSE between the two models per individual experiment not a single difference is statistically significant. This can be explained when thinking about how TAN differs from NB: it has added dependencies between the feature variables, but if these dependencies are not present in the data, then it will add dependencies with no significant influence on the probabilities in the model. This, however, can not be said about data generated from a TAN model, as the "true" model will have significant dependencies between the feature variables, which NB cannot learn but TAN can.

In Figure 6 the performance of NB and TAN are plotted against each other on data generated by a TAN model. Although the performance of NB and TAN still lies very close to each other in this

experiment, there seems to be more of a distinction than with the experiment with NB generated data. TAN seems to be the better performing model in Figure 6, but the averages of the metrics are very close to each other.

An interesting point to mention is that the biggest difference for Brier score and RMSE between NB and TAN is from the same data-set. However, this data-set scores nearly identical on accuracy between NB and TAN, with only a 0.05 difference in favor of TAN.

Overall all metrics are lower compared to the experiment with the NB generated data. This is either caused by the more intricate interactions between the feature variables or the more inaccurate representation of the class probabilities, as can be seen by the slightly higher Bayes Error in Figure 8 compared to Figure 7. The Bayes Error might on average become lower if the size of the data-sets would be made larger. But as the Bayes Error is still far below 0.5, I believe the issue of the larger Bayes Error is not significant enough.

3.4 Experiment 3: Random Bayesian Network

This experiment will use data-sets generated from Bayesian networks with random structures. These random structures are created in three steps:

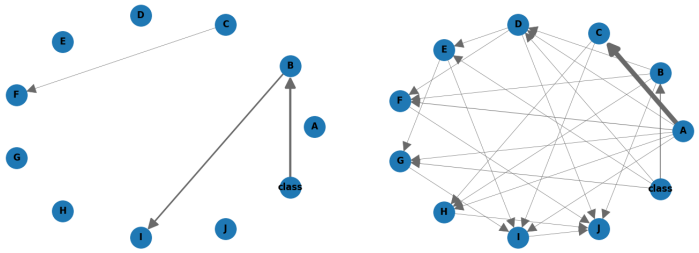
1. Given the probability of an edge between any two variables δ, create an edge between those variables with probability δ.

2. Turn every edge into a directed edge to create a directed acyclic graph (DAG). The graph is guaranteed acyclic by ordering the variables from *class* to feature variable A and then alphabetically until the tenth feature variable J and then choosing the direction from the first in the ordering to the latter in the ordering.

3. Create a Bayesian network from the DAG by filling the nodes with random probabilities sampled from a uniform distribution between 0 and 1.

Examples of networks generated this way can be seen in Figure 9. As δ has a large influence on how these networks are structured, the experiment is run with the following values of δ: 0.1, 0.5, 0.7, and 0.9. The thickness of the edges shows the mutual information (Equation 5) of the two variables. Mutual information is a measure of how much the uncertainty of one variable decreases by knowing the value of the other variable. High mutual information between variables means that knowing the value of one of the variables gives high certainty of the value of the other variable. We say that two variables with high mutual information have a strong influence on each other. Edges with high mutual information are interesting to look at, because they have strong influence within the Bayesian networks.

It is remarkable that most edges in the random networks, as can be seen in Figure 9, have very low mutual information, as this means that most individual edges have very little influence on the probability distributions. This can be seen by looking how thick the edge $A \rightarrow C$ is in Figure 9b, which has a mutual information of 0.633. This is caused by how the random probabilities are generated and mutual information is calculated. To find the $P(x, y)$ in Equation 5, all other parent-variables of the child-variable are summed out. However, the parent-variable for which we are calculating the mutual information has a uniformly random influence on each of these conditional probabilities. With more parent-variables to sum over, this summation is more likely to cancel out the influence a single parent has on the child-variable. It is therefore no surprise that the highest



(a) Example of a Random Network with $\delta = 0.1$

(b) Example of a Random Network with $\delta = 0.5$

(c) Example of a Random Network with $\delta = 0.7$

(d) Example of a Random Network with $\delta = 0.9$

Figure 9: Examples of Random networks generated with different values of δ. The thickness of the edges represents the mutual information (Equation 5) between the variables.

mutual information shown is between a child-variable with only a single parent variable. How this influences the outcome of each experiment is discussed separately for each experiment.

3.4.1 Random Networks with $\delta = 0.1$

3.4.1.1 Results

In Figure 10 we see the performance of NB and TAN on the same data-sets plotted against each other. We can see that the performance of NB and TAN is very similar for these networks. The differences in some of the RMSE scores are relatively bigger than in Accuracy and Brier scores. These outliers in RMSE correspond to the bigger differences in Brier score and Accuracy, all in favor of TAN.

In Figure 11 we show the performance of NB and TAN on each performance metric.

3.4.1.2 Discussion

The performance of both NB and TAN in the experiment with $\delta = 0.1$, see Figure 10, is very similar to the performance on Independent Networks in Figure 3 when it comes to accuracy and Brier score. This is likely due to the data-generating networks being very sparse, as is seen in Figure 9a.

In the experiment with independent variables, TAN always lost to NB on RMSE, which is no longer the case here. This is likely caused by variables that are independent of the class, but dependent on each other, in the data-generating model. Naive Bayes will only consider the

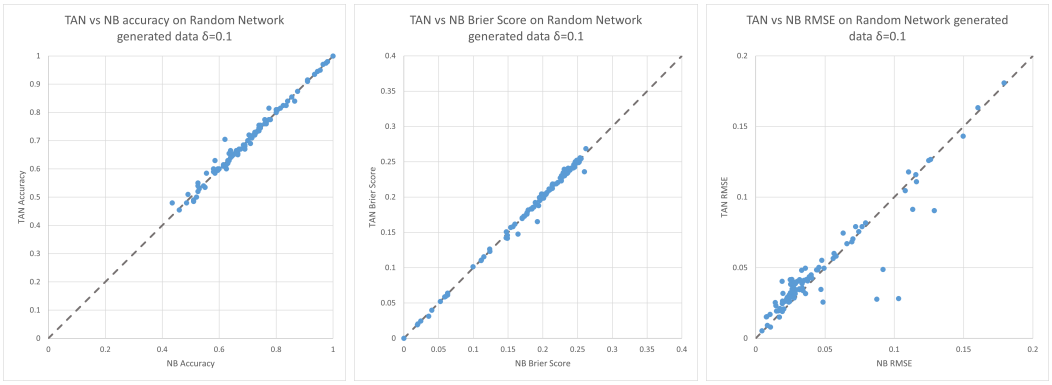
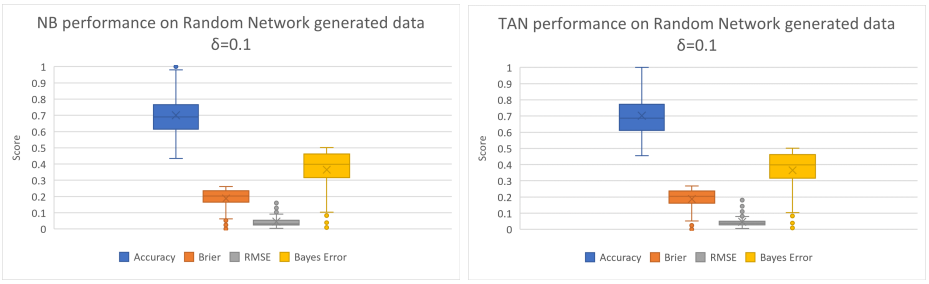


Figure 10: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets.



(a) (b)

Figure 11: Performance of Naive Bayes and TAN on data-sets generated by Random Bayesian networks ($\delta = 0.1$). Average metrics NB: 0.701 Accuracy, 0.188 Brier score, 0.045 RMSE Average metrics TAN: 0.702 Accuracy, 0.187 Brier score, 0.047 RMSE.

patterns, that these variables create, to be connected to the class variable, whereas TAN can detect the influence these variables have on each other and thereafter neglect the influence they have on the class variable, as they are not connected to the class variable.

3.4.2 Random Networks with $\delta = 0.5$

3.4.2.1 Results

Figure 12 shows the performance of NB and TAN on data-sets generated from random Bayesian networks with $\delta = 0.5$. The graph shows very similar results for NB and TAN. However, there are some differentiating data-points, both in favor of NB and in favor of TAN. Most of the data-points not on the equivalence line are in favor of TAN.

In Figure 13 we show the performance of NB and TAN on each performance metric.

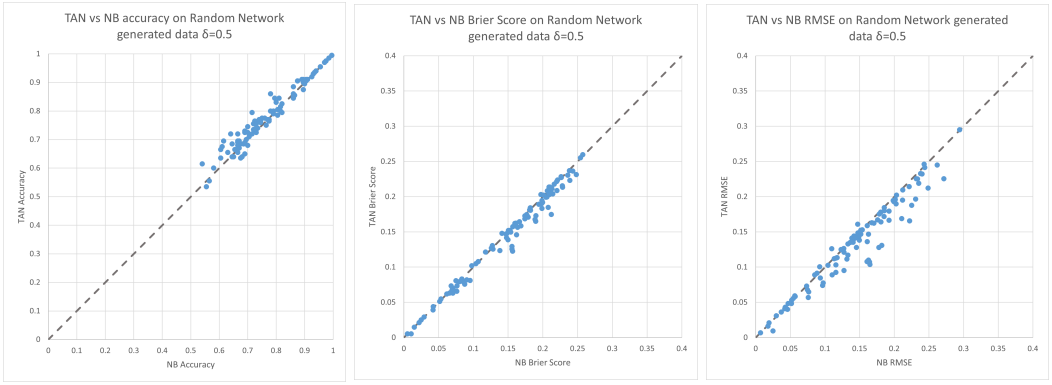
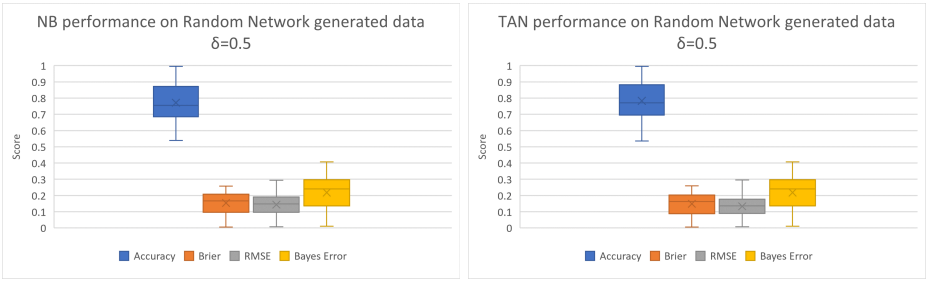


Figure 12: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets.



(a) (b)

Figure 13: Performance of Naive Bayes and TAN on data-sets generated by Random Bayesian networks ($\delta = 0.5$). Average metrics NB: 0.771 Accuracy, 0.154 Brier score, 0.144 RMSE. Average metrics TAN: 0.783 Accuracy, 0.149 Brier score, 0.133 RMSE.

3.4.2.2 Discussion

With a higher value of δ the overall performance of NB and TAN seem to still be similar, as can be seen in Figure 12. As the data-generating graphs are less sparse, see Figure 9b, both models perform better, as more features can be used to infer the probability of the class. Although the difference in accuracy seems to be more spread out in Figure 12 than in Figure 10, there are still no large differences in the metrics. The extreme outliers in RMSE have disappeared, supporting the theory that they were the result of variables independent of the class variable, which is less likely to be the case in these data-generating network.

3.4.3 Random Networks with $\delta = 0.7$

3.4.3.1 Results

In Figure 14 we show the performance of NB and TAN on the same data-sets. We see that there are no Brier score or RMSE results in favor of NB by a significant margin, whereas there three

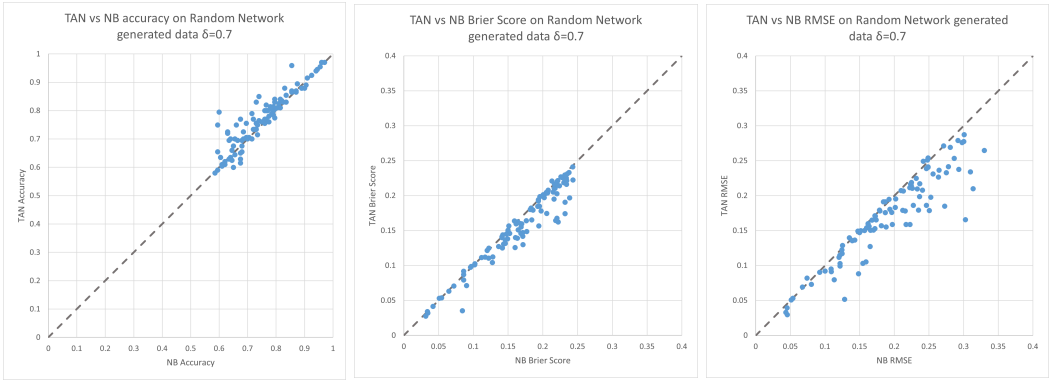
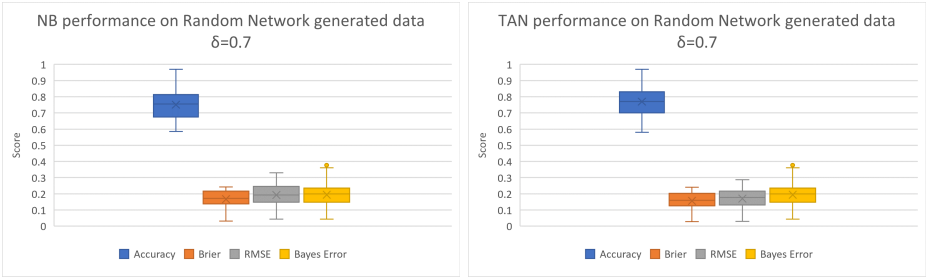


Figure 14: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets.

Accuracy results in favor of NB over TAN. All other results are either a draw between the two techniques, or in favor of TAN.

In Figure 15 we show the performance of NB and TAN on each performance metric.



(a) (b)

Figure 15: Performance of Naive Bayes and TAN on data-sets generated by Random Bayesian networks ($\delta = 0.7$). Average metrics NB: 0.751 Accuracy, 0.167 Brier score, 0.192 RMSE. Average metrics TAN: 0.770 Accuracy, 0.156 Brier score, 0.171 RMSE.

3.4.3.2 Discussion

By increasing δ further to 0.7, the performance of NB and TAN does not seem to change as much as expected, as can be seen in Figure 14 compared to Figure 12. The best performances of all statistics have lowered, but the averages are about the same. There are however more data-sets where TAN clearly outperforms NB. The largest outliers in accuracy, Brier score and RMSE are from the same data-sets and the patterns in these networks are discussed more thoroughly in Section 3.4.5.

The network shown in Figure 9c generated the data-set which is the most in favor of TAN in accuracy and Brier score. Although TAN also outperformed on RMSE on this data-set, it is not the biggest difference for this metric. Not much stands out from this specific model. The class

variable is well connected to the other variables and no singular dependency is much stronger than the other dependencies.

3.4.4 Random Networks with $\delta = 0.9$

3.4.4.1 Results

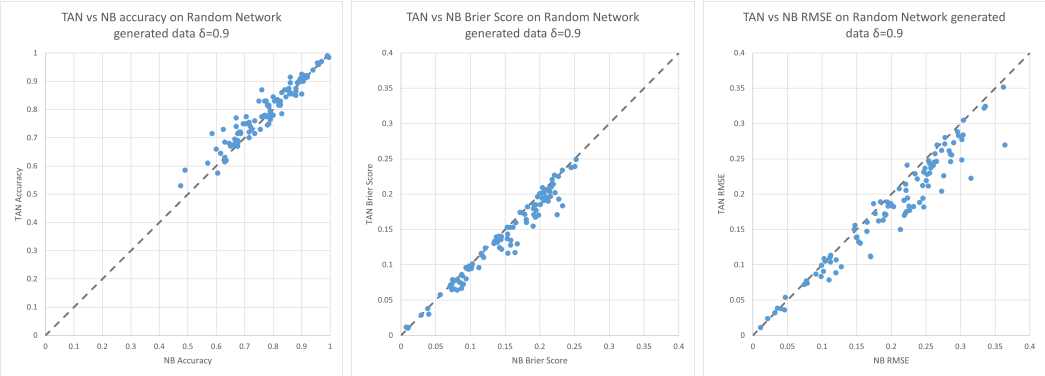


Figure 16: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets.

Figure 16 shows the performance of NB and TAN on random Bayesian networks with $\delta = 0.9$. In the results we see that no single Brier score is significantly in favor of of NB, only one RMSE score is in favor of NB, and only 3 Accuracy scores are in favor of NB. Most Accuracy and Brier scores are not in favor of either technique, whereas most RMSE scores are in favor of TAN.

It is interesting to note that in this experiment, NB scored less than 0.5 Accuracy on two data-sets. This means that the structure it learned from the data-set was detrimental to its performance when compared to only knowing the probability of the binary class. In the case of a classifier only knowing this single probability, its worse case scenario would have an expected Accuracy of 0.5, which is the case if the class probability is 0.5.

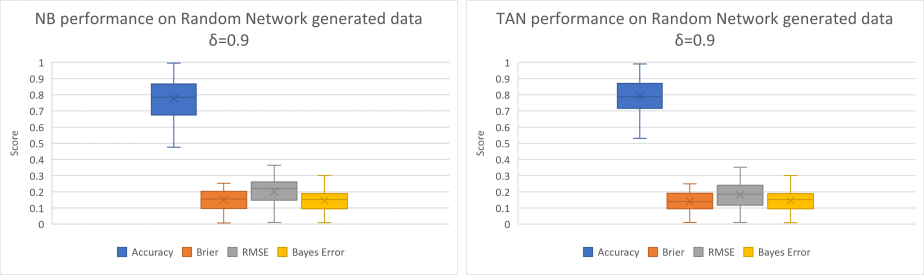
In Figure 17 we show the performance of NB and TAN on each performance metric.

3.4.4.2 Discussion

The largest value of δ used in experiments is 0.9, as the structural variance decreases with larger values of δ. A fully connected graph with $\delta = 1$ is not a realistic Bayesian network, but as these networks are very close to a fully connected graph, these networks can also be considered as not realistic.

We can see the average performance of NB and TAN of the experiments with $\delta = 0.9$ stay almost the same compared to the experiments with $\delta = 0.7$ by comparing Figures 14 and 16. Their Brier scores and RMSE also show similar patterns, but there is a change when it comes to accuracy. The highest measured accuracy for $\delta = 0.9$ is higher than that for $\delta = 0.7$, but also the lowest measured accuracy is lower.

It was not expected for both NB and TAN to still perform this good in terms of the three performance metrics. A likely reason for this is the aforementioned low mutual information between



(a) (b)

Figure 17: Performance of Naive Bayes and TAN on data-sets generated by Random Bayesian networks ($\delta = 0.9$). Average metrics NB: 0.776 Accuracy, 0.150 Brier score, 0.202 RMSE. Average metrics TAN: 0.793 Accuracy, 0.141 Brier score, 0.182 RMSE.

the variables, which is even lower with the higher values of δ, as more parent nodes with random probabilities lead to lower mutual information values. Because all these dependencies have low impact on the probability distributions, both NB and TAN will be able to accurately classify the classes, as not knowing these low impact influences on the class probability does not impact the performance.

The higher accuracies can be attributed to the lower Bayes Error of the data-generating models when comparing Figures 15 and 17. The lower accuracies, however, can not be attributed to the Bayes Error. This shows that the increased amount of low impact dependencies in the networks do make it harder for the more simple networks to perform well.

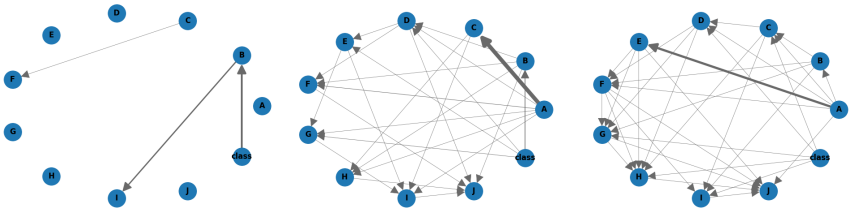
3.4.5 Structural patterns

Although the performance metrics of the experiments with random networks do show a difference in performance between NB and TAN for different levels of connectedness of the data-generating models, there is no clear best or worse case of connectedness. So instead of the varying levels of connectedness, I started looking at the differences in the graph structure of the Bayesian networks and the probability distributions in each data-generating network in each experiment. To visualize this, all data-generating models were drawn with the width of the edges determined by the Mutual Information between those two variables, see Figure 18 for some examples.

The examples in Figure 18 were not chosen randomly, but because they illustrate a trend among the Bayesian networks that generated data-sets on which TAN outperformed NB on all three performance metrics: at least two variables in the network have high mutual information with each other. In most cases these variables do not include the class variable, but in Figure 18i the high mutual information is between one feature variable and the class variable.

As previously mentioned, edges with high mutual information are not frequent in these networks. This general infrequency combined with high frequency within the data-generating models on which TAN outperforms NB leads me to believe this is a possible cause for TAN to outperform NB.

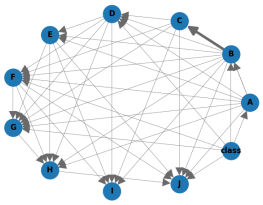
However, not all data-generating networks that generated data-sets on which TAN outperformed NB contained this pattern. Nor did all data-generating networks that did contain this pattern result in a data-set on which TAN outperformed NB on all three performance metrics. More experiments were performed to confirm that the presence of at least two variables with high mutual information



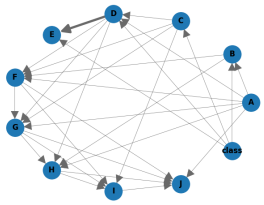
(a) Example of a Random Network with $\delta = 0.1$

(b) Example of a Random Network with $\delta = 0.5$

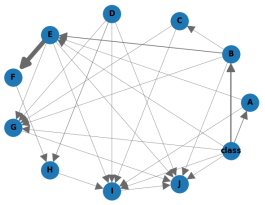
(c) Example of a Random Network with $\delta = 0.5$



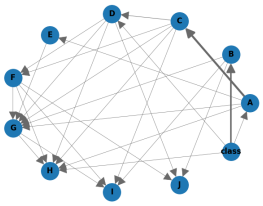
(d) Example of a Random Network with $\delta = 0.7$



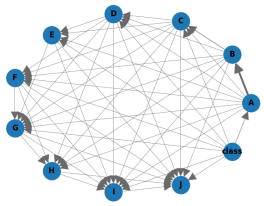
(e) Example of a Random Network with $\delta = 0.5$



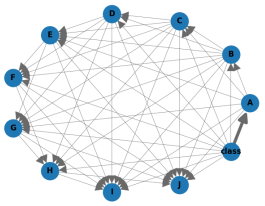
(f) Example of a Random Network with $\delta = 0.5$



(g) Example of a Random Network with $\delta = 0.7$



(h) Example of a Random Network with $\delta = 0.9$



(i) Example of a Random Network with $\delta = 0.9$

Figure 18: Examples of Random Bayesian networks where TAN outperformed NB on accuracy, Brier score and RMSE. The thickness of each edges represent the mutual information between the variables.

with each other results in TAN outperforming NB, which are described in the next section.

3.5 Experiment 4: High Mutual Information

The following three experiments are used to confirm that high mutual information between two feature variables leads to TAN outperforming NB. First, an extreme case with Bayesian networks of only three variables, which will also show how having a high mutual information between two feature variables differs from having a high mutual information with the class variable. TAN is expected to outperform NB with a high mutual information between two feature variables, as the features with a high mutual information will be connected in the TAN model due to the algorithm used to find the connections between the feature variables, see Algorithm 1.

The second experiment with high mutual information will use random Bayesian networks with increased mutual information between at least two feature variables. By increasing the amount of

edges with high mutual information, we expect TAN to outperform NB. The last experiment will show if this theory also holds on data generated by "real world" Bayesian networks.

3.5.1 Three Variable Network

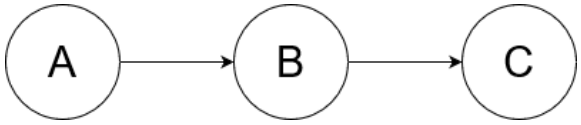


Figure 19: The structure of the Bayesian network used in this experiment, called the *TVN structure*.

The structure of the Bayesian networks used in this experiment is the same for the 500 times it is repeated, which can be seen in Figure 19. The reason for repeating this experiment 500 times is that, because of the networks only having three variables in the network, it takes considerably less time than the experiments with 11 variables in the network. The probabilities for the network were found using Algorithm 2, which was used for each of the 500 times this experiment was repeated. The algorithm returns a single network using the structure in Figure 19, where each node is assigned probabilities, such that the mutual information between nodes A and B and nodes B and C are maximal, while the mutual information between nodes A and C are minimal. These properties will show how having a node with a strong influence on the network separated by a single node affects the classification of the class variable.

Each of the 500 networks are found using Algorithm 2. The algorithm tries 200 probability distributions for node B and 200 for node C, for a total of $200 \times 200 = 40.000$ combinations of probability distributions. The 200 probability distributions for each class are created by sampling 100 random probability distributions from a uniform distribution and the other 100 distributions are created by flipping one half of the distributions around, for example: $P(B|A) = 0.8$ and $P(B|\neg A) = 0.4$, then the flipped distribution would be $P(B|A) = 0.8$ and $P(B|\neg A) = 1 - 0.4 = 0.6$. In the algorithm this is called the $ReverseNegClass$ function. This was done because the mutual information between features is likely to increase when the difference between the conditional probability given $True$ and $False$ increases. We are more likely to find a combination of probability distributions with a high score by calculating the score for the different probability distributions.

Using the steps above, we find a Bayesian network with a high mutual information between nodes A and B, and nodes B and C, while keeping the mutual information between nodes A and C minimal. The chosen function for calculating the score was chosen for maximizing a balance between the mutual information values that are desired to be high and the mutual information that is desired to be low. We use square roots of the mutual information, because we prefer a balance where $MI(A, B) = 0.5$ and $MI(B, C) = 0.5$ over $MI(A, B) = 0.9$ and $MI(B, C) = 0.1$. Both mutual informations here sum to 1, but by taking the square root in the scoring function, the option of both mutual informations being 0.5 is preferred. We wanted the mutual information between nodes A and C to weigh the same as the other part of the scoring function, thus we also take the root here and then double it.

Just like all previous experiments, a data-set the size of 2000 data-points is generated from these Bayesian networks. These data-sets are also split into training-sets containing 90% of the data-points and test-sets containing the remaining 10% of the data-points. However, where this experiment differs from the previous experiments is that the training of NB and TAN is done twice

Algorithm 2 Tree Variable Probability Search

$BestModel \leftarrow Null$
$BestScore \leftarrow 0$
$Model \leftarrow TVNStructure$
$Model.A \leftarrow 0.5$
$ProbabilitiesB \leftarrow List$
$ProbabilitiesC \leftarrow List$
for i in 1..100 **do**
    $ProbabilitiesB.Append(Random.Uniform(0, 1, (2, 2))$
    $ProbabilitiesC.Append(Random.Uniform(0, 1, (2, 2))$
end for
for i in 1..100 **do**
    $ProbabilitiesB.Append(ReverseNegGiven(ProbabilitiesB[i]))$
    $ProbabilitiesC.Append(ReverseNegGiven(ProbabilitiesC[i]))$
end for
for Prob B in $ProbabilitiesB$ **do**
    $Model.B \leftarrow B$
    for Prob C in $ProbabilitiesC$ **do**
        $Model.C \leftarrow C$
        $Score \leftarrow \sqrt{MI(A,B)} + \sqrt{MI(B,C)} - 2 \times \sqrt{MI(A,C)}$
        if $Score > BestScore$ **then**
            $BestModel \leftarrow Model$
            $BestScore \leftarrow Score$
        end if
    end for
end for
Return $BestModel$

for both: once where node B is the class and once where node C is the class. The experiment where node B is the class variable is called the *Class in Middle* experiment, and the experiment where node C is the class variable is called the *Class on leaf* experiment. If node A was chosen instead of node C, the results would probably not differ much, because the directions of the arrows in the network do not matter as the joint distribution stays the same.

By repeating the experiment for both the scenario where both edges with high mutual information are connected to the class variable and the scenario where one of the edges is between two feature variables, we can see how the separation of an edge with high mutual information influences the performance of NB and TAN. The results of this experiment can be seen in Figure 20.

3.5.1.1 Results

Figure 20 shows the results of NB and TAN on the Tree Variable Networks, divided on having the class variable in the middle or at the leaf node. This division for the plot was chosen as it provides clear clusters of points on the graphs.

We can clearly see in Figure 20 that the *Class on Leaf* experiment's performance is worse than the *Class in Middle* experiment, both for NB and TAN. Another thing we can clearly see is that TAN outperforms NB on the *Class on Leaf* experiment on all three performance metrics. And we

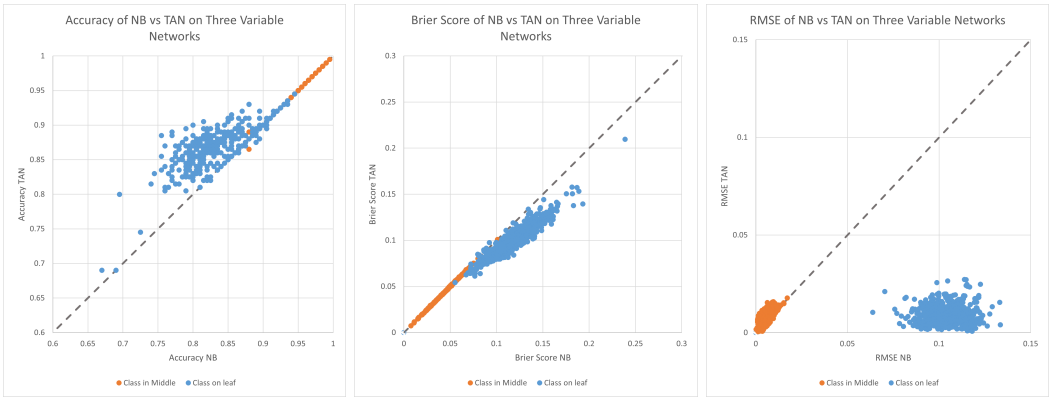


Figure 20: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets.

can also see that NB and TAN perform very similarly on the *Class in Middle* experiment, except that NB outperforms TAN on the RMSE metric.

The *Class on Leaf* experiment shows that having two feature variables with high mutual information leads to TAN outperforming NB on all three performance metrics in most cases. And in these cases where TAN does not outperform NB, both models perform at least equally as good.

In Figure 21 we show the performance of NB and TAN on *Class on Leaf* generated data and *Class in Middle* generated data. We can see that the Bayes Error is higher for the *Class on Leaf* experiment.

3.5.1.2 Discussion

The clear difference in performance between the two experiments shows that having more strong edges connected to the class variable has a positive influence on the performance of both NB and TAN.

The better performance of TAN over NB in the *Class on Leaf* experiment can be attributed to TAN having the data-generating network's structure as a subset of its own structure: TAN has all the edges in the TVN structure, but with an added edge between node A and node C. NB in this experiment does not have the "true" edge between nodes A and B.

A similar case can be made for why NB performs better than TAN in terms of RMSE in the *Class in Middle* experiment. In this experiment, the NB structure is the same as the TVN structure: the class is connected to the two feature variables and there are no other edges present. In this experiment, TAN has an added edge between nodes A and C, which is not present in the data-generating network. This leads to TAN being very accurate in performance, but making slightly bigger mistakes in terms of the predicted probability in comparison to NB. But as the structure of TAN still has the "true" structure as a subset of its own structure, it might be able to learn the class probabilities just as accurate as NB given a larger data-set.

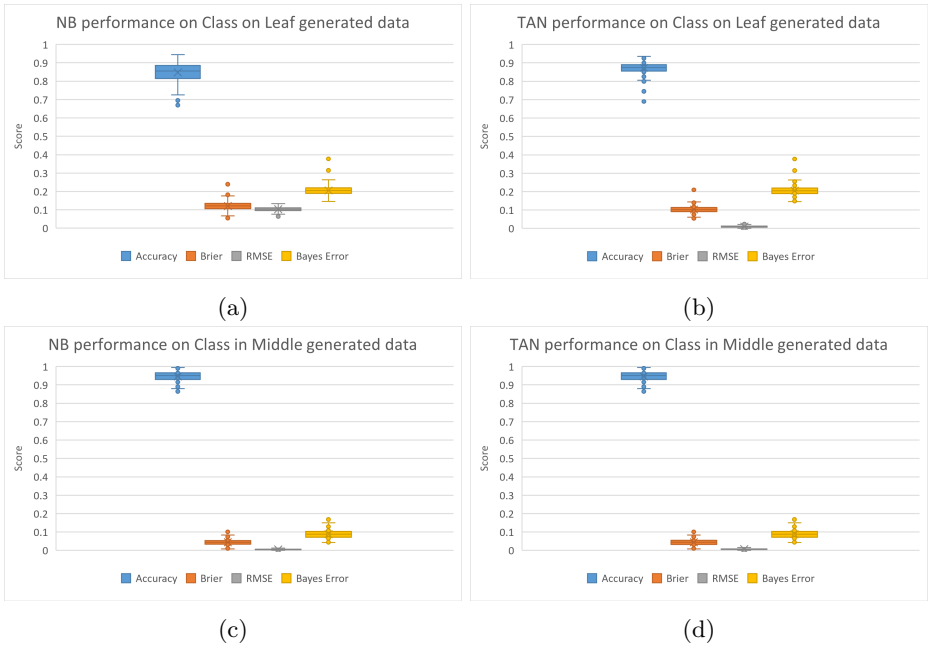


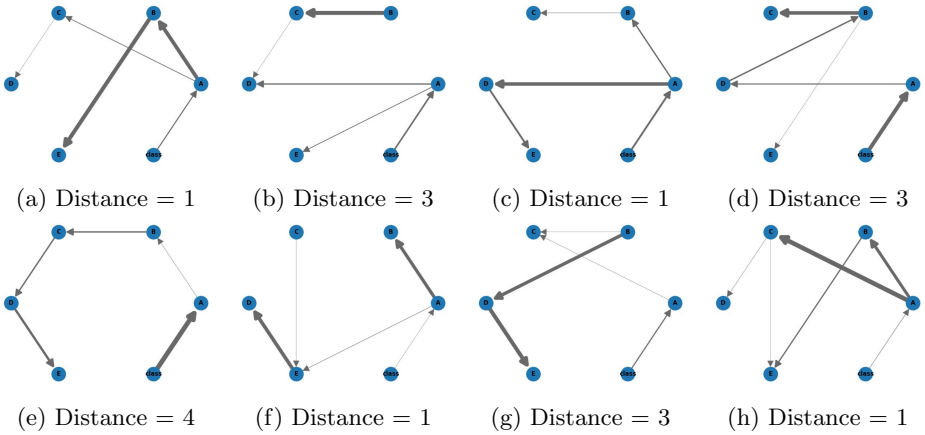
Figure 21: Performance of Naive Bayes and TAN on data-sets generated by Three Variable Bayesian networks. Average metrics NB on *Class on Leaf*: 0.848 Accuracy, 0.120 Brier score, 0.104 RMSE. Average metrics NB on *Class in Middle*: 0.945 Accuracy, 0.044 Brier score, 0.005 RMSE. Average metrics TAN on *Class on Leaf*: 0.873 Accuracy, 0.103 Brier score, 0.010 RMSE. Average metrics TAN on *Class in Middle*: 0.945 Accuracy, 0.044 Brier score, 0.007 RMSE.

3.5.2 Mutual Information Boosted Random Bayesian Network

To find if the presence of high mutual information between feature variables also causes TAN to outperform NB in more complex networks, random Bayesian networks with $\delta = 0.5$ are used, similarly to the experiments in Section 3.4.2. The reason for the selection of $\delta = 0.5$ is that with less parent variables, it becomes easier to find probability distributions that result in high mutual information between two feature variables. This experiment was repeated 100 times, resulting in 100 different data-generating models and 100 different data-sets on which NB and TAN were trained. There are however differences with the experiments in Section 3.4.2, which are as follows:

- Each network has at least one edge between two feature variables with a mutual information of 0.5 or higher.
- The networks only have 5 feature variables.
- Only one of the feature variables is connected to the class variable. Making it similar to the *Class on Leaf* experiment, but now with a larger network.

Examples of such networks can be found in Figure 22. Having only feature variable A connected to the class allows for more feature variables to have a high mutual information with each other,



(a) Distance = 1 (b) Distance = 3 (c) Distance = 1 (d) Distance = 3

(e) Distance = 4 (f) Distance = 1 (g) Distance = 3 (h) Distance = 1

Figure 22: Examples of Random Bayesian networks with $\delta = 0.5$ with at least one edge between feature variables with mutual information of 0.5 or higher. The thickness of each edges represent the mutual information between the variables.

as the features will have less possible parent variables. This also allows the experiment to better test if it matters if the high mutual information feature variables are further away from the class variable than in the previous experiment. If the class variable would be connected to more than one feature variable, then the range of distances from the class variable would be reduced. The minimal distance for this is 1, as at least one edge has to be traversed from the class node to arrive at a feature variable. The maximal distance can be 4, as can be seen in Figure 22e. The direction of the edges does not matter when computing the distance.

The amount of feature variables had to be reduced due to time constraints on the search for probability distributions that have high mutual information.

Lastly, the network structures and probability distributions of the variables were found using the algorithm in Algorithm 3.

Algorithm 3 Network Search

$BestModel \leftarrow Null$
$BestTotalMI \leftarrow 0$
while $BestModel = Null$ **do**
  $model \leftarrow RandomModelConnected(\delta = 0.5)$
  $AllMI \leftarrow CalculateMutualInformations(model)$
  $TotalMI \leftarrow Sum(AllMI)$
  $BestFeatureMI \leftarrow MaxFeatureMI(model, AllMI)$
  if $MaxFeatureMI \geq 0.5$ & $TotalMI \geq BestTotalMI$ **then**
    $BestModel \leftarrow model$
    $BestTotalMI \leftarrow TotalMI$
  end if
end while
Return $BestModel$

3.5.2.1 Results

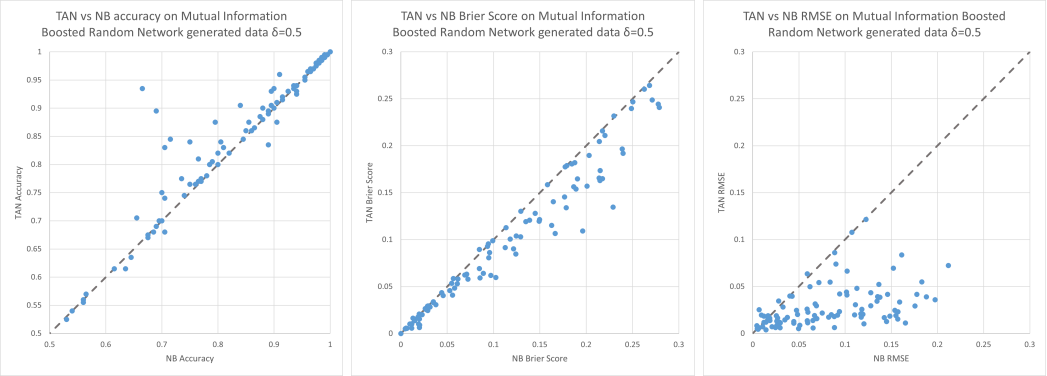
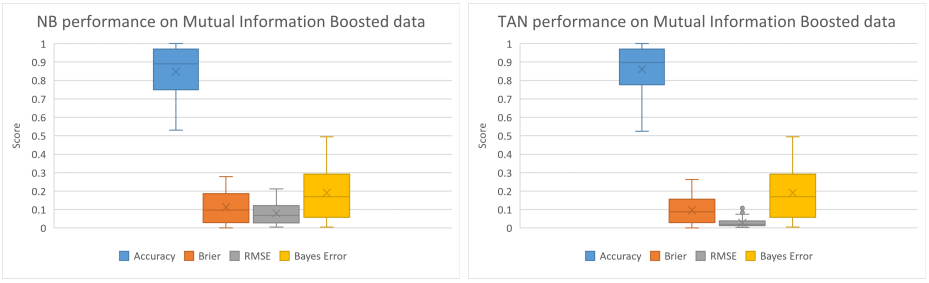


Figure 23: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets.

Figure 23 shows the performance of NB and TAN on data-sets generated by Mutual Information Boosted Bayesian networks. We see that there are some big differences in Accuracy, while most points are near equal performance. For Brier score and RMSE we see that TAN almost always outperforms NB.

In Figure 24 we show the distribution of the performance of NB and TAN on the data-sets. Here we see that The average performance of both models are very close and the variability of Accuracy and Brier score to also be very close. But it shows that the RMSE for TAN is lower than that of NB.



(a) (b)

Figure 24: Performance of Naive Bayes and TAN on data-sets generated by Mutual Information Boosted Bayesian networks ($\delta = 0.5$). Average metrics NB: 0.848 Accuracy, 0.112 Brier score, 0.080 RMSE. Average metrics TAN: 0.862 Accuracy, 0.097 Brier score, 0.028 RMSE.

The distance from the class node to the closest feature variable with an edge that has a high mutual information with another feature variable was measured to see if the position of the edge with high mutual information influences how NB and TAN perform. To better visualize this in Figure 25, the data-generating networks were divided into categories based on the performance metrics of NB and TAN on the data-sets generated by the networks as follows:

- *TAN wins all*: all networks in this category generated data-sets on which TAN outperformed NB on all three performance metrics. This category contained 22 networks.
- *TAN wins two*: all networks in this category generated data-sets on which TAN outperformed NB on two of three performance metrics and the last performance metric was a draw, as the result falls within the standard error. This category contained 24 networks.
- *Draw*: all networks in this category generated data-sets on which NB and TAN performed equally good on at least two out of three performance metrics. The third performance metric was allowed to be a win for TAN, as this was only the case for 2 networks. This category contained 45 networks.
- *NB wins one or more*: all networks in this category generated data-sets on which NB outperformed TAN on at least one performance metric. This category has the least strict rules for including networks, yet still only contained 9 networks.

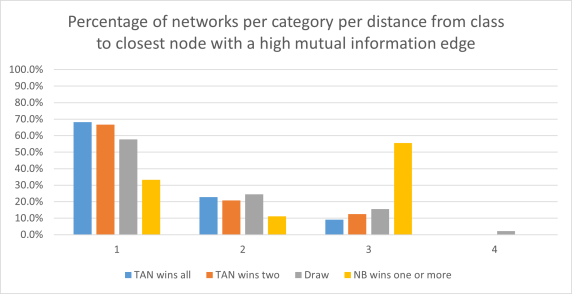


Figure 25: Percentage of data-generating networks per distance from class node to closest feature variable with an edge that has a high mutual information with another feature variable. Category sizes: TAN wins all = 22 networks, TAN wins two = 24 networks, Draw = 45 networks, NB wins one or more = 9 networks.

3.5.2.2 Discussion

As can be seen in Figures 23 and 25, TAN significantly outperformed NB in 46 out of 100 cases. However, the results seem to be less in favor of TAN than the *Class on Leaf* experiment in Section 3.5.1. Figure 25 shows that the networks with the high mutual information edge closer to the class variable are more likely to generate a data-set on which TAN outperforms NB. The combination of the *Class on Leaf* experiment being more in favor of TAN than this experiment and the closer high mutual information edges leading to better performance of TAN compared to NB, leads to the belief that TAN is likely to outperform NB when there are feature variables with high mutual information whose dependency properties are close with the class variable.

3.5.3 Real world network

All previous experiments have been performed on data-sets generated from Bayesian networks that were created using some form of randomness in the network structure and probability distributions.

In order to extend the findings of how mutual information influences NB and TAN's performances to the real world, a similar experiment as the used experiments has been performed.

For this experiment three Bayesian networks are used for generating the data-sets: the Alarm network ([3], see Figure 26a), the Child network ([24], see Figure 26b, and the Insurance network ([4], see Figure 26c). This allows us to be confident in the dependency properties that will be present in the generated data-sets. In practice, an expert on the domain of a data-set would be able to indicate some known dependency properties between the feature variables. These dependency properties in combination with either predicted strength of edges by the domain expert, or by calculating the mutual information from the data-set, would give similar insight on the dependency properties in a data-set as knowing the exact data-generating Bayesian network gives.

For deciding which node in each of the networks would be used as the class variable two things were considered: does the network have the desired dependency property of strong mutual information edges close to the class and does the variable make sense to be the class given the context of the network. This lead to the following choices for class variables: *Hypovolemia* in Alarm, *Disease* in Child, and *Accident* in Insurance. Next I will explain why these nodes were chosen.

The Alarm network([3]) encodes medical knowledge about a patient to calculate the probability of 8 different diagnoses: *Hypovolemia*, *Left Ventricular Failure*, *Anaphylaxis*, *Insufficient Analgesia*, *Pulm. Embolus*, *Intubation*, *Kinked Tube*, and *Disconnection*. Given the context of the network, these would be the variables that make the most sense to be the class variable. Of these variables, only 4 show the desired dependency property: *Hypovolemia*, *Left Ventricular Failure*, *Intubation*, and *Kinked Tube*. Of these variables, the variable with the least skewed marginal probability distribution was chosen as class variable: *Hypovolemia*. The probability of *Hypovolemia* without knowing the value of any feature variables is $P(Hypovolemia = True) = 0.2$ in the Alarm network. The marignal probability of the *Hypovolemia* variable and the other chosen class variables can be found in Figure 27.

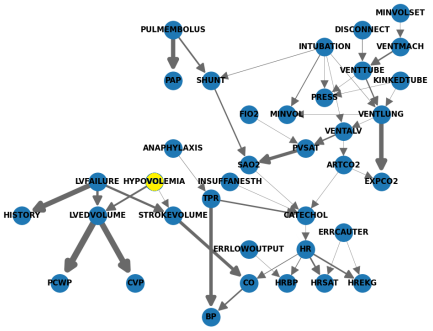
The Child network([24]) is another network encoding medical knowledge to calculate the probability of different diseases of a child patient. These probabilities lie in the *Disease* node, which also happens to show the desired dependency property. Therefore the Disease variable is chosen as the class variable.

The Insurance network([4]) is used for estimating the expected claim cost for a car insurance policyholder. This would make *MedicalCost*, *LiabilityCost*, or *PropertyCost* the most logical class variable given the context. Of these variables, only the *MedicalCost* variable shows the desired dependecy property. However, this variable has a very skewed likelihood with the probability $P(MedicalCost = Thousand) = 0.928$, which would make the margin for error of the classification task too small. Therefore another important variable in the network was chosen as class variable: *Accident*. In this node the probabilities for how likely accidents with different levels of severity are, which is important for estimating if a insurance policyholder will claim damages.

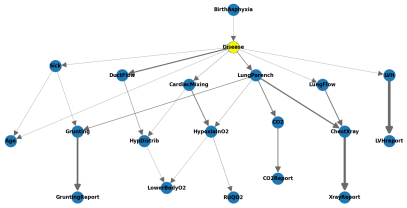
For the experiment, a data-set of 2000 data-points is generated from each of the three Bayesian networks. Naive Bayes and TAN are trained on 90% of the data-set and tested on the remaining 10%.

3.5.3.1 Results

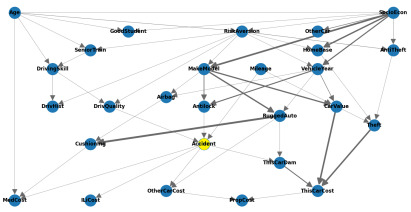
In Figure 28 we see the results of all three metrics in a single graph, including error bars indicating the standard error of the means of each score. TAN outperforms NB on the Child and Insurance data-sets on all three performance metrics with a significant margin. On the Alarm data-set neither



(a) The Alarm Bayesian network([3]). The thickness of each edges represent the mutual information between the variables in the network.



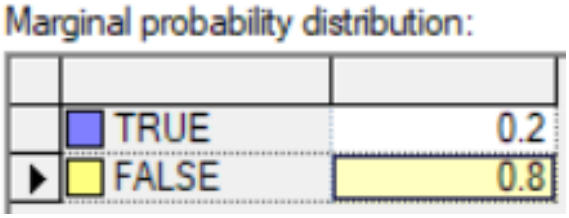
(b) The Child Bayesian network([24]). The thickness of each edges represent the mutual information between the variables in the network.



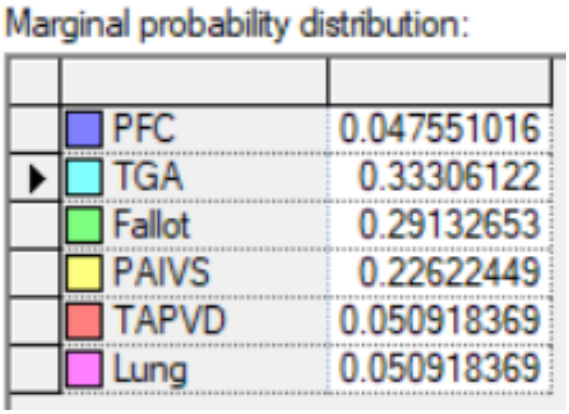
(c) The Insurance Bayesian network([4]). The thickness of each edges represent the mutual information between the variables in the network.

Figure 26: The Alarm, Child and Insurance Bayesian networks. The thickness of each edges represent the mutual information between the variables. The yellow node indicates the class variable.

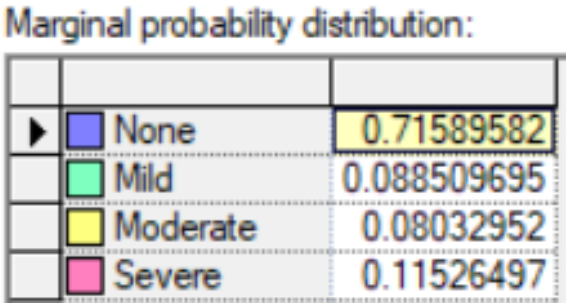
NB or TAN outperforms the other, as the difference in performance falls within the standard error of the mean.



(a) The marginal distribution of the Hypovolemia variable in the Alarm network([4]).



(b) The marginal distribution of the Disease variable in the Child network([4])..



(c) The marginal distribution of the Accident variable in the Insurance network([4]).

Figure 27: The marginal distribution of the Hypovolemia, Disease and Accident variables from the Alarm, Child and Insurance Bayesian networks.

3.5.3.2 Discussion

TAN outperforming NB on two out of three data-sets helps support the belief that feature variables with high mutual information help TAN outperform NB. TAN and NB performing equally as good on one of the data-sets shows that feature variables with high mutual information do not always

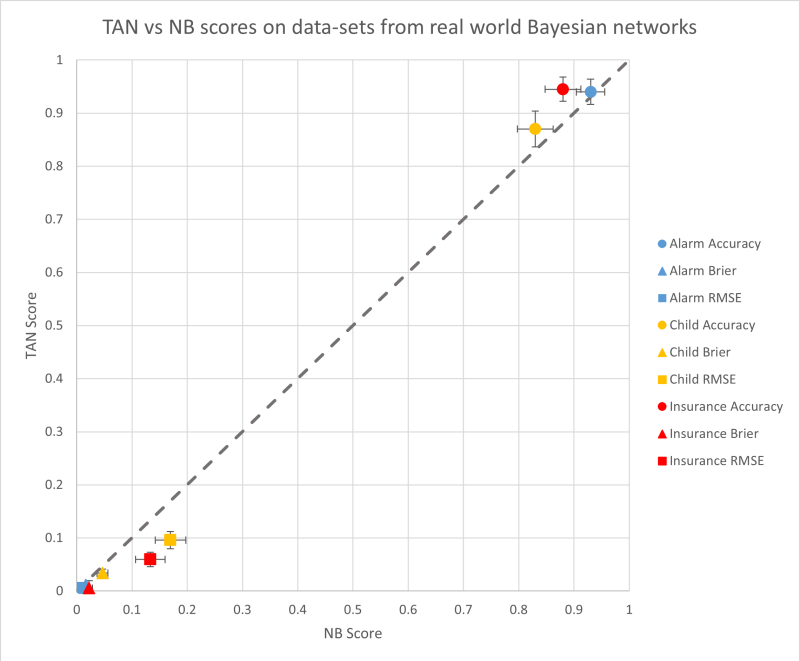


Figure 28: Difference per performance metric per generated data-set between Naive Bayes and TAN on the same data-sets. Error bars show the standard error of the mean for every statistic. Bayes Errors: Alarm: 0.013, Child: 0.055, Insurance: 0.030.

lead to TAN outperforming NB.

Preferably this experiment would be repeated for more than three real world Bayesian networks. Repeating a similar experiment with more networks would make the support for the found belief stronger, which is a topic for future research. This is explained further in Section 4. Furthermore, without the domain knowledge required for making assumptions about dependency properties in a data-set, it was not possible to repeat the experiment for the data-sets present in Table 1.

4 Conclusion

The goal of this research is to explain the difference in performance between naive Bayesian classifiers and tree-augmented Bayesian network classifiers. We have explained that research on tree-augmented Bayesian network classifiers do not focus on *why* these classifiers sometimes perform better or worse than naive Bayesian classifiers. We have shown that in the related works and our experiments that in most cases the TAN classifiers perform better than naive Bayesian classifiers. Because of this, finding what causes the difference in performance between these classifiers is an interesting topic.

In our experiments we analyzed the performance of naive Bayes and tree-augmented Bayesian networks trained on data-sets generated by different Bayesian networks. First Bayesian networks with random structures were used for this analysis. From this we have learned that both types of

classifiers work better if there is some structure within the data, as can be seen from the results of the random structure experiment with $\delta = 0.5$. When the structure in the data becomes more complex, which is the case in the experiments with $\delta = 0.7$ or 0.9, TAN outperforms NB more often than with more simple structures. This answers our first sub-question: *how do the two classifiers compare?*

By analyzing the networks on which TAN performed better than NB, we have found that TAN is more likely to outperform NB when there are at least two feature variables with high mutual information present in the network. We tested this hypothesis using experiments on data-sets that were generated from Bayesian networks with this dependency pattern. We tested how features with high mutual information affect the performance of NB and TAN when they are directly and indirectly connected to the class variable in the data-generating Bayesian network. The result of these experiments show that TAN is more capable of accurately estimating the class probability using feature variables that are not directly correlated to the class variable. Afterwards, we tested the hypothesis on data-sets generated by real-world Bayesian networks. The results of this experiment show that feature variables with high mutual information makes TAN more likely to outperform NB, but it is not a guarantee that TAN will outperform NB. This answers our second sub-question: *Can performance differences be explained by dependencies in the data?*

We have given different hypotheses to answer our sub-questions, which combined can help answer our research question: *what explains the difference between naive Bayesian classifiers and tree augmented Bayesian network classifiers?*. We did not solve the answer completely and only the future might tell if this ever will be the case. Future research on this research question could look into why our hypothesis on the influence of at least two variables with high mutual information in the data-generating Bayesian network does not always lead to TAN outperforming NB or what other factors can help explain the difference in performance, such as the prior probability of the class variable.

Another possible direction for future research is to look at the loss of information that features contain about the class when assuming the Naive Bayes model, instead of mutual information. Rish [23] has shown that there is a stronger correlation between the accuracy of Naive Bayes and this loss of information. It is unknown if TAN has this same correlation between accuracy and the loss of information that features contain about the class when assuming the TAN model, nor is it known if this loss of information can be used to predict whether NB or TAN will outpeform the other.

Other research for better understanding the behavior of TAN would be repeating similar experiments as de Wachter [25] for finding the worst case scenario of TAN. When repeating the experiments of de Wachter, TAN is expected to perform much better on his Linked Networks, as these use the same structure as a TAN model. However, how TAN will behave on the Complete Networks that he used is unknown.

References

[1] Ankur Ankan. *pgmpy*. URL: `https://pgmpy.org/`. (accessed 01/12/2022).

[2] A Bazila Banu and Ponniah Thirumalaikolundusubramanian. "Comparison of Bayes classifiers for breast cancer classification". In: *Asian Pacific Journal of Cancer Prevention: APJCP* 19.10 (2018), p. 2917.

[3] Ingo A Beinlich, Henri Jacques Suermondt, R Martin Chavez, and Gregory F Cooper. "The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks". In: *AIME 89*. Springer, 1989, pp. 247–256.

[4] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. "Adaptive probabilistic networks with hidden variables". In: *Machine Learning* 29.2 (1997), pp. 213–244.

[5] Glenn W Brier et al. "Verification of forecasts expressed in terms of probability". In: *Monthly weather review* 78.1 (1950), pp. 1–3.

[6] Jesús Cerquides and Ramon López De Mántaras. "Tractable Bayesian learning of tree augmented naıve Bayes models". In: *ICML*. 2003, pp. 75–82.

[7] Shenglei Chen, Geoffrey I Webb, Linyuan Liu, and Xin Ma. "A novel selective naıve Bayes algorithm". In: *Knowledge-Based Systems* 192 (2020), p. 105361.

[8] Jie Cheng and Russell Greiner. "Comparing Bayesian network classifiers". In: *Uncertainty in Artificial Inteligence, 1999* (1999).

[9] Imme Ebert-Uphoff. *Measuring connection strengths and link strengths in discrete Bayesian networks*. Tech. rep. Georgia Institute of Technology, 2007.

[10] Python Software Foundation. *Python*. URL: `https://www.python.org/`. (accessed 01/12/2022).

[11] Nir Friedman, Dan Geiger, and Moises Goldszmidt. "Bayesian network classifiers". In: *Machine learning* 29.2 (1997), pp. 131–163.

[12] David J. Hand and Keming Yu. "Idiot's Bayes: not so stupid after all?" In: *International Statistical Review / Revue Internationale de Statistique* 69.3 (2001), pp. 385–398.

[13] Liangxiao Jiang, Zhihua Cai, Dianhong Wang, and Harry Zhang. "Improving tree augmented naive Bayes for class probability estimation". In: *Knowledge-Based Systems* 26 (2012), pp. 239–245.

[14] Liangxiao Jiang, Dianhong Wang, Zhihua Cai, and Xuesong Yan. "Survey of improving naive Bayes for classification". In: *Advanced Data Mining and Applications* (2007), pp. 134–145.

[15] Ludmila Kuncheva and Zoe Hoare. "Error-dependency relationships for the Naıve Bayes classifier with binary features". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.4 (2008), pp. 735–740.

[16] Ludmila I Kuncheva. "On the optimality of naıve Bayes with dependent binary features". In: *Pattern Recognition Letters* 27.7 (2006), pp. 830–837.

[17] Yuguang Long, Limin Wang, and Minghui Sun. "Structure extension of tree-augmented naive Bayes". In: *Entropy* 21.8 (2019), p. 721.

[18] Inc. pandas via NumFOCUS. *pandas: pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.* URL: `https://pandas.pydata.org/`. (accessed 01/12/2022).

[19] NumPy. *NumPy The fundamental package for scientific computing with Python*. URL: `https://numpy.org/`. (accessed 01/12/2022).

[20] Judea Pearl. "Bayesian networks". In: *UCLA Department of Statistics Papers* (2011).

[21] Judea Pearl. *Bayesian networks: A model of self-activated memory for evidential reasoning*. Tech. rep. 1985.

[22] Irina Rish. "An empirical study of the naive Bayes classifier". In: *IJCAI 2001 workshop on empirical methods in artificial intelligence* 3.22 (2001), pp. 41–46.

[23] Irina Rish, Joseph Hellerstein, and Jayram Thathachar. "An analysis of data characteristics that affect naive Bayes performance". In: *IBM TJ Watson Research Center* 30 (2001), pp. 1–8.

[24] David J Spiegelhalter. "Learning in probabilistic expert systems". In: *Bayesian statistics* 4 (1992), pp. 447–465.

[25] Matthijs de Wachter. "On the errors introduced by the naive Bayes independence assumption". In: *Master Thesis Artificial Intelligence of Utrecht University* (2018).