

Choosing the Best Bayesian Classifier: An Empirical Study

Stuart Moran*

Yulan He[†]

Kecheng Liu*

Abstract—It is often difficult for data miners to know which classifier will perform most effectively in any given dataset. Usually an understanding of learning algorithms is combined with detailed domain knowledge of the dataset at hand to lead to the choice of a classifier. We propose an empirical framework that quantitatively assesses the accuracy of a selection of classifiers on different datasets, resulting in a set of classification rules generated by the J48 decision tree algorithm. Data miners can follow these rules to select the most effective classifier for their work. By optimising the parameters used for learning, a set of rules were learned that select with 78% accuracy (with 0.5% classification accuracy tolerance), the most effective classifier.

Index Terms—Bayesian networks; Data mining; Classification; Search algorithm; Decision tree.

I INTRODUCTION

The past 20 years have seen a dramatic increase in the amount of data being stored in electronic format. The accumulation of this data has taken place at an explosive rate and it has been estimated that the amount of information in the world doubles every two years [1]. Within this ocean of data, valuable information lies dormant.

Data mining uses statistical techniques and advanced algorithms to search the data for hidden patterns and relationships. However, as data expands and the importance of data mining increases, a problem emerges. There are many different classifiers and many different types of dataset resulting in difficulty in knowing which will perform most effectively in any given case. It is already widely known that some classifiers perform better than others on different datasets. Usually an understanding of learning algorithms is combined with detailed domain knowledge of the dataset at hand for the choice of classifier. Experience and deep knowledge will of course affect the choice of the most effective classifier - but are they always right? It is always possible that another classifier may unknowingly work better.

*Informatics Research Centre, University of Reading, Reading RG6 6WB, UK Tel/Fax: 44-118-3786606/3784421 E-mails: {stuart.moran,k.liu}@henley.reading.ac.uk

[†]Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes MK7 6AA, UK Tel/Fax: 44-1908-653800/653169 Email: y.l.he.01@cantab.net

In deciding which classifier will work best for a given dataset there are two options. The first is to put all the trust in an expert's opinion based on knowledge and experience. The second is to run through every possible classifier that could work on the dataset, identifying rationally the one which performs best. The latter option, while being the most rigorous, would take time and require a significant amount of resources, especially with larger datasets, and as such is impractical. If the expert consistently chooses an ineffective classifier, the most effective classification rules will never be learned, and resources will be wasted. Neither method provides an efficient solution and as a result it would be extremely helpful to both users and experts, if it were known explicitly which classifier, of the multitude available, is most effective for a particular type of dataset.

We therefore propose a framework to quantify which of a selection of classifiers is most effective at mining a given dataset in terms of accuracy (for our experiments, speed was not a focus). From this assessment, the J48 learning algorithm [2] is used to generate a series of rules in the form of a decision tree which then enables data miners to select the most accurate classifier given their particular dataset. (To the best of our knowledge, no other work has been attempted in such a way.)

This paper is organised as follows. Section II presents the proposed empirical framework for automatically selecting the best Bayesian classifier. Section III discusses the performance evaluation metrics. Section IV presents the experimental results from the 39 datasets selected. Finally, Section V concludes the paper.

II PROPOSED FRAMEWORK

The empirical framework for automatically selecting the best classifier is depicted in Fig. 1, which consists of four main processes, *Dataset Categorisation*, *Classifier Training*, *Results Sampling*, and *Classification Rules Generation*.

The ultimate aim of this research was to find a set of rules that would allow a user to predict which is the best classifier for use on their dataset. It was decided that this could be attempted by applying a learning algorithm to the initial analysis of which classifier performs best

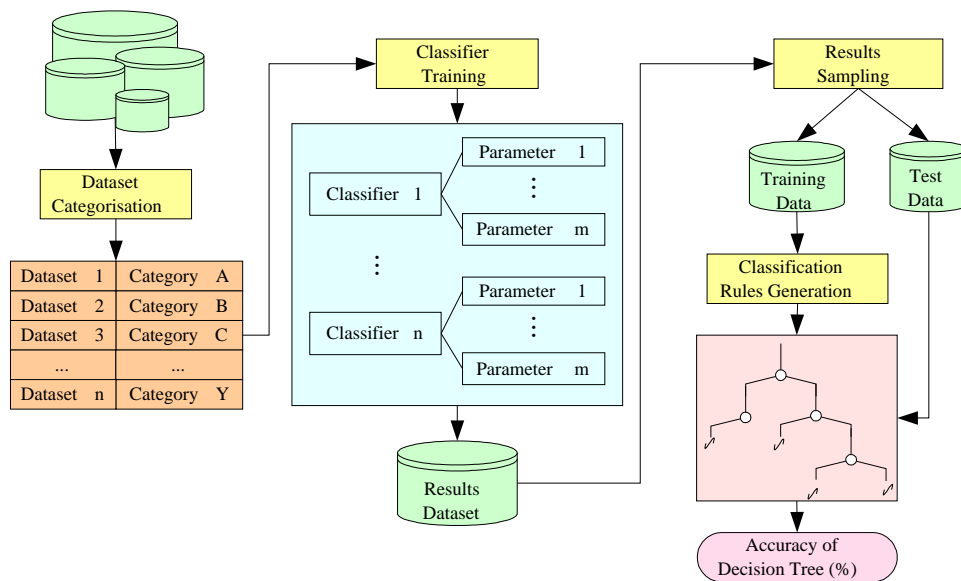


Figure 1: An empirical framework for automatically selecting the best classifier.

for each dataset. If a rational system for selecting the optimum classifier on the basis of the type of dataset to be analysed could be identified, then the process of data mining could be made significantly more effective. For the ease of use, a set of rules in a decision tree format is desirable. To create such a tree the effective learning algorithm J48 [2] was selected. The following will discuss each of the four processes in details.

A Dataset Categorisation

Categorising the datasets has two advantages. First, it allows an identification of which classifier performs best for a particular type of dataset. Secondly, it provides a means to select a representative sample against which a learning algorithm is tested.

A total of 39 datasets were selected from the UCI Machine Learning repository¹, the WEKA Web site², and the Promise repository³. The characteristics of each dataset and whether univariate outlier detection has been performed are listed in Table 1. Given that the datasets are of different sizes, in terms of both attributes and instances, it was difficult to use a generic categorising system. Nonetheless, having looked at the data as a whole it was decided that the median values of total instances and attribute numbers should be used as the basis on which to categorise them. These were 286 instances and 16 attributes. Using the median values meant that an equal number of large and small datasets, as defined by the values, would sit either side of the boundary.

Using these values as thresholds; the datasets were ini-

tially split into four groups: those datasets with > 286 and those with ≤ 286 instances, and within each of these groups those datasets with > 16 attributes and those with ≤ 16 . This represented the datasets entirely in terms of their structure, without reference to the type of attributes. Using only the structure of a dataset kept the categorisation simple. No datasets were ever 100% numeric, due to the fact that the class must always be a categorical value when used with Bayesian classifiers [2]. For this reason it was decided that three sub-categories were to be created, one which housed all the datasets that were 100% categorical and two others which were 50% – 99% categorical and 1% – 49% categorical, respectively. This means that there exist 12 different categories of dataset. This is summarised in Table 2. Some will hold more than others, but at least any samples taken will be representative, using the categorical system as shown in Table 2.

Table 2: The criteria for categorising datasets based on the number of instances, attributes and the % of attributes that are categorical.

> 286 Instances					
> 16 Attributes			< 16 Attributes		
100%	$\geq 50\%$	$< 50\%$	100%	$\geq 50\%$	$< 50\%$
≤ 286 Instances					
> 16 Attributes			< 16 Attributes		
100%	$\geq 50\%$	$< 50\%$	100%	$\geq 50\%$	$< 50\%$

B Classifier Training

After datasets have been categorised, various classifiers are then trained on them. We mainly focused on Bayesian network [3] (BN) classifiers. Altogether 8 BN classifiers have been investigated including Naïve Bayes (NB) [4],

¹<http://archive.ics.uci.edu/ml/>

²<http://www.cs.waikato.ac.nz/ml/weka/>

³<http://promisedata.org/>

Table 1: List of details of all the datasets used.

Dataset	Inst	Attr	Categorical (%)	Numeric (%)	Missing Values	Univariate Outliers	Source
Balance Scale	625	4	25.00	75.00	0	None	UCI
Balloons	20	5	100.00	0.00	0	NA	UCI
Breast Cancer	286	10	100.00	0.00	9	NA	UCI
Bridges	95	12	66.67	33.33	88	Yes	UCI
Car	1728	7	100.00	0.00	0	NA	UCI
Chess - krvk	28056	7	57.14	42.86	0	None	UCI
Chess - krvskp	3196	36	100.00	0.00	0	NA	UCI
CM1	498	22	4.5	95.45	0	Yes	Promise
Congress Voting	428	17	100.00	0.00	392	NA	UCI
Contact Lenses	24	5	100.00	0.00	0	NA	WEKA
Credit Screening	690	16	62.50	37.50	67	Yes	UCI
Cylinder Bands	502	39	48.72	51.28	352	Yes	UCI
Disease	10	5	100.00	0.00	0	NA	UCI
Ecoli	336	9	22.22	77.78	0	Yes	UCI
Eucalyptus	736	20	35.00	65.00	448	Yes	WEKA
Flag	194	30	10.00	90.00	0	Yes	UCI
Grub-Damage	155	9	77.78	22.22	0	None	WEKA
Horse-Coli	268	23	69.57	30.43	1927	Yes	UCI
Image	210	16	6.25	93.75	0	Yes	UCI
Ionosphere	351	35	2.86	97.14	0	None	UCI
KC1	2109	22	4.55	95.45	0	Yes	Promise
KC2	522	22	4.55	95.45	0	Yes	Promise
Lymphography	148	19	84.21	15.79	0	Yes	UCI
Monk	122	7	100.00	0.00	0	NA	UCI
Mushroom	8124	22	100.00	0.00	2480	NA	UCI
Nursery	12960	8	100.00	0.00	0	NA	UCI
PC1	1109	22	4.55	95.45	0	Yes	Promise
Pasture	36	23	8.70	91.30	0	None	WEKA
Post-Operative	90	9	88.89	11.11	3	None	UCI
Segment-Challenge	1500	20	5.00	95.00	0	Yes	WEKA
Soybean-large	301	36	2.78	97.22	684	Yes	UCI
Soybean-small	47	36	2.78	97.22	0	None	UCI
Squash-stored	52	25	16.00	84.00	6	None	WEKA
Squash-unused	52	24	16.67	83.33	39	None	WEKA
Tae	151	6	16.67	83.33	0	None	UCI
Tic-Tac-Toe	958	10	100.00	0.00	0	None	UCI
Titanic	2201	4	100.00	0.00	0	None	WEKA
Weather	14	5	60.00	40.00	0	None	WEKA
White-Clover	63	32	15.63	84.38	0	Yes	WEKA

Averaged One Dependence Estimator (AODE) [5], Tree-Augmented Naïve Bayes (TAN), BN with different structure learning algorithms such as K2 (BN-K2), Genetic Search (BN-GS), Simulated Annealing (BN-SA), Greedy Hill Climber (BN-HC), and Repeated Hill Climber (BN-RHC). The following will briefly discuss each of them in turn. More details about these well-known algorithms can be found in the given references.

Bayesian networks are probabilistic directed acyclic graphs consisting of a set of variables (nodes) and a set of directed edges (arcs) between variables. Each variable has a finite set of mutually exclusive states and each variable A_k with parents B_1, \dots, B_n is assigned the probability distribution table $P(A_k|B_1, \dots, B_n)$. In the network, each node represents an attribute and the edges represent the cause-effect relationships between them. Each node has a node probability table which stores the joint probability distribution for all the possible states of the attribute, given all of its parents. These are then used to predict the class probabilities for any given instance. To calculate the probability distribution:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i|\text{parent}(X_i)) \quad (1)$$

where n is the total number of states of an attribute, X_i

is a state of the attribute X and $\text{parent}(X_i)$ represents the set of parent nodes for state X_i .

When using Bayesian Networks, difficulty arises when a considerable amount of attributes and classes are looked at. An enormous number of instances are needed to estimate the probabilities for accurate classification. Naïve Bayes [4](NB) is the simplest Bayesian learning classifier and it assumes each attribute to be equally important and independent of the others in an instance, given the class. To calculate the probability distribution:

$$P(X, y) = P(y)P(X|y) = P(y) \prod_{d=1}^n P(X_d|\text{parent}(X_d)) \quad (2)$$

One Dependence Estimators (ODE) are weaker-independence variants of the naïve Bayes classifier, where one attribute is chosen to be a parent of all the others, in addition to the class attribute. Averaged One Dependence Estimators (AODE) [5] overcome the attribute independence assumption of naïve Bayes. A one-dependence classifier is built for each attribute, in which an attribute is set to be the parent of all the other attributes, and by averaging over all possible ODE's a highly accurate classification can be achieved. To cal-

culate the probability distribution:

$$P(y, X) = \frac{\sum_{i:1 \leq i \leq n \& F(x_i) \geq m} P(y, x_i) P(X|y, x_i)}{|\{i : 1 \leq i \leq n \& F(x_i) \geq m\}|} \quad (3)$$

Where $F(x_i)$ is the number of instances having attribute-value x_i and is used to enforce the limit m on the conditional probability estimate.

Tree augmented naïve Bayes (TAN) [6] is an extension of the naïve Bayes classifier. TAN removes the naïve Bayes assumption that all the features are independent. The dependencies between variables, other than the class, are also taken into account. By adding a tree-like structure to naïve Bayes, each attribute may have, in addition to the class attribute, one other parent from amongst the others.

The aforementioned methods can be viewed as a search for a structure that fits the data best. Starting with a graph with nodes for each attribute but no edges, each algorithm uses a search method to add edges to the graph, based on a metric that checks whether the new structure is better than the old one. If so, the edge is kept and the process is continued. The algorithm will stop when there is no better structure (i.e. the best structure has been found). This means however, that the structure depends on the type of search and the metric used to measure its quality, and so each algorithm should produce a different result.

Bayesian Network classifiers use different search algorithms to find the optimal way of representing the data. The accuracy of classification will be different depending on which particular search algorithm is used by the classifier. Thus the following search algorithms have been investigated and treated as classifiers in their own right.

- **K2 (BN-K2)** [7]. K2 is a score-based greedy search algorithm for learning Bayesian networks from data. It maximises the probability of an optimal graph topology, given a dataset, by using a Bayesian score to rank different graphs. The algorithm is restricted by an order on the variables.
- **Genetic Search (BN-GS)** [2]. Genetic Algorithms (GAs) are search algorithms based on the idea of exploring a set of solutions represented by a population of individuals by using the principle of natural selection. This results in finding the best solution, or structure in this case, to be used.
- **Simulated Annealing (BN-SA)** [8]. Simulated annealing (SA) is a general purpose combinatorial optimisation algorithm. The algorithm has been inspired by the process of annealing metal, to harden it. The basic idea of simulated annealing is to assign to the problem a temperature (a control parameter) and think of the cost of a solution as an energy level. The solution then corresponds with the state of the metal; as the temperature is lowered, the solution becomes more defined, with less moves or states available to it to change to.
- **Greedy Hill Climber (BN-HC)** [2]. Imagine that all of the possible solutions to a given problem are represented as a three-dimensional landscape. HC will follow the graph from node to node, always increasing the value of the solution, until a local maximum is reached. This Bayes Network learning algorithm uses a hill climbing algorithm adding, deleting and reversing arcs. The search is not restricted by an order on the variables (unlike K2).
- **Repeated Hill Climber (BN-RHC)** [2]. Repeated Hill Climber searches for Bayesian network structures by repeatedly generating a random network and applying to it the hill climbing algorithm mentioned above. The best network found is returned. The advantage of this algorithm is that when HC gets stuck at a node, a new node is chosen at random and HC is restarted. This is repeated k times and the algorithm returns the best maximum found.

The 8 classifiers described above are then applied to the 39 fully prepared datasets (See Table 1). Each algorithm has a variety of parameter settings available and all possible combinations are tested (29 in total). The software used to complete this testing is the WEKA [9] workbench. The number of parameters investigated were however restricted by what was provided by WEKA, and are listed in Table 3. Each of the parameters is explained below [2]:

- *UseKernelEstimator* (k). The kernel estimator, when set to true, is designed for use with numeric attributes rather than a normal distribution. This means that a difference in performance should be seen depending on the number of numeric values available.
- *UseSupervisedDiscretisation* (Sd). Supervised discretisation is used to convert numeric attributes to nominal ones.
- *initAsNaiveBayes* (iNb). When set to true (default), the initial structure used for learning learning is a naïve Bayes Network. When set to false, an empty network is used as the initial network structure.
- *markovBlanketClassifier* (Mb). After a network structure is learned, a correction is applied ensuring all nodes in the network are a part of the Markov blanket of the classifier node.
- *RandomOrder* (R). The order of the nodes in the network is random, as opposed to the order found in the dataset.

- *useArcReversal (Ar)*. When set to true, an arc between two nodes is reversed by adding arcs from the opposite parents respectively.
- *useTournamentSelection (Ts)*. This determines the method of selecting a population in a genetic search algorithm. When set to true, tournament selection is used where two networks are picked at random and the best solution is allowed to continue. When set to false, the top scoring structures are selected.

AODE had no optional parameter settings. A problem with the AODE classifier is that it can only be applied to datasets that are completely categorical, meaning any numeric data must be discretised via the minimum description length method. During preliminary tests of the WEKA software, it was often found that no intervals were created and all of the variables were included for each class. This would place an unfair bias on the data, and be non-representational of the original dataset, which would affect the classification accuracy. For this reason, it was decided to not use discretisation, meaning the AODE classifier could only be used on 12 datasets instead of 39 datasets. After *Classifier Training*, a large ‘results’ dataset (1073 instances) is formed consisting of the accuracy of the classifiers learned for all the combinations of the parameters tested.

C Results Sampling

Using a variety of sampling techniques, different samples are taken from the results dataset and stored in a smaller test dataset. The sampling techniques used is stratified sampling where the datasets are divided into sub-populations (in this case categories) and then a sample is taken from each sub-category to make a larger sample set. The sample in this case is almost artificial as it is specifically chosen; the advantage is that the sample is guaranteed to be representative of the category where it comes from. The purpose of the *Results Sampling* step is to extract the representative samples from each of the 39 datasets to form a test set. The remaining instances then form a training set to derive the classification rules as will be described in the subsequent subsection.

D Classification Rules Generation

The last step is to use a learning algorithm to analyse the results generated. Here, the J48 [2] decision tree algorithm is used to generate classification rules in the form of a decision tree. A set of rules will have been created that assigns the most effective classifier available (of those tested) to a particular dataset (See Figure 4).

The accuracy of these rules is easily discovered by applying the appropriate decision tree to the test data (sampled from the results dataset). This is possible as we

know which classifier performed the best on each dataset within the sample from the *Classifier Training* step.

In summary, the framework generates two main outputs:

- *A method for choosing the best classifier*. Using the decision tree learned by J48 from the ‘results’ data, the user can simply follow the binary tree answering the relevant questions about their dataset. Eventually they will reach a leaf node which will tell them the best classifier to use.
- *The most effective Bayesian classifier for a specific category of dataset*. Given that a method for categorising datasets was created, it is possible to find which classifier performs the best in any given category. Data miners could then consider which category their particular dataset belongs to and know which classifier performs best for that category. This provides a practical human solution to the problem of choosing the best classifier in addition to the decision tree produced by J48.

While only a limited subset of classifiers and dataset types could be tested here, this research shows the feasibility and the potential of the proposed framework. With a more comprehensive analysis, the final set of rules generated can be expected to be more successful.

III PERFORMANCE EVALUATION METRICS

A statistical analysis is carried out to assess the performance of the different classifiers for comparison. These statistics are explained here.

- *Accuracy*. In our experiments, accuracy measures how well each classifier performs during the cross-validation.
- *Mean absolute error*. An absolute error is the range of possible values in terms of the unit of measurement e.g. 10cm±0.5cm. The mean absolute error is then the weighted average of all the absolute errors found from cross validations.
- *Relative absolute error*. This measure is a ratio of the mean absolute error of the learning algorithm over the mean absolute error found by predicting the mean of the training data. The lower the percentage, the better the performance of the classifier compared to just predicting the mean.

Once each classifier was run against every dataset, the statistics of their performance are collated against each dataset in the form of a table. This detailed statistical comparison is a comprehensive way of analysing the

Table 3: The parameter settings each BN classifier makes use of is indicated by a 'X'.

Parameters	NB	TAN	BN-K2	BN-HC	BN-GS	BN-SA	BN-RHC
KernelEstimator (k)	X						
S-Discretisation (Sd)	X						
Intiate As NB (iNb)			X	X			X
markovBlanket (Mb)		X	X	X		X	X
RandomOrder (R)			X				X
ArcReversal (Ar)				X			
TournamentSelect(Ts)					X		

performance of different classifiers. However, such a detailed analysis can be difficult to interpret. Therefore a more robust method was created to represent the relative performance of each classifier. For each dataset, every algorithm was ranked by assigning a score to its performance. The lowest accuracy was given 1, and this value was incremented for each algorithm that performed better. If two algorithms had the same performance, they were both given the same score. The best performing algorithm for each individual dataset could then be easily identified.

Performance was evaluated and scores assigned on the basis of the accuracy of each of the algorithms. This is a good indicator of performance at a glance. However it was found that many of the algorithms had a similar accuracy, and so the mean absolute errors were also taken into account. The algorithm with the highest accuracy and the lowest mean absolute error (MAE) was then ranked as the best performing algorithm. Given the situation where two different algorithms produce the same accuracy and mean absolute error, the relative absolute error was used. If this value turned out to be the same, then the algorithms were considered to be equally effective.

IV EXPERIMENTAL RESULTS

A The Best Algorithm for Each Category

The criteria for each category of data has been discussed in Section A. In the first instance it was necessary to place the datasets into categories so that a fair sample of datasets could be taken for subsequent analysis by a learning algorithm. An additional benefit of categorising the dataset is that the best performing classifier for each type of dataset can be identified. This can be used as a guide for data miners who wish to quickly identify a classifier for the type of dataset they are working with. Each algorithm for each category was ranked on the scoring algorithm described in Fig. ???. This allows the performance of the different algorithms to be easily visualised. An example of the performance of each algorithm with various parameter settings for the Category A datasets is

shown in Fig. 2. The best performing parameter settings for each category are summarised in Table 4.

Table 4: Best performing classifier with its parameter settings per category.

Category	Best Performing
A	BN-GS and BN - SA
B	TAN
C	BN-HC (-iNb)
D	No datasets
E	TAN
F	TAN
G	TAN, BN-RHC and BN-HC
H	TAN
I	No datasets
J	TAN
K	BN-HC (-iNb) and BN-HC (-iNb Mb Ar)
L	TAN

For categories B, E, H, J and L it is clear that TAN performs the best. Categories A, C, F, G, and K are discussed below.

- *Category A.* BN-GS and BN-SA were proved to work the best on category A datasets. TAN performed extremely poorly on this category. It is interesting to note that all parameter settings for the NB classifier performed the same in this category.
- *Category C.* It is interesting to note that BN-HC (-iNb) performed the best, but BN-RHC (-iNb) performed the worst. When used in conjunction with -iNb on these datasets, it can be seen that Mb actually reduces the accuracy for BN-HC, BN-k2 and BN-RHC. This reduction, as a result of Mb used with -iNb is the only exception to the rule that Mb with -iNb improves performance. Both this and the difference between BN-RHC and BN-HC, is highly likely to be a result of the content of the data, as opposed to the structure itself.
- *Category F.* In this category TAN performed the best, and BN-k2(-iNb R) performed the worst. When Mb is used on its own accuracy is decreased

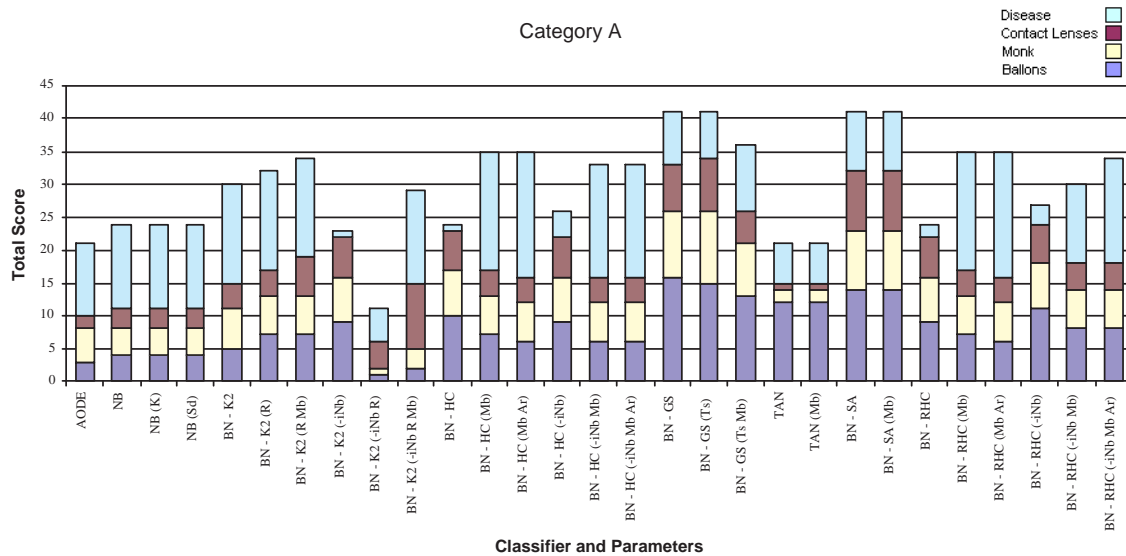


Figure 2: The performance of each classifier with various parameter settings on category A.

but when used with -iNb it increases. This adds to the evidence that the use of Mb with -iNb does improve performance.

- *Category G.* In this category TAN, BN-RHC and BN-HC all performed equally best which is an interesting result considering the search algorithms work in different ways. NB parameters all performed the same again, suggesting some correlation between performance and the percentage of the dataset that is categorical. AODE performed very well on this dataset and is highly likely due to the fact that the dataset is 100% categorical.
- *Category K.* BN-HC(-iNb Mb Ar) performed the best in this category. TAN actually performed quite badly on this category when taking into account its overall excellent performance on the other categories. Ar makes no real difference to BN-HC and BN-RHC. After an in depth comparison, it is also clear that the following patterns hold true for the majority of datasets when put into categories.

In most cases, the use of the Markov Blanket on a dataset improves the results. One of the few exceptions to this is the BN-GS, where accuracy appears to reduce. When used without using a naïve Bayes structure initially (-iNb), it was found that for BN-HC and BN-RHC the accuracy drops. The only exception was on category C, where BN-HC (-iNb) performed the best.

Arc Reversals did have a small positive effect, but in general added no improvements to the accuracy and in some cases did worse than if it had not been used. So it can be

suggested that it is better not to use Arc Reversal. BN-HC and BN-RHC should in general not be used without an initial naïve Bayes structure.

For the BN-K2 classifier, when used with -iNb and a random ordering R, its accuracy decreases significantly. When looking at how the algorithm performs when only used with -iNb, it is clear that a random ordering should not be used.

B The Best Performing Classifier Overall

The algorithms were all ranked using the scoring method described in Section III. The comparison of the overall performance of the algorithms based on their ranking is shown in Fig. 3. It is clear from this analysis that the TAN classifier performed the best with approximately 20 more points than the next best performing, the BN-HC classifier with parameters (-iNb and Ar).

The BN-RHC classifier did show an overall better accuracy than the BN-HC, which was expected, even if there was only a slight difference. However when BN-RHC was initialised without a naïve Bayes structure (-iNb), it performed significantly worse than when the Markov Blanket (Mb) was found. For most of the search algorithms, using the Markov Blanket improved the results, regardless of other parameters.

This improvement occurred because all the nodes were ensured to be conditionally independent given the class, and would effectively be forced into a naïve Bayes-like structure. This meant that the Markov blanket was the only knowledge needed to predict the behaviour of the

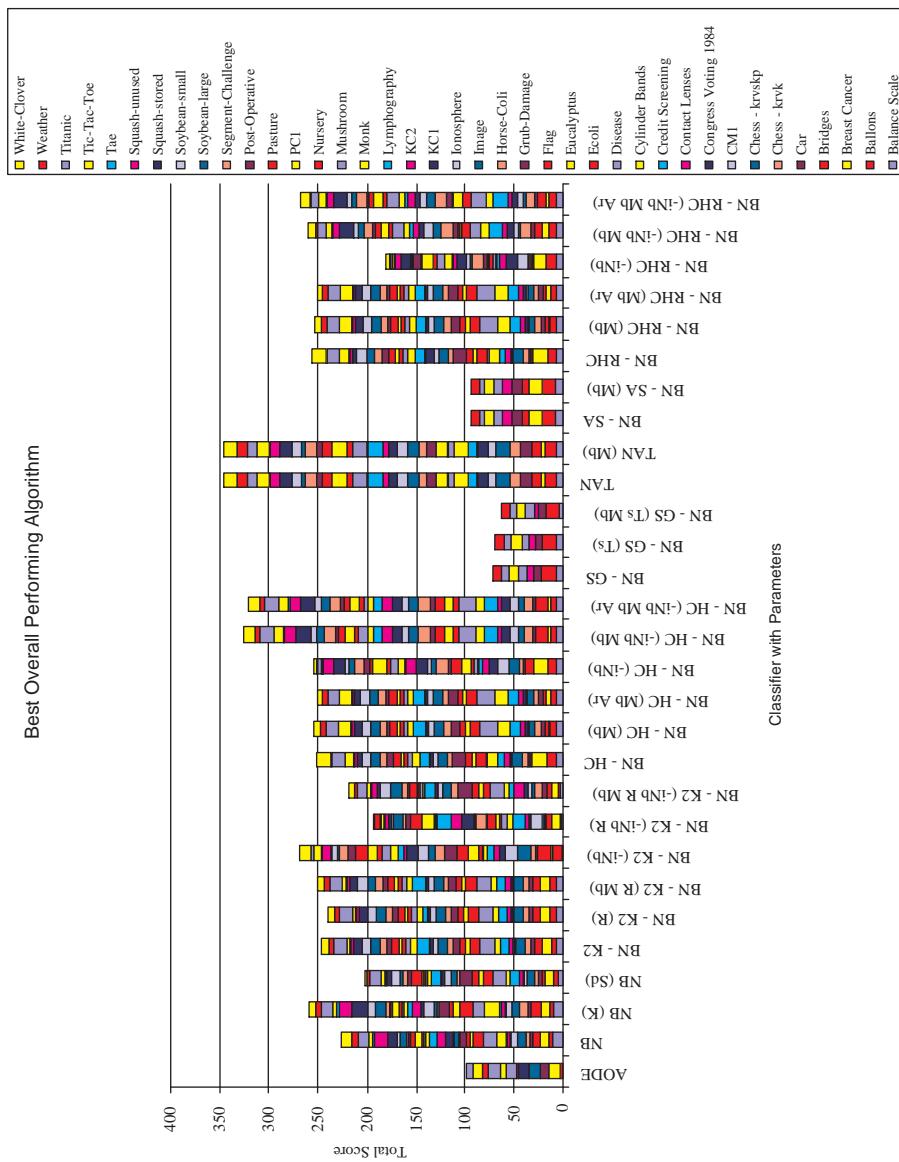


Figure 3: Comparison of overall performance of various classifiers based upon the proposed scoring algorithm.

class, and is possibly why naïve Bayes does not outperform other Bayesian classifiers.

Of the BN-K2 parameters, BN-K2(-iNb R) performed the worst, with a large difference between this and the next best performing. Given that the BN-K2 (-iNb) performed significantly better without R, this provides suitable evidence against the use of R. If -iNb is used with R, then the whole modelling process is random, including the original structure and the structure of the nodes in the network (as opposed to the order found in a dataset).

Naïve Bayes performed relatively well, especially when considering its strong conditional independence property, adding to Webb’s evidence [5] relating to the power behind the naïve Bayes classifier. It should be noted that the naïve Bayes algorithm performed best with the use of Kernel estimator (parameter k). It outperformed all the other Bayesian classifiers on the following datasets: Balloon, Cylinder Band, E.coli and Tae. Table 5 gives a summary of the top five algorithms in order of ranking.

Table 5: List of the best performing classifiers overall.

Rank	Algorithm
1st =	TAN
1st =	TAN (Mb)
2nd	BN-HC (-iNb Mb)
3rd	BN-HC (-iNb Mb Ar)
4th	BN-K2 (-iNb)
5th	BN-RHC (-iNb Mb Ar)

C Evaluation of the Classification Rules Generated

After obtaining the results from the *Classifiers Training* step, a decision tree could be generated by J48 to automatically select the best classifier to be used for a particular dataset. The output of the decision tree could either be one of the 8 classifiers, or a classifier with a specific parameter settings. For the latter case, the decision tree would be able to predict precisely which classifier to use and also with what kinds of parameter settings. However, the total number of classes (the classifier types) to be captured by the decision tree increased dramatically to 29 in total.

The most accurate decision tree is shown in Figure 4. Using this decision tree, the predicated classifiers for the test datasets obtained through stratified sampling is listed in Table 6. It can be observed from Table 6 that our proposed framework is able to select the best performed Bayesian classifier for 5 datasets out of the total 9 datasets. For the ‘KC2’ and ‘Mushroom’ datasets, although the framework failed to select the best classifier, there is no significant difference between the actual classification accuracies and the best classification accuracies with a marginal drop of 0.3% and 0.4% respec-

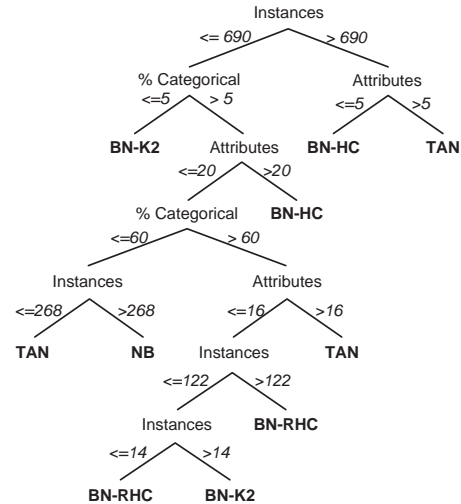


Figure 4: The most accurate binary decision tree learned.

tively. Thus, with 0.5% classification accuracy tolerance, the overall accuracy achieved by the decision tree is 78%.

Table 6: The predicated classifier for the 9 test datasets using the most accurate decision tree.

Dataset	Best Classifier	Best Accuracy	Predicted Classifier	Test Accuracy
Ballons	BN-GS	100	BN-K2	100
Bridges	BN-K2	76.8421	BN-RHC	73.6842
Car	TAN	94.6181	TAN	94.6181
KC2	BN-K2	82.567	BN-HC	82.1839
Mushroom	TAN	100	BN-HC	99.5446
Pasture	BN-K2	83.3333	BN-HC	77.7778
PC1	TAN	92.3354	TAN	92.3354
Soybean-large	BN-K2	92.0266	BN-K2	92.0266
Squash-stored	BN-HC	67.3077	BN-HC	67.3077

D Discussion

The TAN classifier has proven to be the best performing overall, and if a user is unsure as to which algorithm to use on their datasets, then TAN would be the recommended option.

With regards to the various parameter settings used, it was found that when a Markov Blanket correction is made (Mb), the performance of an algorithm in general improves. However, if the correction is made while using a random order of nodes (R), performance drops dramatically. When R is used with BN-K2 on its own, its accuracy also falls.

If the initial network used for learning a structure is initialised as a naïve Bayes network (-iNb) both the accuracies of BN-HC and BN-RHC drop.

The BN-SA classifier will probably have found the optimal Bayesian network structure during the long periods of computing time given to it on many of the datasets, but the ‘temperature’ may have stayed too high forcing

the algorithm to search too many high energy states, or the algorithm may have been stuck in a local minima as a result of a too low temperature.

The BN-GS algorithm may also have found the optimal solution during the first stages of the search, but the algorithm continues to search for more solutions until a near optimal solution is found. If the algorithm could have been halted and the solution pulled out near an optimal state, the algorithm will have most likely performed better than the TAN algorithm. Finally, on datasets with 100% categorical data, the naïve Bayes classifier performs the same regardless of which parameter settings are used.

V CONCLUSIONS

This paper provides a means for judging which classifiers are the best to be used for a given dataset. This therefore contributes a very useful resource to inexperienced or casual data miners. Also, this paper presents to the best of our knowledge a first attempt to produce a set of rules through learning algorithms to identify the best classifiers available. The results show that the J48 algorithm derived a decision tree that could, with 78% accuracy (with 0.5% classification accuracy tolerance), predict the best classifier to use on an unseen dataset. The fact that this degree of accuracy was achieved on a limited number of datasets, and a limited number of classifiers and their parameter settings, shows the potential of the framework in generating more accurate decision trees – which in turn would allow a user to choose the best algorithm for their dataset.

ACKNOWLEDGEMENT

Stuart Moran wishes to thank Daniel Rodriguez and Stephen Moran for valuable discussions and to express his special thanks to Liz Dowthwaite and Peter Moran for their proof reading and support throughout the research.

REFERENCES

- [1] W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus. Knowledge discovery in databases: An overview. *Knowledge Discovery in Databases*, pages 1–27, 1991.
- [2] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.
- [3] J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann, 1988.
- [4] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103 – 130, 1997.
- [5] G.I. Webb, J. Boughton, and Z. Wang. Not so naive bayes: Aggregating one-dependence estimators. *Machine Learning*, 58(1):5–24, January 2005.
- [6] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.
- [7] G.F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, October 1992.
- [8] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671 – 681, May 1983.
- [9] Eibe Frank, Geoff Holmes, Mike Mayo, Bernhard Pfahringer, Tony Smith, and Ian Witten. *Weka3: Data Mining Software in Java*. The University of Waikato, 2006. <http://www.cs.waikato.ac.nz/ml/weka/>.