

# An Empirical Framework for Automatically Selecting the Best Bayesian Classifier

Stuart Moran\*

Yulan He<sup>†</sup>

Kecheng Liu\*

*Abstract*—Data miners have access to a significant number of classifiers and use them on a variety of different types of dataset. This large selection makes it difficult to know which classifier will perform most effectively in any given case. 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 and the sampling techniques applied, a set of rules were learned that select with 78% accuracy (with 0.5% classification accuracy tolerance), the most effective classifier.

*Keywords:* Bayesian networks; Data mining; Classification; Search algorithm; Decision tree.

## 1 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.

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 2 presents the proposed empirical framework for automatically selecting the best Bayesian classifier. Section 3 discusses the performance evaluation metrics. Section 4 presents the experimental results from the 39 datasets selected. Finally, Section 5 concludes the paper.

## 2 Proposed Framework

The empirical framework for automatically selecting the best classifier is depicted in Figure 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

\*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

<sup>†</sup>School of Engineering, Computing and Mathematics, University of Exeter, North Park Road, Exeter EX4 4QF, UK Tel/Fax: 44-1392-263650/264067 Email: y.l.he.01@cantab.net

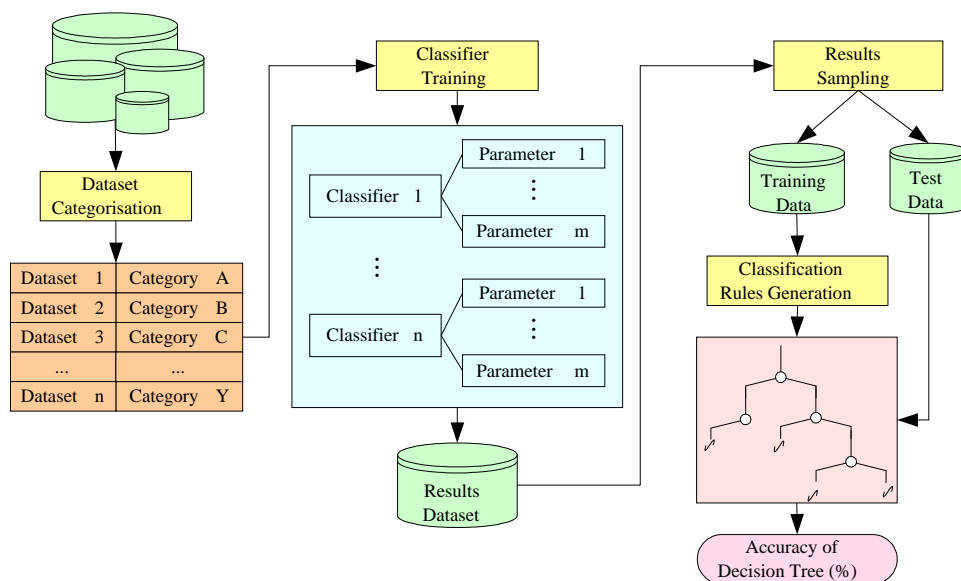


Figure 1: An empirical framework for automatically selecting the best classifier.

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 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.

## 2.1 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 repository<sup>1</sup>, the WEKA Web site<sup>2</sup>, and the Promise repository<sup>3</sup>. 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 initially split into four groups: those datasets with  $> 286$  and those with  $\leq 286$  instances, and within each of these groups those datasets with  $> 16$  attributes and those with  $\leq 16$ . 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%	≥ 50%	< 50%	100%	≥ 50%	< 50%
≤ 286 Instances					
> 16 Attributes			≤ 16 Attributes		
100%	≥ 50%	< 50%	100%	≥ 50%	< 50%

<sup>1</sup><http://archive.ics.uci.edu/ml/>

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

<sup>3</sup><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

## 2.2 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], Averaged One Dependence Estimator (AODE) [5], Tree-Augmented Naïve Bayes (TAN) [6], BN with different structure learning algorithms such as K2 (BN-K2) [7], Genetic Search (BN-GS) [2], Simulated Annealing (BN-SA) [8], Greedy Hill Climber (BN-HC)[2], and Repeated Hill Climber (BN-RHC) [2]. More details about these well-known algorithms can be found in the given references.

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 parameter settings investigated include UseKernelEstimator (k), UseSupervisedDiscretisation (Sd), initAsNaiveBayes (iNb), markovBlanketClassifier (Mb), RandomOrder (R), useArcReversal (Ar), and useTournamentSelection (Ts).

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.

## 2.3 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 subsequence subsection.

## 2.4 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 cre-

ated that assigns the most effective classifier available (of those tested) to a particular dataset (See Figure 3).

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.

### 3 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* 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.  $10\text{cm} \pm 0.5\text{cm}$ . The mean absolute error is then the weighted average of all the absolute errors found from cross validations.
- *Relative absolute error* 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 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.

## 4 Experimental Results

### 4.1 The Best Algorithm for Each Category

The criteria for each category of data has been discussed in Section 2.1. 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 Section 3. 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 Figure 2. The best performing parameter settings for each category are summarised in Table 3.

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.

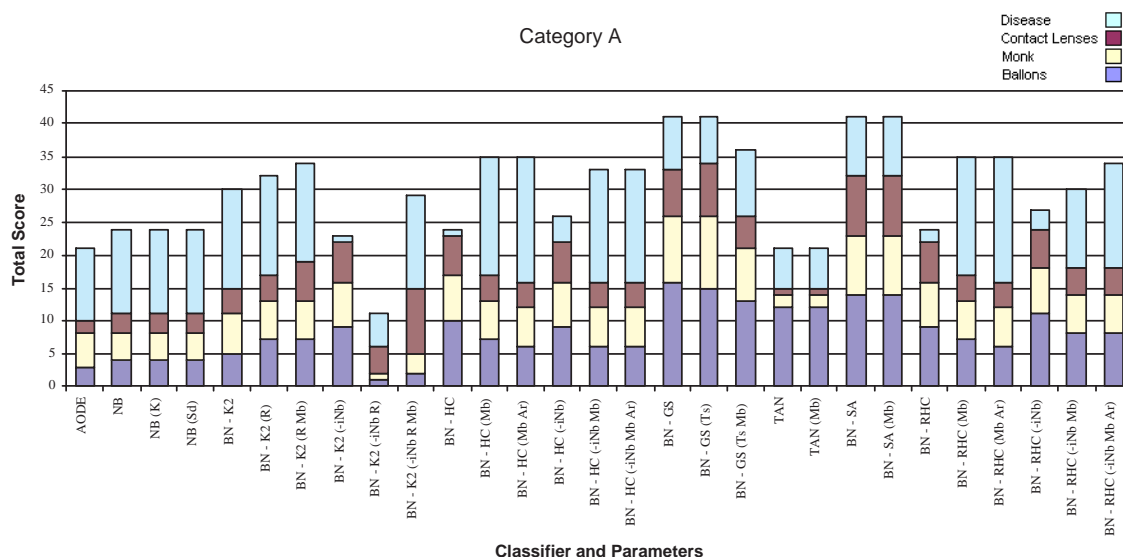


Figure 2: The performance of each classifier with various parameter settings on category A.

Table 3: 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

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.

## 4.2 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.

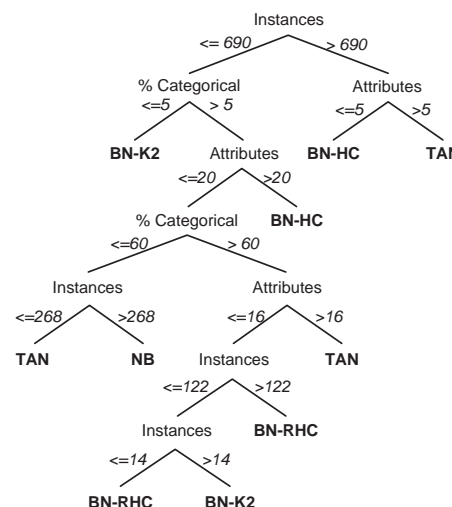


Figure 3: The most accurate binary decision tree learned.

The most accurate decision tree is shown in Figure 3. Using this decision tree, the predicated classifiers for the test datasets obtained through stratified sampling is listed in Table 4. It can be observed from Table 4 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% respectively. Thus, with 0.5% classification accuracy tolerance, the overall accuracy achieved by the decision tree is 78%.

### 4.3 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.

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.

Table 4: 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

## 5 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.

## 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/>.