Appendix A
Experimental Framework Used in This Book

This appendix describes the materials and methods used in this book, as well as the evaluation techniques and performance metrics employed.

A.1 Software Tools

The experiments performed in this book were executed using the software tools Matlab and Weka, which are described in the following paragraphs.

- Matlab [1] is a numerical computing environment, well known and widely used by scientific researchers. It was developed by MathWorks in 1984 and its name comes from *Matrix Laboratory*. Matlab allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.
- Weka (Waikato Environment for Knowledge Analysis) [2] is a collection of machine learning algorithms for data mining tasks. The algorithms can be either applied directly to a dataset or called from your own Java code. Weka contains tools for data preprocessing, classification, regression, clustering, association rules, and visualization. It is also well suited for developing new machine learning schemes.

A.2 Datasets

A common procedure for testing the adequacy of a given feature selection methods is to check its performance over a benchmark dataset. To facilitate this task, several repositories of data exist that will be listed in Section A.2.1. In this book, specifically, we have used synthetic datasets (the relevant features are known a priori), classical datasets extracted from the widely used UCI repository [3] and also some datasets that belong to the category of DNA microarray datasets. After presenting the most popular repositories, subsequent subsections will present the main characteristics of the datasets employed in this book.

A.2.1 Data Repositories

In this subsection we present some public data repositories covering a wide spectrum of data which could be useful for feature selection researchers.

- Miscellaneous repositories:
  - *The UC Irvine Machine Learning Repository* (UCI), from University of California, Irvine :
    `http://archive.ics.uci.edu/ml/`
  - *UCI KDD Archive*, from University of California, Irvine:
    `http://kdd.ics.uci.edu`
  - *LIBSVM Database*:
    `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`
  - *Public Data Sets*, from Amazon Web Services:
    `http://aws.amazon.com/datasets`
  - *The Datahub*:
    `http://datahub.io/dataset`
- Specific repositories for microarray data:
  - *ArrayExpress*, from the European Bioinformatics Institute:
    `http://www.ebi.ac.uk/arrayexpress/`
  - *Cancer Program Data Sets*, from the Broad Institute:
    `http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi`
  - *Dataset Repository*, from the Bioinformatics Research Group of Universidad Pablo de Olavide:
    `http://www.upo.es/eps/bigs/datasets.html`
  - *Feature Selection Datasets*, from Arizona State University:
    `http://featureselection.asu.edu/datasets.php`
  - *Gene Expression Model Selector*, from Vanderbilt University:
    `http://www.gems-system.org`
  - *Gene Expression Omnibus*, from the National Insititutes of Health:
    `http://www.ncbi.nlm.nih.gov/geo/`
  - *Gene Expression Project*, from Princeton University:
    `http://genomics-pubs.princeton.edu/oncology/`
  - *Kent Ridge Bio-Medical Dataset Repository*, from the Agency for Science, Technology and Research:
    `http://datam.i2r.a-star.edu.sg/datasets/krbd`

– *Stanford Microarray Database*, from Stanford University:
  http://smd.stanford.edu/

A.2.2 Synthetic Datasets

Several authors choose to use artificial data since the desired output is known; therefore a feature selection algorithm can be evaluated independently of the classifier used. Although the final goal of a feature selection method is to test its effectiveness over a real dataset, the first step should be on synthetic data. The reason for this is twofold [4]:

1. Controlled experiments can be developed by systematically varying chosen experimental conditions, such as adding more irrelevant features or noise in the input. This fact facilitates drawing more useful conclusions and testing the strengths and weaknesses of the existing algorithms.
2. The main advantage of artificial scenarios is the knowledge of the set of optimal features that must be selected; thus the degree of closeness to any of these solutions can be assessed in a confident way.

The synthetic datasets used in this book try to cover different problems: increasing number of irrelevant features, redundancy, noise in the output, alteration of the inputs, nonlinearity of the data, etc. These factors complicate the task of the feature selection methods, which are very affected by them, as will be shown afterwards. Besides, some of the datasets have a significantly higher number of features than samples, which implies an added difficulty for the correct selection of the relevant features. Table A.1 shows a summary of the main problems covered by them, as well as the number of features and samples and the relevant attributes which should be selected by the feature selection methods. Notice whilst the main characteristic of each dataset is emphasized, it can have other characteristics too.

A.2.2.1 CorrAL

The CorrAL dataset [5] has six binary features (i.e., $f_1, f_2, f_3, f_4, f_5, f_6$), and its class value is $(f_1 \wedge f_2) \vee (f_3 \wedge f_4)$. Feature f_5 is irrelevant and f_6 is correlated to the class label by 75%.

CorrAL-100 [6] was constructed by adding 93 irrelevant binary features to the previous CorrAL dataset. The data for the added features were generated randomly. Both datasets (CorrAL and CorrAL-100) have 32 samples that are formed by considering all possible values of the four relevant features and the correlated one (2^5). The correct behavior for a given feature selection method is to select the four relevant features and to discard the irrelevant and correlated ones. The correlated feature is redundant if the four relevant features are selected, and, besides, it is correlated

Table A.1: Summary of the synthetic datasets used. "Corr." stands for "Correlation"

Dataset	No. of features	No. of samples	Relevant features	Corr.	Noise	Nonlinear	No. feat >> No. samples
Corral	6	32	1–4	✓			
Corral-100	99	32	1–4	✓			✓
Led-25	24	50	1–7		✓		
Led-100	99	50	1–7		✓		✓
Madelon	500	2400	1–5		✓	✓	
Monk1	6	122	1,2,5			✓	
Monk2	6	122	1–6			✓	
Monk3	6	122	2,4,5		✓		
Parity3+3	12	64	1–3			✓	
SD1*	4020	75	G_1, G_2				✓
SD2*	4040	75	$G_1 - -G_4$				✓
SD3*	4060	75	$G_1 - -G_6$				✓
XOR-100	99	50	1,2			✓	✓

* G_i means that the feature selection method must select only one feature within the ith group of features.

to the class label by 75%, so if one applies a classifier after the feature selection process, 25% error will be obtained.

A.2.2.2 XOR-100

XOR-100 [6] has two relevant binary features and 97 irrelevant binary features (randomly generated). The class attribute takes binary values and the dataset consists of 50 samples. Features f_1 and f_2 are correlated with the class value with XOR operation (i.e., class equals $f_1 \oplus f_2$). This is a hard dataset for the sake of feature selection because of the small ratio between number of samples and number of features and due to its nonlinearity (unlike the CorrAL dataset, which is a multi-variate dataset).

A.2.2.3 Parity3+3

The parity problem is a classic problem where the output is $f(x_1,\ldots,x_n) = 1$ if the number of $x_i = 1$ is odd and $f(x_1,\ldots,x_n) = 0$ otherwise. The Parity3+3 dataset is a modified version of the original parity dataset. The target concept is the parity of three bits. It contains 12 features among which three are relevant, another 3 are redundant (repeated) and another six are irrelevant (randomly generated).

A.2.2.4 The Led Problem

The Led problem [7] is a simple classification task that consists of, given the active leds on a seven segment display, identifying the digit that the display is representing. Thus, the classification task to be solved is described by seven binary attributes (see Figure A.1) and ten possible classes available ($C = \{0,1,2,3,4,5,6,7,8,9\}$). A 1 in an attribute indicates that the led is active, and a 0 indicates that it is not active.

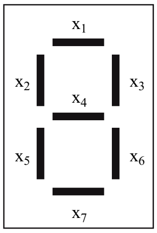

Fig. A.1: Led scheme

Two versions of the Led problem will be used: the first one, Led25, adding 17 irrelevant attributes (with random binary values), and the second one, Led100, adding 92 irrelevant attributes. Both versions contain 50 samples. The small number of samples was chosen because we are interested in dealing with datasets with a large number of features and a small sample size. Also, different levels of noise (altered inputs) have been added to the attributes of these two versions of the Led dataset: 2%, 6%, 10%, 15% and 20%. In this manner, the tolerance to different levels of noise of the feature selection methods tested will be checked. Note that, as the attributes take binary values, adding noise means assigning an incorrect value to the relevant features.

A.2.2.5 The Monk Problems

The MONK problems [8] rely on an artificial robot domain, in which robots are described by six different discrete attributes $(x_1,\ldots,x_6)$. The learning task is a binary classification task. The logical description of the class of the Monk problems is the following:

- Monk1: $(x_1 = x_2) \vee (x_5 = 1)$
- Monk2: $(x_n = 1)$ exactly two $n \in 1,2,3,4,5,6$
- Monk3: $(x_5 = 3 \wedge x_4 = 1) \vee (x_5 \neq 4 \wedge x_2 \neq 3)$

In the case of Monk3, among the 122 samples, 5% are misclassifications, i.e., noise in the target.

A.2.2.6 SD1, SD2 and SD3

These three synthetic datasets (SD1, SD2 and SD3) [9] are challenging problems because of their large number of features (around 4,000) and the small number of samples (75), besides a large number of irrelevant attributes.

SD1, SD2 and SD3 are three-class datasets with 75 samples (each class containing 25 samples) generated based on the approach described by [10]. Each synthetic dataset consists of both relevant and irrelevant features. The relevant features in each dataset are generated from a multivariate normal distribution using mean and covariance matrices [9]. Besides, 4,000 irrelevant features are added to each dataset, where 2,000 are drawn from a normal distribution of N(0,1) and the other 2,000 are sampled with a uniform distribution $U[-1,1]$. It is necessary to introduce some new definitions of multiclass relevancy features: full class relevant (FCR) and partial class relevant (PCR) features. Specifically, FCR denotes genes (features) that serve as candidate biomarkers for discriminating between all cancer types. However, PCR are genes (features) that distinguish subsets of cancer types.

SD1 is designed to contain only 20 FCR and 4,000 irrelevant features. Two groups of relevant genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in the same group are redundant with each other and the optimal gene subset for distinguishing the three classes consists of any two relevant genes from different groups.

SD2 is designed to contain 10 FCR, 30 PCR, and 4,000 irrelevant features. Four groups of relevant, i.e., FCR and PCR, genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in each group are redundant to each other, and in this dataset, only genes in the first group are FCR genes while genes in the three last groups are PCR genes. The optimal gene subset to distinguish all the three classes consists of four genes, one FCR gene from the first group and three PCR genes each from one of the three remaining groups.

SD3 has been designed to contain only 60 PCR and 4,000 irrelevant features. Six groups of relevant genes are generated from a multivariate normal distribution, with 10 genes in each group. Genes in the same group are designed to be redundant to each other and the optimal gene subset to distinguish all the three classes thus consists of six genes with one from each group.

It has to be noted that the easiest dataset in order to detect relevant features is SD1, since it contains only FCR features and the hardest one is SD3, due to the fact that it contains only PCR genes, which are more difficult to detect.

A.2.2.7 Madelon

The Madelon dataset [11] is a two-class problem originally proposed in the NIPS 2003 feature selection challenge. The relevant features are situated on the vertices of a five-dimensional hypercube. Five redundant features were added, obtained by multiplying the useful features by a random matrix. Some of the previously defined features were repeated to create 10 more features. The other 480 features are drawn from a Gaussian distribution and labeled randomly. This dataset presents high dimensionality both in number of features and in number of samples and the data were distorted by adding noise, flipping labels, shifting and rescaling. For all these reasons, it conforms to a hard dataset for the sake of feature selection.

A.2.3 DNA Microarray Datasets

DNA microarray data classification is a serious challenge for machine learning researchers due to its high dimensionality and small sample size. Typical values are around 10,000 gene expressions and 100 or less tissue samples. For this reason, these types of datasets are usually employed to test the efficiency of a feature selection method. Some datasets were originally divided to training and test sets whilst others only have a training set. On the one hand, Table A.2 shows the main characteristics of the datasets employed in this book having a unique training set (for example, for applying five-fold cross validation). On the other hand, Table A.3 presents the characteristics of the datasets used with separated training and test sets. All the datasets described in this section are available for download in [12] and [13].

Table A.2: Dataset description for binary microarray datasets

Dataset	Attributes	Samples	Distribution
Brain	12625	21	33–67%
CNS	7129	60	35–65%
Colon	2000	62	35–65%
DLBCL	4026	47	49–51%
GLI	22283	85	31–69%
Ovarian	15154	253	36–64%
SMK	19993	187	48–52%

Table A.3: Dataset description for binary microarray datasets with train and test sets

Dataset	Attributes	Samples Train	Samples Test	Train distribution	Test distribution
Breast	24481	78	19	44–56%	37–63%
Prostate	12 600	102	34	49–51%	26–74%

A.3 Validation Techniques

To evaluate the goodness of the selected set of features, it is necessary to have an independent test set with data which have not been seen by the feature selection method. In some cases, the data come originally distributed into training and test sets, so the training set is usually employed to perform the feature selection process and the test set is used to evaluate the appropriateness of the selection. However, not all the datasets come originally partitioned. To overcome this issue, several validation techniques exist, and in the following we describe those used in this book.

A.3.1 k-Fold Cross-validation

This is one of the most famous validation techniques [14]. The data (D) is partitioned into k nonoverlapping subsets $D_1, \ldots, D_k$ of roughly equal size. The learner is trained on $k-1$ of these subsets combined together and then applied to the remaining subset to obtain an estimate of the prediction error. This process is repeated in turn for each of the k subsets, and the cross-validation error is given by the average of the k estimates of the prediction error thus obtained. In the case of feature selection, note that with this method there will be k subsets of selected features. A common practice is to merge the k different subsets (either by union or by intersection) or to keep the subset obtained in the fold with the best classification result.

A.3.2 Leave-One-Out Cross-validation

This is a variant of k-fold cross validation where k is the number of samples [14]. A single observation is left out each time.

A.3.3 Bootstrap

This is a general resampling strategy [15]. A *bootstrap sample* consists of n being the number of samples equally likely to be drawn, with replacement, from the original data. Therefore, some of the samples will appear multiple times, whereas others will not appear at all. The learner is designed on the bootstrap sample and tested on the left-out data points. The error is approximated by a sample mean based on independent replicates (usually between 25 and 200). Some famous variants of this method exist, such as *balanced bootstrap* or *0.632 bootstrap* [16]. As in the previous methods, there will be as many subsets of features as repetitions of the method.

A.3.4 Holdout Validation

This technique consists of randomly splitting the available data into a disjoint pair training test [14]. A common partition is to use 2/3 for training and 1/3 for testing. The learner is designed based on the training data and the estimated error rate is the proportion of errors observed in the test data. This approach is usually employed when some of the datasets in a study come originally divided into training and test sets whilst others do not. In contrast to other validation techniques, a unique set of selected features is obtained.

A.4 Statistical Tests

When performing several executions of a method, different results are obtained (e.g., after applying a k-fold cross validation). In this situation, statistical tests may be performed to check if there are significant differences among the medians for each method. In this book, the most used statistical methods were Kruskal-Wallis [17] and a multiple comparison procedure (Tukey's) [18]. The experimental procedure is detailed in the following:

1. A Kruskal-Wallis test is applied to check if there are significant differences among the medians for each model. In this book, we have opted for a level of significance $\alpha = 0.05$.
2. If the nonparametric Kruskal-Wallis test is significant, it means that at least a model exists that is better than the others. In this case, it is necessary to perform a multiple comparison procedure to figure out which model is the best.
3. Finally, the set of models with performance that is not significantly worse than the best is obtained. Among the models in this set, the simplest one should be selected.

A.5 Discretization Algorithms

Many filter algorithms are shown to work on discrete data [19]. In order to deal with numeric attributes, a common practice for those algorithms is to discretize the data before conducting feature selection. For both users and experts, discrete features are easier to understand, use, and explain and discretization can make learning more accurate and faster [20]. In general, the results obtained (decision trees, induction rules) by using discrete features are usually more compact, shorter and more accurate than those by using continuous ones; hence the results can be more closely examined, compared, used and reused. In addition to the many advantages of using discrete data over continuous data, a suite of classification learning algorithms can only deal with the former.

In essence, the process of discretization [21] involves the grouping of continuous values into a number of discrete intervals. However, the decisions of which continuous values to group together, how many intervals to generate, and thus where to position the interval cut points on the continuous scale of attribute values are not always identical for the different discretization methods. The discretizers used in this book are subsequently described. Previously, some notation considerations are given: given a numeric attribute X_i and n training instances for which the values of X_i are known, the minimum and the maximum values are v_{min} and v_{max} respectively. All the discretization methods first sort the values into ascending order.

A.5.0.1 Entropy Minimization Discretization, EMD

This popular method was created by Fayyad and Irani [22]. EMD evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For evaluating each candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidates. The binary discretization is applied recursively, always selecting the best cut point. A minimum description length criterion (MDL) is applied to decide when to stop discretization.

A.5.0.2 Proportional *k*-Interval Discretization, PKID

PKID is a method created by Yang et al. [23]. The idea behind PKID is that discretization bias and variance relate to interval size and interval number. This strategy seeks an appropriate trade-off between the bias and variance of the probability estimation by adjusting the number and size of intervals to the number of training instances. The following compromise is adopted: given a numeric attribute, supposing we have n training instances with known values for the attribute, we discretize it into $\sqrt{n}$ intervals, with $\sqrt{n}$ instances in each interval. Thus, we give equal weight to both bias and variance management. Further, with n increasing, both the number and size of intervals increase correspondingly, which means discretization can decrease both the bias and variance of the probability estimation. This is very desirable, because if a numeric attribute has more instances available, there is more information about it. PKID has greater capacity to take advantage of the additional information inherent in large volumes of training data.

A.6 Classification Algorithms

If we are dealing with real datasets, the relevant features are not known a priori. Therefore, it is necessary to use a classification algorithm to evaluate the performance of the feature selection, focusing on the classification accuracy. Unfortunately, the class prediction depends also on the classification algorithm used, so when testing a feature selection method, a common practice is to use several classifiers to obtain results as classifier-independent as possible. This section describes the most common classification algorithms which are used throughout this book. Notice that some of them only can work with categorical features, whereas others require numerical attributes. In the first case, the problem is often solved by discretizing the numerical features. In the second case, it is common to use a conversion method which assigns numerical values to the categorical features.

A.6.1 Support Vector Machine, SVM

A Support Vector Machine [24] is a learning algorithm typically used for classification problems (text categorization, handwritten character recognition, image classification, etc.). More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training data point of any class (so-called functional margin), since, in general, the larger the margin, the lower the generalization error of the classifier. In its basic implementation, it can only work with numerical data and binary classes.

A.6.2 Proximal Support Vector Machine, PSVM

This method classifies points, assigning them to the closest of two parallel planes (in input or feature space) that are pushed as far apart as possible [25]. The difference with a Support Vector Machine (SVM) is that PSVM classifies points by assigning them to one of two disjoint half-spaces. The PSVM leads to an extremely fast and simple algorithm by generating a linear or nonlinear classifier that merely requires the solution of a single system of linear equations.

A.6.3 C4.5

C4.5 is a classifier developed by [26], as an extension of the ID3 algorithm (Iterative Dichotomiser 3). Both algorithms are based on decision trees. A decision tree classifies a pattern building a descending filtering of itself until finding a leaf, which points to the corresponding classification. One of the improvements of C4.5 with respect to ID3 is that C4.5 can deal with both numerical and symbolic data. In order to handle continuous attributes, C4.5 creates a threshold and, depending on the value that takes the attribute, the set of instances is divided.

A.6.4 Naive Bayes, NB

A naive Bayes classifier [27] is a simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions. This classifier assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. A naive Bayes classifier considers each of the features to contribute independently to the probability that a sample belongs to a given class, regardless of the presence or absence of the other features. Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In fact, naive Bayes classifiers are simple, efficient and robust to noise and irrelevant attributes. However, they can only deal with symbolic data, although discretization techniques can be used to preprocess the data.

A.6.5 k-Nearest Neighbors, k-NN

k-nearest neighbor [28] is a classification strategy that is an example of a "lazy learner." An object is classified by a majority vote from its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors (where k is some user-specified constant). If $k = 1$ (as is the case in the experiments in this book), then the object is simply assigned to the class of that single nearest neighbor. This method is more adequate for numerical data, although it can also deal with discrete values.

A.6.6 One-Layer Feedfoward Neural Network, One-Layer NN

This algorithm consists of training a single-layer feedfoward neural network using a new supervised learning method proposed by [29]. The method is based on the use of an alternative cost function that measures the errors before the nonlinear activation functions instead of after them, as is normally the case. As an important consequence, the solution can be obtained easily and rapidly because the new cost function is convex. Therefore, the absence of local minima is assured in this situation.

A.7 Evaluation Measures

In order to evaluate the behavior of the feature selection methods after applying a classifier, several evaluation measures need to be defined.

- *True positive (TP)*: percentage of positive examples correctly classified as so.
- *False positive (FP)*: percentage of negative examples incorrectly classified as positive.
- *True negative (TN)*: percentage of negative examples correctly classified as so.
- *False negative (FN)*: percentage of positive examples incorrectly classified as negative.
- $Sensitivity = \frac{TP}{TP+FN}$
- $Specificity = \frac{TN}{TN+FP}$
- $Accuracy = \frac{TN+TP}{TN+TP+FN+FP}$
- $Error = \frac{FN+FP}{TN+TP+FN+FP}$

A.7.1 Multiple-Criteria Decision-Making

Multiple-criteria decision-making (MCDM) [30] is focused on evaluating classifiers from different aspects and producing rankings of them. A multi-criteria problem is formulated using a set of alternatives and criteria. Among many MCDM methods that have been developed up to now, *technique for order of preference by similarity to ideal solution* (TOPSIS) [31] is a well-known method that will be used. TOPSIS finds the best algorithms by minimizing the distance to the ideal solution whilst maximising the distance to the anti-ideal one. The extension of TOPSIS proposed by [32] is used in this research.

References

1. MATLAB. *version 8.1.0.604 (R2013a)*. The MathWorks Inc., 2013.
2. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. The Weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.
3. Bache, K. and Lichman, M. UCI Machine Learning Repository, 2013. University of California, Irvine, School of Information and Computer Sciences. http://archive.ics.uci.edu/ml .
4. Belanche, L. A. and González, F. F. Review and evaluation of feature selection algorithms in synthetic problems. *arXiv preprint arXiv:1101.2320*, 2011.
5. John, G. H., Kohavi, R. and Pfleger, K. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994.
6. Kim, G., Kim, Y., Lim, H. and Kim, H. An MLP-based feature subset selection for HIV-1 protease cleavage site analysis. *Artificial Intelligence in Medicine*, 48(2):83–89, 2010.
7. Breiman, L. *Classification and Regression Trees*. CRC Press, 1993.
8. Thrun, S. B., Bala, J. W., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K. A., Dzeroski, S., Fisher, D. H., Fahlman, S. E. and others. The monk's problems a performance comparison of different learning algorithms. Technical Report CMU-CS-91-197, Carnegie Mellon University, 1991.
9. Zhu, Z., Ong, Y. S. and Zurada, J. M. Identification of full and partial class relevant genes. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(2):263–277, 2010.
10. Díaz-Uriarte, R. and Alvarez De Andres, S. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006.
11. Guyon, I., Gunn, S., Nikravesh, M. and Zadeh, L. *Feature Extraction: Foundations and Applications*. Springer, 2006.
12. Broad Institute. Cancer Program Data Sets. http://www.broadinstitute.org/cgi-bin/cancer/datasets.cgi
13. Kent Ridge Bio-Medical Dataset. http://datam.i2r.a-star.edu.sg/datasets/krbd
14. Bramer, M. *Principles of Data Mining*. Springer, 2007.
15. Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, pages 1–26, 1979.
16. Efron, B. and Tibshirani, R. *An Introduction to the Bootstrap*, volume 57. Chapman & Hall/CRC, 1993.
17. Wolfe, D. A. and Hollander, M. *Nonparametric Statistical Methods*, 1973.
18. Hsu, J. C. *Multiple Comparisons: Theory and Methods*. CRC Press, 1996.
19. Liu, H. and Setiono, R. Feature Selection via Discretization. *Journal of IEEE Transactions on Knowledge and Data Engineering*, 9(4):642-645, 1997.
20. Liu, H., Hussain, F., Tan, C. L. and Dash, M. Discretization: An Enabling Technique. *Data Mining and Knowledge Discovery Journal*, 6(4):393-423, 2002.
21. Janssens, D., Brijs, T., Vanhoof, K. and Wets, G. Evaluating the performance of cost-based discretization versus entropy and error-based discretization. *Computers and Operations Research Journal*, 33(11):3107-3123, 2006.
22. Fayyad, U. M. and Irani, K. B. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *13th International Joint Conference on Artificial Intelligence*, pages 1022-1029, 1993.
23. Yang, Y. and Webb, G. I. Proportional *k*-Interval Discretization for Naive-Bayes Classifiers. In *12th European Conference on Machine Learning*, pages 564–575, 2001.
24. Vapnik, V. N. *Statistical Learning Theory*. Wiley, 1998.
25. Fung, G. and Mangasarian, O. L. Proximal support vector machine classifiers. In *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 77–86. ACM, 2001.
26. Quinlan, J. R. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

27. Rish, I. An empirical study of the naive Bayes classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, volume 3, pages 41–46, 2001.
28. Aha, D. W., Kibler, D. and Albert, M. K. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.
29. Castillo, E., Fontenla-Romero, O., Guijarro-Berdiñas, B. and Alonso-Betanzos, A. A global optimum approach for one-layer neural networks. *Neural Computation*, 14(6):1429–1449, 2002.
30. Zeleny, M. and Cochrane, J. L. *Multiple Criteria Decision Making*. McGraw-Hill, 1982.
31. Hwang, C. L. and Yoon, K. *Multiple Attribute Decision Making*. Springer, 1981.
32. Olson, D. L. Comparison of weights in TOPSIS models. *Mathematical and Computer Modelling*, 40(7):721–727, 2004.