Learning with few examples: an empirical study on leading classifiers

Christophe Salperwyck and Vincent Lemaire

Abstract— **Learning algorithms proved their ability to deal with large amount of data. Most of the statistical approaches use defined size learning sets and produce static models. However in specific situations: active or incremental learning, the learning task starts with only very few data. In that case, looking for algorithms able to produce models with only few examples becomes necessary. The literature's classifiers are generally evaluated with criterion such as: accuracy, ability to order data (ranking)... But this classifiers' taxonomy can dramatically change if the focus is on the ability to learn with just few examples. To our knowledge, just few studies were performed on this problem. The study presented in this paper aims to study a larger panel of both algorithms (9 different kinds) and data sets (17 UCI bases).**

I. Introduction

Learning machines have shown their ability to deal with huge volumetry on real problems [1], [2]. Nevertheless most of the works were realized for data analysis on homogeneous and stationary data. Usually learning machines use data set with fixed sizes and produce static models. However in certain situations, the learning task starts with only few data. In such cases finding algorithms able to produce accurate models with few data and low variance is an advantage. Active and incremental learning are the two main learning problems where a learning machine able to learn with few data is necessary. This study only focuses on supervised learning.

Active learning [3] is used when lots of data are available but labeling them is expensive (indeed labels are bought). In that case the goal is to select the smallest amount of data which will provide the best model. These data are expected to be very expressive and an algorithm able to deliver an accurate model with just few data is needed in order to avoid buying more data.

Incremental learning [4] start to learn with few examples as it theoretically has to learn from the first provided examples. The model is then improved as new examples are arriving. The model quality at the beginning depends on the algorithm capacity to learn fast with few examples. Incremental learning research started a long time ago but it recently reappears with data stream mining. Indeed numerous software are generating data streams: sensor networks, web pages access logs... These data arrive fast and are only visible once. Therefore it is mandatory to learn them as soon as they are arriving (on-line learning). Incremental learning appears to be a natural solution to solve streams problems. An example is Hoeffding trees [5] which are widely used in incremental learning on data streams. The tree construction is incremental and nodes are transformed into leaves as examples are arriving. Having a classifier in the tree leaves [6] before they will be transformed, appears to improve the tree accuracy. A classifier that can learn with few data will provide a pertinent local model in the leaves.

The most used classifiers such as decision tree, neural network, support vector machine... are often evaluated with criterion such as accuracy, ability to rank data... But this classifiers taxonomy can be completely different if the focus is on their ability to learn with just few examples.

To our knowledge, the state of art presents only few studies on the learning performance versus the size of the learning data set: in [7] the performance on small and not balanced text datasets is studied using 3 different classifiers (Support Vector Machine (SVM), naive Bayes and logistic regression). In [8], the authors focus on the learning time contrary to this study which focus on the performance versus the size of the training set. In [9] and in [10] the focus is respectively on Parzen Windows and k nearest neighbor. In [11] the construction of linear classifiers is considered for very small sample sizes using a stability measure. In [12] the influence of the training set size is evaluated on 4 computer-aided diagnosis problems using 3 different classifiers (SVM, C4.5 decision tree, k-nearest neighbor). In [13] the authors look at how increasing data set size affects bias and variance error decompositions for classification algorithms. The conclusions of these papers will be compared to the results of this empirical study at the end of this paper.

The present work aims to study a larger panel both of learning algorithms (9 different kinds) and data sets (17 from UCI). In a first part (section II) we will present the classifiers that will be used in this study and their parameters. The experimental protocol will be presented section III: data sets, split between training and test sets, evaluation criterion. Section IV will present the results and analyze them depending on the typology of the classifiers. In the last part we will conclude and propose future works related to this study.

II. Learning speed and classifiers typology

A. Learning speed

Firstly this study does not focus on the bounds or convergence time for a classifier given a training set of n examples. This would correspond to determine the CPU time needed for this classifier to learn n examples. In this study the "learning speed" means the (minimum) number or training examples

Authors are in the group 'Profiling and Datamining', Orange Labs, 2 avenue Pierre Marzin, 22300 Lannion, France (phone: +33 296 053 107; email: firstname.name@orange-ftgroup.com).

a given classifier needs to obtain an "interesting" solution to a problem. Figure 1 shows an example of two classifiers on a same problem with two different learning speeds. X-axis shows the number of examples (n) used to train the classifier and Y-axis the AUC obtained (AUC=$f(n)$).

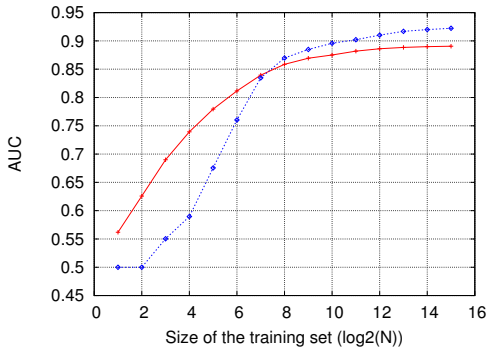


Fig. 1. Illustration of the learning speed of two classifiers (AUC in testing)

This figure illustrates two different behaviors: the "red" (+) algorithm reaches a better performance in a first stage but then when n become larger the "blue" (∘) algorithm is finally better. In order to always have the best model available, we obviously have to use both models: at the beginning the red (+) model and then the blue (∘) one.

This perfectly illustrates the purpose of this study: are there better types of learning algorithms to learn on small training data sets?

B. Typology of benchmarked algorithms

Different families of classification models based on learning parameters exist. They could be: (i) linear classifier for which a hyperplane is learned (this kind of classifiers is based on a linear combination of features such as $y = \sum_j w_j X_j$ where X_j represents a feature and w_j its weight); (ii) non linear classifiers such as multi layer perceptrons, k nearest neighbors or decision trees.

Knowing that: (i) these families can partially overlap each other; (ii) their placement is not always easy (a decision tree can be seen as many hyperplanes - one for each leaf); (iii) a linear classifier can deal with non linear classification problem if a projection is done previously (the "kernel trick" in SVM); (iv) sub families also exist. For example linear classifiers can be dispatched into two big sub families to estimate vectors parameters w:

- generative model is able to randomly generate instances, based on an estimate of the joint probability distribution over explicative variables and the target values. It uses an estimate of the conditional probability $P(X|C)$ and the class probability $P(C)$. The linear discriminant analysis (LDA - [14]) and the naive Bayes classifier [15] are examples of generative models.
- discriminative model is only interested in predicting target values, and do not necessarily require statistical modeling of the joint distribution. Linear regression, logistic regression, perceptron and support vector machines are discriminative models.

Other classifiers can also be found: (i) probabilistic classifiers which estimate conditional probabilities of classes given an input vector ($P(C|X)$) constituted by combinations of explanatory variables (this is not the case with many classifiers as scores returned by support vector machines (SVM)); (ii) parametric classifiers which assume that explanatory variables follow a probability law defined beforehand (a priori). One example is the naive Bayes classifier when presupposing that data follow Gaussian distributions [16].

This study can not cover all the classifier families but tried to explore the space at best. Table I summarizes the tested classifiers in relation to their families. These classifiers and their parameters are presented in the following sub section.

	Linear classifier	Non linear classifier
Generative model	Naive Bayes, Selective naive Bayes	Bayesian network, nearest neighbor
Discriminant model	Logistic regression, SVM	Decision tree, Forest of decision trees

TABLE I
TYPOLOGY VERSUS BENCHMARKED CLASSIFIERS.

C. Benchmarked classifiers

In order to evaluate the classifiers presented in the previous section (see Table I), two public software[1] were used: WEKA [17] (version 3.7.1) and Khiops [18] (version 7.0). All algorithms were used without any prior knowledge. For both software default values were mostly used (if not, parameters which were tuned are described below) and results are presented in section IV.

- Weka - Bayes
  - Unsupervised [19]: standard naive Bayes. Numeric estimator precision values are chosen based on analysis of the training data.
  - Supervised: same as before but a supervised discretization (MDL) is used to convert numeric attributes to nominal ones.
  - BayesNet [20]: Bayes network learning using various search algorithms and quality measures. Used with default parameters: SimpleEstimator and K2 search algorithm.
- Weka - Nearest-neighbor [21]: uses normalized Euclidean distance to find the training instance closest to the given test instance, and predicts the same class as this training instance.
- Weka - Regression:
  - Logistic regression: multinomial logistic regression model with a ridge estimator which is based on [22].

[1] All experiments in this benchmark are reproducible.

  - SVM: Support Vector Machine implementation by [23]. The Weka wrapper is made by [24]. Parameters used were: C-SVC for SVM type, 100MB for cache size and "normalize" set to true. Two kernel types were tried: linear and radial basis function (RBF). A cross-validation was used to optimize RBF parameters for all training set sizes.
- Weka - Trees:
  - ADTree [25]: decision tree with many subtrees and combined with boosting.
  - J48 [26]: C4.5 decision tree proposed by Quinlan.
  - SimpleCart [27]: binary tree based on the Gini coefficient.
  - Random Forest [28]: forest of random trees. Default number of trees in the forest is 10. 40 was also tried.
- Weka - Vote: VFI [29]: classification by voting feature intervals. Intervals are constructed around each class for each attribute (basically discretization). Class counts are recorded for each interval on each attribute and the classification is by voting. The default parameters are using the "weight feature intervals by confidence". It was also tried without it.
- Khiops[2]: a tool developed by Orange Labs. It implements, for supervised classification, a Naive Bayes and a Selected Naive Bayes.
  Naive Bayes or Selective Naive Bayes [30] were tested with different pretreatments on numerical and nominal attributes. On nominal attributes two pretreatments were tested:
  - Basic Grouping: a group per observed value
  - MODL discretization [31]

  On numerical attributes three pretreatments were tested:
  - Equal Frequency
  - Equal Width
  - MODL grouping method [32]

  The two unsupervised Equal Frequency and Equal Width methods are used with a fixed number of bins set to 10. If the number of observed values is below 10 the number of bins is reduced to the number of observed data.

III. Experimental protocol

The experimental protocol aims to show the impact of the training size on the learner performances. Firstly, data sets are presented: they have different characteristics and are recognized by the data mining community. Then the training/testing split of the data sets is explained. Finally a criterion which point up algorithms that learn well with few data is presented.

A. Data set

Different data sets from the data mining community were used in this study. We chose a panel of data sets from the UCI repository [33]. Most of these data sets were already used for benchmarking classifiers in the literature. Data sets with only nominal features, or only numerical features, or a mix of both were used. Sizes are from one hundred to many thousands examples. Table II presents the data sets characteristics in terms of number of examples, number of numerical and nominal features, accuracy of the majority vote classifier.

This study only focuses on binary classification problems. For problems with more than two classes the experimental protocol should be different.

	Data set	#Var	#Num. feat.	#Nom. feat.	#Exa. (n)	Maj. acc.
1	Adult	15	7	8	48842	0.7607
2	Australian	14	6	8	690	0.5550
3	Breast	10	10	0	699	0.6552
4	Bupa	6	6	0	345	0.5797
5	Crx	15	6	9	690	0.5550
6	German	24	24	0	1000	0.7
7	Heart	13	10	3	270	0.5555
8	Hepatitis	19	6	13	155	0.7935
9	Horsecolic	27	7	20	368	0.6304
10	Hypothyroid	25	7	18	3163	0.9522
11	Ionosphere	34	34	0	351	0.6410
12	Mushroom	22	0	22	8416	0.5332
13	Pima	8	8	0	768	0.6510
14	SickEuthyroid	25	7	18	3163	0.9073
15	Sonar	60	60	0	208	0.5336
16	Spam	57	57	0	4307	0.6473
17	Tictactoe	9	0	9	958	0.6534

TABLE II
DATASET CHARACTERISTICS

B. Split of the data sets into learning and testing sets

In order to generate small data sets and keep the widely used 10 cross validations, first a 90%/10% split was used. Then small training data sets were randomly drawn from the 90% training set. The training data sets sizes are taken from the following sizes: $S = \{S_1, S_2, ..., S_{max}\} = \left\{2, 2^2, 2^3, , \ldots, 2^{(\lfloor log_2(0.9n-1) \rfloor)}\right\}$; where n is the original size of the data set[3]. This draw was performed 10 times for all data sets in order to obtain a mean and variance on the result. The algorithm performance is evaluated on the remaining 10% of the cross validation. Figure 2 illustrates this protocol.

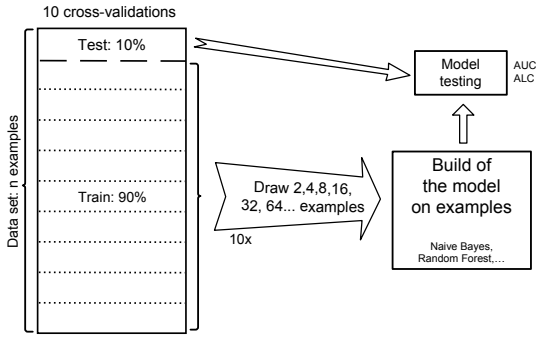


Fig. 2. Data sets construction

[2] www.khiops.com

[3] 0.9n come from the 10 cross validation.

C. Evaluating criterion: ALC

Those experiments give for each dataset an AUC curve [34] on the test dataset versus the number of examples used to train the learning machine. Each curve is constituted of $|S|$ points for all the learning machines as shown in figure 3.

Performances in prediction were evaluated with Area under the Learning Curve (the ALC criterion is proposed in [35]). The AUC (Area Under the ROC Curve) is calculated for every point $(2, 4, 8...)$ defined in the data set. The obtained score corresponds to the normalized ALC calculated as follow:

$$score = \frac{(ALC - Arand)}{(Amax - Arand)}$$

where $Amax$ is the area under the best achievable learning curve (*i.e.* 1) and $Arand$ is the area under the learning curve obtained by random predictions (*i.e.* 0.5).

This criterion has good properties considering the goal of this benchmark. Indeed this criterion emphasizes the AUC on smaller subsets, what is especially the focus of this study. The x-axis is logarithmic which means that the AUC for 2 examples will contribute to the ALC as much as the AUC for 4096 examples.

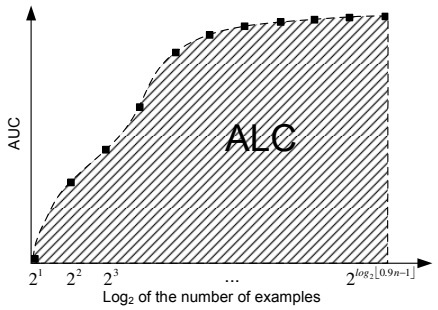


Fig. 3. ALC computation: area under AUC curve

IV. Results

This section presents the study results. The notations used are first described, then tables and curves are presented, finally they are analyzed and an interpretation is proposed.

A. Notations

In order to show compact results, we had to use abbreviation for algorithm names. The first part of the name represents the software used: "W" for algorithms in Weka and no prefix for those which came from Khiops. The name of the tested algorithm constitutes the second part of the abbreviation. The following list gives the meaning of all abbreviations:

- Weka
  - W-ADT: ADTree
  - W-BN: BayesNet - Bayesian network
  - W-IB1: Nearest-neighbor
  - W-RegLog: logistic regression
  - W-NB-NS: non supervised naive Bayes
  - W-NB-S: supervised naive Bayes
  - W-RF10/40: Random Forest with 10/40 trees
  - W-VFI / W-VFIn: VFI with/without option "weight feature intervals by confidence"
  - W-SCart: SimpleCart
  - W-SVM-Lin/W-SVM-RBF: SVM with a linear/radial basis function kernel type
- Khiops. The format is Algorithm – Cont_Var – Nom_Var
  - Algorithm - NB: Naive Bayes or SNB: Selective Naive Bayes
  - Cont_Var for continuous variables - EF: EqualFrequency / EW: EqualWidth / M: MODL
  - Nom_Var for nominal variables - BG: Basic Grouping / M: MODL

B. Tables and curves

1) Tables: The ALC performances obtained by the different algorithms on all data sets are presented in table III. The last line gives the averaged ALC on all data sets for a given algorithm. In order to have a more synthetic view, table V shows the averaged rank, averaged ALC and averaged final AUC (AUC on the largest training set: $2^{(\lfloor log_2(0.9n-1) \rfloor)}$).

As this study focuses on learning from small data sets, the beginning of the AUC curve is particularly interesting. To highlight it, ALC is calculated just between 2 and 2^6 examples so that the ending of the AUC curve is ignored. These results are presented in table IV which is similar to the previous one (table III). The computation was done in the same way for the ranks. The results are presented in table VI.

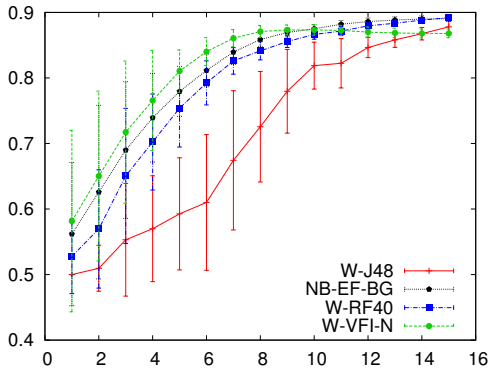
The results variances are shown using a box plot per algorithm on figure 5.

2) Curves: Four curves were chosen to show the performances of different algorithms. The idea is to compare the behavior of a particular algorithm with the best algorithms in this study. The comparison is done on data sets adapted to this particular algorithm in order to observe how the best algorithms perform in that case. The best example is the figure 4(d) which shows a data set working very well for the logistic regression but not that well for the naive Bayes.

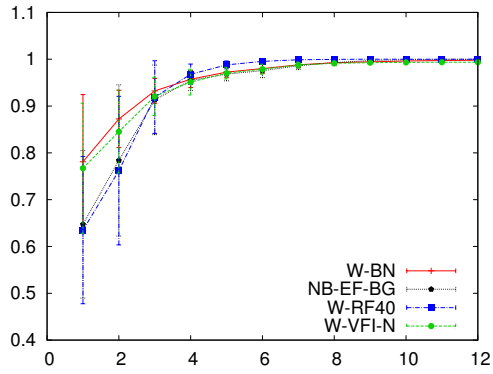
C. Analysis

1) Global analysis: This study was done without a fine tuning of the algorithms' parameters but mainly using their default values. C4.5 (J48), CART (SimpleCart) and SVM are references in data mining but this study shows that they don't perform well on small datasets. Indeed they have the worst averaged ranks and worst averaged ALCs.

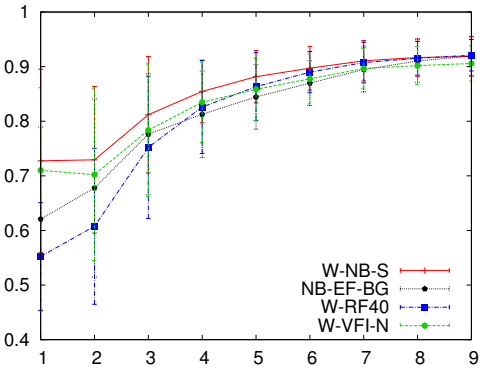
On the contrary, naive Bayes algorithms are known to perform well on small datasets [36]. Figures 4(a) and 4(b) confirm it. Very rapidly the generated classifiers perform well. After seeing around $2^5(32)$ to $2^8(256)$ examples they are almost at their maximum accuracy. Even if naive Bayes classifiers perform better than C4.5, they are not the best on the smaller datasets.



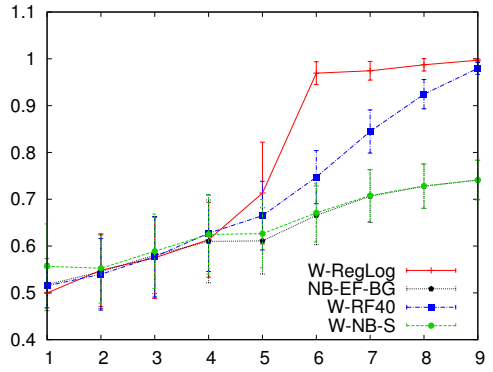
(a) Adult data set



(b) Mushroom data set



(c) Crx data set



(d) TicTacToe data set

Fig. 4. AUC versus the size of the training set (log_2)

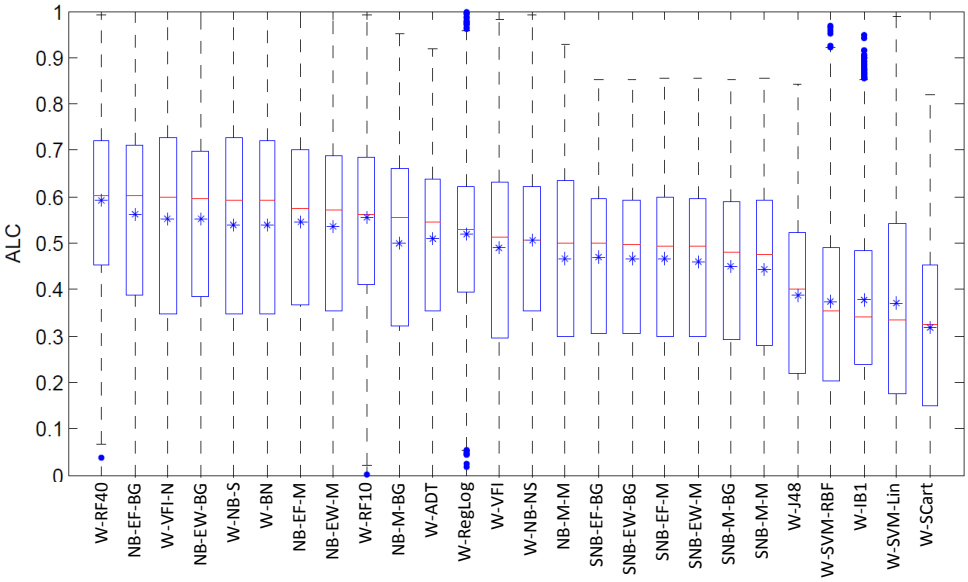


Fig. 5. Results variance using box plots

	W-ADT	W-BN	W-IB1	NB-EF-BG	NB-EF-M	NB-EW-BG	W-NB-NS	W-NB-S	W-RegLog	W-RF10	W-RF40	W-SVM-Lin	W-VFI	W-VFI-N
Adult	58.05	**64.28**	32.60	62.31	58.98	62.64	59.88	63.70	54.15	54.73	58.44	18.04	58.02	63.80
Australian	59.40	71.55	48.99	65.75	64.07	64.11	56.43	**72.15**	55.16	59.88	63.23	53.56	59.03	69.06
Breast	74.97	89.12	80.85	**92.31**	**92.31**	92.18	88.26	89.04	87.28	88.25	89.89	83.30	88.40	92.30
Bupa	20.17	2.02	8.79	14.81	14.81	13.20	12.52	1.98	**24.06**	21.74	23.99	3.02	9.05	9.32
Crx	60.16	70.12	47.73	63.91	62.79	62.32	56.30	**70.63**	54.59	59.02	62.47	51.44	57.33	66.57
German	25.98	13.10	11.23	27.61	27.61	26.13	30.53	13.12	26.69	27.29	**31.69**	18.58	24.84	26.89
Heart	47.19	62.19	42.42	61.31	58.26	60.21	54.56	62.84	51.25	53.64	57.15	47.33	49.75	**64.22**
Hepatitis	39.92	60.38	26.77	49.64	46.22	47.78	37.92	**63.16**	37.27	46.93	50.36	31.86	46.71	58.88
Horsecolic	52.73	59.81	41.35	51.87	44.01	52.06	40.61	61.58	48.31	57.34	**62.27**	36.28	54.20	58.31
Hypothyroid	59.21	50.78	25.42	65.87	**66.75**	64.06	52.81	48.77	59.27	63.05	66.13	24.89	52.94	60.41
Ionosphere	57.42	59.18	39.91	65.91	65.91	64.68	56.58	59.06	43.17	65.86	**70.29**	42.43	52.06	61.54
Mushroom	82.20	**92.23**	83.38	88.99	86.26	88.99	92.07	92.07	87.20	88.76	89.97	84.20	89.25	91.01
Pima	37.70	30.76	21.79	41.50	41.50	40.16	38.35	30.84	41.99	40.68	**44.43**	24.63	26.77	30.06
SickEuthyroid	60.46	52.57	22.90	59.44	59.29	59.50	50.28	49.25	57.49	61.87	**66.85**	3.70	45.41	47.93
Sonar	32.64	36.54	32.29	38.10	38.10	37.32	31.72	36.51	36.05	41.80	**48.32**	31.54	23.12	34.37
Spam	66.53	72.86	45.95	79.93	79.93	77.29	71.07	72.76	68.76	75.92	**80.08**	58.29	67.72	78.29
Tictactoe	32.40	28.40	30.55	26.91	20.52	26.91	28.73	28.73	**53.24**	37.77	41.83	16.32	26.00	27.61
Averaged ALC	51.01	53.88	37.82	56.25	54.55	55.27	50.51	53.89	52.11	55.56	**59.26**	37.02	48.86	55.33

	W-J48	NB-EW-M	NB-M-BG	NB-M-M	SNB-EF-BG	SNB-EF-M	SNB-EW-BG	SNB-EW-M	SNB-M-BG	SNB-M-M	W-SVM-RBF	W-SCart
Adult	41.68	59.25	64.06	56.63	55.25	55.13	55.73	55.52	57.92	56.02	15.41	40.97
Australian	51.60	62.32	65.60	63.22	53.20	54.13	52.04	53.04	54.85	57.18	42.16	42.70
Breast	66.84	92.18	76.24	76.24	79.29	79.29	79.40	79.40	76.30	76.30	81.05	57.85
Bupa	10.54	13.20	1.91	1.91	13.58	13.58	12.00	12.00	1.91	1.91	16.61	8.53
Crx	51.68	61.01	63.73	62.90	52.92	54.38	51.99	53.15	54.50	57.31	40.21	44.03
German	14.39	26.13	14.14	14.14	22.27	22.27	22.11	22.11	14.15	14.15	16.17	10.06
Heart	29.21	56.90	57.34	45.95	40.91	40.75	38.92	38.76	42.81	41.28	43.17	22.88
Hepatitis	21.05	44.66	46.83	40.49	30.99	32.43	31.32	32.50	30.52	31.34	25.28	10.62
Horsecolic	40.18	44.06	55.08	49.01	45.12	40.67	45.36	40.90	47.42	43.50	10.45	2.04
Hypothyroid	43.79	65.06	54.46	52.09	56.56	56.62	60.77	61.03	53.04	52.56	42.55	39.54
Ionosphere	42.89	64.68	53.04	53.04	55.15	55.15	55.48	55.48	53.42	53.42	45.23	36.27
Mushroom	77.71	86.26	88.99	86.26	82.56	81.77	82.56	81.77	82.56	81.77	79.76	74.62
Pima	25.34	40.16	29.71	29.71	37.65	37.65	37.56	37.56	29.89	29.89	26.63	20.08
SickEuthyroid	47.80	59.64	52.14	45.32	56.16	55.95	55.57	55.68	46.20	45.49	39.23	45.21
Sonar	19.40	37.32	31.97	31.97	25.26	25.26	25.21	25.21	27.12	27.12	29.37	16.20
Spam	52.67	77.29	66.63	66.63	65.93	65.93	60.53	60.53	66.33	66.33	55.96	46.11
Tictactoe	21.79	20.52	26.91	20.52	25.50	20.29	25.50	20.29	25.50	20.29	24.02	25.47
Average	38.74	53.57	49.93	46.83	46.96	46.54	46.59	46.17	44.97	44.46	37.25	31.95

TABLE III

ALC PER ALGORITHM AND DATA SET

	W-ADT	W-BN	W-IB1	NB-EF-BG	NB-EF-M	NB-EW-BG	W-NB-NS	W-NB-S	W-RegLog	W-RF10	W-RF40	W-SVM-Lin	W-VFI	W-VFI-N
Adult	28.64	39.37	23.07	40.85	30.41	40.73	29.73	38.23	30.01	30.58	33.46	9.02	37.73	**46.18**
Australian	45.86	65.32	42.35	57.02	53.94	55.99	44.12	**66.17**	45.52	48.43	52.13	42.89	51.11	62.37
Breast	61.52	83.70	75.44	88.89	88.89	**88.92**	82.76	83.63	82.09	82.88	85.20	77.85	84.21	88.86
Bupa	14.19	2.02	5.74	10.48	10.48	9.76	8.26	1.96	**17.75**	14.90	16.28	3.01	8.30	8.66
Crx	46.73	62.86	40.99	54.27	52.00	53.40	43.71	**63.59**	44.37	46.86	50.82	39.15	48.45	58.91
German	17.49	4.54	7.54	17.20	17.20	16.92	18.51	4.59	15.51	18.64	**21.74**	12.00	15.29	18.44
Heart	42.48	59.44	41.15	58.07	54.24	57.68	49.47	60.14	47.95	50.22	53.59	44.20	46.64	**61.78**
Hepatitis	36.16	57.85	24.18	46.30	41.94	45.26	32.16	**60.91**	35.04	43.53	46.90	28.87	43.34	55.85
Horsecolic	44.41	54.46	35.25	44.70	33.88	45.11	29.08	**57.08**	41.05	49.18	54.92	27.15	49.17	52.52
Hypothyroid	26.39	9.91	9.17	38.32	**38.81**	37.85	16.01	7.44	33.30	35.21	37.81	18.25	35.21	34.46
Ionosphere	45.63	47.00	30.88	58.20	58.20	56.82	44.75	46.89	35.35	55.36	**60.61**	32.75	37.83	50.35
Mushroom	61.75	**84.61**	65.36	77.27	71.14	77.27	84.41	84.41	73.47	75.56	78.09	67.80	77.66	82.42
Pima	28.26	19.00	17.53	31.50	31.50	31.17	26.76	19.12	30.58	32.82	**35.95**	18.55	26.51	31.45
SickEuthyroid	31.74	17.08	10.56	37.90	36.64	39.65	20.62	12.17	32.90	35.28	**42.33**	4.40	33.59	34.56
Sonar	26.88	31.71	26.03	33.12	33.12	33.15	26.72	31.65	33.80	36.10	**42.42**	27.83	22.51	33.67
Spam	44.03	54.31	29.49	68.25	68.25	67.19	55.39	54.33	57.14	60.64	66.77	46.13	55.83	**68.26**
Tictactoe	17.09	19.82	16.59	17.54	10.61	17.54	20.27	20.27	**27.38**	19.21	21.63	11.03	16.04	18.71
Average	36.43	41.94	29.49	45.88	43.01	45.55	37.22	41.92	40.19	43.26	47.10	30.05	40.55	**47.50**

	W-J48	NB-EW-M	NB-M-BG	NB-M-M	SNB-EF-BG	SNB-EF-M	SNB-EW-BG	SNB-EW-M	SNB-M-BG	SNB-M-M	W-SVM-RBF	W-SCart
Adult	11.21	30.34	40.53	19.72	23.43	20.65	23.48	21.34	24.63	17.83	2.04	4.35
Australian	38.85	52.99	57.47	52.79	36.58	37.57	36.15	37.43	39.30	42.05	27.54	24.83
Breast	54.86	88.92	63.19	63.19	67.84	67.84	68.03	68.03	63.14	63.14	74.68	40.97
Bupa	6.78	9.76	1.31	1.31	8.87	8.87	8.14	8.14	1.30	1.30	9.22	4.23
Crx	39.50	51.15	54.52	52.17	36.12	37.66	36.06	37.56	38.72	42.01	24.55	26.97
German	8.43	16.92	3.82	3.82	9.83	9.83	10.95	10.95	3.81	3.81	9.13	2.88
Heart	25.93	53.66	53.81	39.91	34.19	33.94	32.90	32.73	36.44	34.35	40.36	17.85
Hepatitis	18.19	41.07	43.88	35.83	26.66	27.93	26.98	28.10	25.99	26.20	22.47	8.54
Horsecolic	31.30	34.56	48.65	39.77	34.61	27.70	35.41	28.48	37.81	30.81	4.59	1.67
Hypothyroid	11.47	38.19	18.51	14.28	19.32	19.50	29.47	29.63	14.87	14.49	20.44	7.11
Ionosphere	31.15	56.82	39.30	39.30	41.67	41.67	42.30	42.30	38.53	38.53	33.14	21.60
Mushroom	52.99	71.14	77.27	71.14	62.30	60.64	62.30	60.64	62.30	60.64	59.59	46.68
Pima	18.38	31.17	15.81	15.81	24.77	24.77	26.04	26.04	15.86	15.86	19.37	9.92
SickEuthyroid	16.31	38.65	22.80	9.66	25.84	25.53	27.55	27.60	9.53	8.90	16.14	10.63
Sonar	15.39	33.15	26.04	26.04	18.60	18.60	19.24	19.24	21.24	21.24	25.02	11.52
Spam	32.00	67.19	41.92	41.92	38.96	38.96	35.31	35.31	40.13	40.13	38.09	21.05
Tictactoe	8.92	10.61	17.54	10.61	13.53	9.83	13.53	9.83	13.53	9.83	9.99	5.50
Average	24.80	42.72	36.85	31.60	30.77	30.09	31.40	30.79	28.65	27.71	25.67	15.66

TABLE IV

ALC PER ALGORITHM AND DATA SET WITH AREA BETWEEN 2 AND $2^6 = 64$ EXAMPLES DURING TRAINING

Algorithm	Avg rank	Avg ALC	Avg final AUC
W-RF40	**3.65**	**59.26**	**91.30**
NB-EF-BG	4.65	56.25	88.04
NB-EW-BG	6.59	55.27	86.82
W-VFI-N	6.94	55.33	83.13
NB-EF-M	7.24	54.55	88.04
W-RF10	7.29	55.56	89.70
W-BN	8.53	53.88	87.36
W-NB-S	8.82	53.89	87.35
NB-EW-M	9.18	53.57	86.80
W-RegLog	10.94	52.11	88.55
W-ADT	11.24	51.01	88.66
W-NB-NS	11.29	50.51	87.09
NB-M-BG	12.53	49.93	86.81
W-VFI	13.94	48.86	82.04
SNB-EF-BG	15.76	46.96	88.15
SNB-EW-BG	16.35	46.59	87.06
NB-M-M	16.65	46.83	86.84
SNB-EF-M	16.94	46.54	88.20
SNB-EW-M	17.47	46.17	86.99
SNB-M-BG	17.53	44.97	87.07
SNB-M-M	18.94	44.46	87.19
W-IB1	20.59	37.82	78.07
W-SVM-Lin	20.65	37.02	76.44
W-SVM-RBF	20.71	37.25	81.51
W-J48	22.18	38.74	82.84
W-SCart	24.41	31.95	81.70

TABLE V

AVERAGED RANK, AVERAGED ALC AND AVERAGED FINAL AUC

Algorithm	Avg rank	Avg ALC	Avg final AUC
W-RF40	**4.35**	47.10	**86.21**
W-VFI-N	4.65	**47.50**	82.43
NB-EF-BG	4.65	45.88	83.45
NB-EW-BG	5.41	45.55	82.34
W-RF10	6.88	43.26	84.09
NB-EF-M	7.65	43.01	83.29
NB-EW-M	8.47	42.72	82.12
W-BN	9.41	41.94	81.58
W-NB-S	9.47	41.92	81.28
W-RegLog	10.00	40.19	79.98
W-VFI	10.53	40.55	77.91
W-NB-NS	11.88	37.22	82.60
W-ADT	12.24	36.43	83.02
NB-M-BG	12.76	36.85	80.92
SNB-EW-BG	16.88	31.40	81.93
SNB-EF-BG	17.18	30.77	82.44
NB-M-M	17.59	31.60	80.29
W-SVM-Lin	17.59	30.05	72.97
SNB-EW-M	18.24	30.79	81.76
SNB-EF-M	18.59	30.09	82.20
W-IB1	18.71	29.49	72.46
SNB-M-BG	19.59	28.65	80.18
W-SVM-RBF	19.82	25.67	72.98
SNB-M-M	21.35	27.71	80.12
W-J48	21.94	24.80	74.14
W-SCart	25.18	15.66	70.52

TABLE VI

AVERAGED RANK, AVERAGED ALC AND AVERAGED FINAL AUC WITH AREA BETWEEN 2 AND $2^6 = 64$ EXAMPLES DURING TRAINING

Tree classifiers do not perform well out of the box (C4.5, Cart), but surprisingly in combination with bagging/boosting techniques they are the best of this study: Random Forest. Figure 4 shows Random Forest (with a forest size of 40: W-RF40) results on four datasets. W-RF40 is the overall best classifier in this study. The data sets Crx and TicTacToe on sub-figures 4(c) and 4(d) are favorable to naive Bayes and logistic regression, but even on them Random Forest is still performing well. In a general manner its performances are better than the other classifiers on most of the data sets.

It is surprising to discover a pretty unknown classifier close to the top of this study: VFI - a vote by majority based on a discretization technique. This algorithm is working pretty well without using the option "weight feature intervals by confidence": W-VFI-N. If the learning phase is stopped after the first 64 examples (Table VI), W-VFI-N is first on ALC and is ranked second.

2) Discriminative models analysis: Discriminative algorithms are represented in this study by the Random Forests (W-RF and W-RF40), the logistic regression (W-RegLog) and SVM (W-SVM-Lin, W-SVM-Log). The Random Forest algorithm set up with a size of 40 trees is the best performer in this study on all aspects: rank, averaged ALC and averaged final AUC. This algorithm ability to try numerous trees with just few features gives good results both at the beginning and at the end of the learning phase.

Even if the logistic regression is positioned after Random Forest, naive Bayes and VFI, its ALC score is not that far. In certain cases it is even much better than all the others classifiers as shown on the figure 4(d) for the TicTacToe date set.

Support Vector Machines (SVM) are not performing well on small data sets. Linear SVMs are a bit better than RBF ones but still their performances are one of the lowest in this study. These results are surprising and further experiments should be conducted to double check them.

3) Generative models analysis: Among the generative models tested in this study, naive Bayes are the most evaluated. The methods used to discretize continuous variables and grouping nominal variables have a great influence on the results. The best choice is constituted by the pairs ("Equal-Frequency", "Basic Grouping") followed by ("EqualWidth", "Basic Grouping"). The regularized discretization [30] and the MODL grouping method [32] are too robust to be able to build a model with just few examples. Regularized methods (MODL approach and selective naive Bayes) are known to have low bias-variance [37]. Therefore, in order to maintain this low bias-variance, their variance but also AUC will be lower on the smaller datasets. This statement is confirmed as the AUC increases later: the classifier is more conservative and prefers not to make highly variable predictions. This robustness makes these classifiers to have a lower score but still they are better than C4.5 or Cart. In this study "simpler methods" have the advantage to more rapidly find patterns and give better results.

Bayesian network represents non linear generative model

(see table I) in this study. It behaves relatively well but they are just below Random Forest, VFI and naive Bayes. On few data sets: Adult and Mushrooms (figure 4(b)), it is the best classifier in terms of ALC.

4) Comparisons with existing analysis: The results in the literature are confirmed in this study. A ranking could be proposed:

- Generative classifiers are better than discriminative classifiers when the number of examples is low [38] and when only one classifier is used.
- Ensemble of classifiers performs very well [39]: bagging [40] of discriminative classifiers performs very well and allows to have a reduced variance (compare to an only one discriminative classifier). In this study, the bagging of discriminative classifiers performs better than a single generative classifier.
- Ensemble of generative classifiers [41] has to be tested and compared to ensemble of discriminative classifiers.
- Regularized methods are robust [37].

Few studies on the learning performance versus the size of the learning data set have been identified in the introduction of this paper. This paragraph seeks if our empirical study conclusions are confirmed.

In [7] the training size and class distribution vary but they are using a different measure: the learning surface, different from the traditional learning curve used in our study. In [8], the authors focus on the learning time contrary to this study which focuses on the performance versus the size of the training set. Therefore both of these studies can not be compared with ours.

In [10] the focus is on the k nearest neighbor. Experimental results show that the nearest neighbor classifier designed on the bootstrap samples outperforms the conventional k-NN classifiers, particularly in high dimensions. Since the bootstrapping method used is closed to the bagging procedure their conclusion does not enter in conflict with our recommendations. In [11] linear classifiers are considered for very small sample sizes using a stability measure. One conclusion of their paper is: if classifiers become instable for small sample sizes, a general stabilizing technique like bagging could improved classifier performance. This conclusion is in accordance with the empirical study presented here. In [12] the influence of the training set size is evaluated on 4 computer-aided diagnosis problems using 3 different classifiers (C4.5, Linear and RBF SVM). They observe that classifier results on small data sets should not be generalized to larger. Their empirical results suggest that, given sufficient training data, SVMs tend to be the best classifiers. In [13] the authors study how the data set size affects bias and variance error decompositions for classification algorithms. The paper conclusion is in two parts. The first one indicates that there is no clear effect of training set size on bias. This is confirmed in our study since the mean rank does not really change between Table V and Table VI. The second part of the conclusion is that variance can be expected to decrease as the training set size increases. This second part confirms our results. Moreover the bagging procedure is able to reduce this variance [40].

5) Recommendation: The study presented in this paper recommends that few algorithms could be used to build a model on small training data sets: random forest and the naive Bayes classifier. Both these algorithms need to be parameterized to reach their best performances. The association ("EqualFrequency", "Basic Grouping") and ("EqualWidth", "Basic Grouping") are the best ones for the naive Bayes and a size of 40 trees for "Random Forest". If we only focus on the early learning stage, a method using vote on intervals (VFI) is ranked as the best methods in this study.

V. Conclusion and Future Works

A. Towards a new criterion?

We previously presented our results in two tables: one containing the results of the area under the AUC curve (ALC) finishing at 2^6 examples (table VI) and another one containing the final AUC (table V). The first table shows the algorithms behaviors at the early learning stage and the second one its performance at the end. On figure 6 the two axis use these two evaluating criterion. On average only "W-RFxx" and "NB-EF-BG" manage to be well positioned on these two criterion. Logistic regression and trees boosting (ADTree) have good final performances but there ALC_{64} is relatively low. On the contrary "VFI" is good on this criterion but its final AUC is not that good.

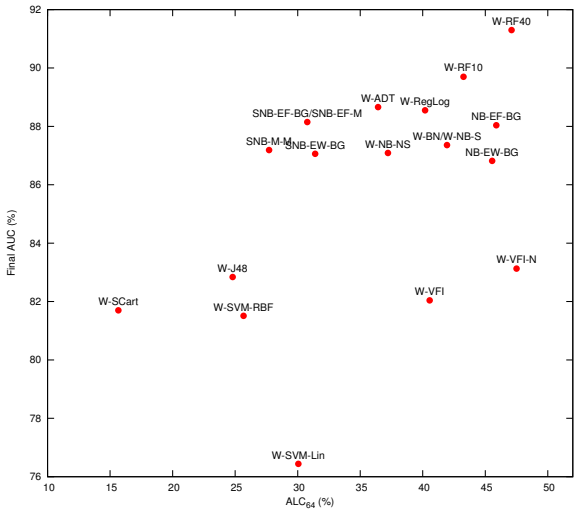


Fig. 6. Positioning of the different algorithms in terms of ALC_{2^6} and AUC_{final} (to avoid text overlapping the classifiers NB-EF-M, NB-EW-M, NB-M-BG, NB-M-M, SNB-EW-M and SNB-M-BG are omitted)

This analysis let us think that the ALC criterion could be replaced by a new criterion which would take into account both of these aspects. Indeed, it would be interesting to have a synthetic criterion allowing us to evaluate algorithm performances on these two axis. This criterion could be written as follow:

$$Criterion = \frac{ALC_{2^6} + AUC_{final}}{2}$$

B. Algorithms combination

In this study very few cases were found where a particular algorithm is very good on the early stage and another one is very good at the end of the learning phase. However if we would be in such situation as it is shown on figure 7, it would be interesting to use two different algorithms: W-VFI-N for the 2^8 first examples and then a selective naive Bayes at the end.

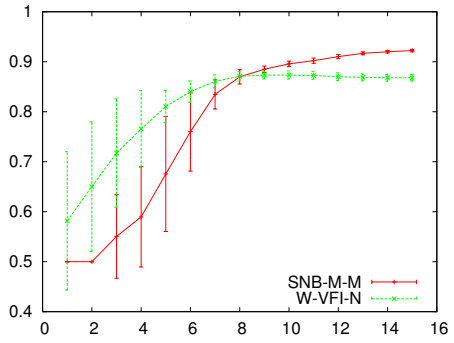


Fig. 7. A data set (Adult) where using 2 algorithms could be beneficial

REFERENCES

[1] I. Guyon, V. Lemaire, G. Dror, and D. Vogel, "Analysis of the kdd cup 2009: Fast scoring on a large orange customer database," *JMLR: Workshop and Conference Proceedings*, vol. 7, pp. 1–22, 2009.

[2] R. Féraud, M. Boullé, F. Clérot, F. Fessant, and V. Lemaire, "The orange customer analysis platform," in *Industrial Conference on Data Mining (ICDM)*, 2010, pp. 584–594.

[3] B. Settles, "Active learning literature survey," 2010, http://pages.cs.wisc.edu/~bsettles/pub/settles.activelearning.pdf.

[4] R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac, "The Multi-Purpose incremental Learning System AQ15 and its Testing Application to Three Medical Domains," *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 1041–1045, 1986.

[5] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 2000, pp. 71–80.

[6] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 2003, pp. 523–528.

[7] G. Forman and I. Cohen, "Learning from little: Comparison of classifiers given little training," *Knowledge Discovery in Databases: PKDD 2004*, pp. 161–172, 2004.

[8] T. Lim, W. Loh, and Y. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine learning*, vol. 40, no. 3, pp. 203–228, 2000.

[9] Y. Muto and Y. Hamamoto, "Improvement of the parzen classifier in small training sample size situations," *Intelligent Data Analysis*, vol. 5, no. 6, pp. 477–490, 2001.

[10] Y. Hamamoto, S. Uchimura, and S. Tomita, "A bootstrap technique for nearest neighbor classifier design," *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 1, JANUARY 1997*, vol. 19, no. 1, pp. 73–79, 1997.

[11] M. Skurichina and R. P. W. Duin, "Stabilizing classifiers for very small sample sizes," in *Proceedings of the 13th International Conference on Pattern Recognition - Volume 2*, 1996, pp. 891–.

[12] A. Basavanhally, S. Doyle, and A. Madabhushi, "Predicting classifier performance with a small training set: Applications to computer-aided diagnosis and prognosis," in *Biomedical Imaging: From Nano to Macro. IEEE International Symposium*, 2009.

[13] D. Brain and G. I. Webb, "On the effect of data set size on bias and variance in classification learning," in *Proceedings of the Fourth Australian Knowledge Acquisition Workshop*, 1999, pp. 117–128.

[14] R. Fisher, "The use of multiple measurements in taxonomic problems." *Annals of eugenics*, vol. 7, pp. 179–188, 1936.

[15] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in *Proceedings of the National Conference on Artificial Intelligence*, no. 415, 1992, pp. 223–223.

[16] J. Gama, P. Medas, and P. Rodrigues, "Learning Decision Trees from Dynamic Data Streams," in *Proceedings of the ACM Symposium on Applied ComputingLearning Decision Trees from Dynamic Data Streams*, 2005.

[17] I. H. Witten and E. Frank, *Data mining: practical machine learning tools and techniques*. Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, second edition, 2005.

[18] M. Boullé, "Khiops: A Statistical Discretization Method of Continuous Attributes," *Machine Learning*, vol. 55, no. 1, pp. 53–69, Apr. 2004.

[19] G. John and P. Langley, "Estimating continuous distributions in Bayesian classifiers," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1995, pp. 338–345.

[20] R. Bouckaert, "Bayesian Network Classifiers in Weka," 2004, http://weka.sourceforge.net/manuals/weka.bn.pdf.

[21] D. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Machine learning*, vol. 6, no. 1, pp. 37–66, 1991.

[22] S. L. Cessie and J. V. Houwelingen, "Ridge estimators in logistic regression," *Applied Statistics*, 1992.

[23] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001, available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[24] Y. EL-Manzalawy and V. Honavar, *WLSVM: Integrating LibSVM into Weka Environment*, 2005, available at http://www.cs.iastate.edu/~yasser/wlsvm.

[25] Y. Freund and L. Mason, "The alternating decision tree learning algorithm," in *Machine learning*, 1999, pp. 124–133.

[26] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[27] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*. Chapman and Hall/CRC, 1984.

[28] L. Breiman, "Random forests," *Machine learning*, vol. 25, no. 2, pp. 5–32, 2001.

[29] G. Demiröz and H. Güvenir, "Classification by voting feature intervals," *Machine Learning: ECML-97*, pp. 85–92, 1997.

[30] M. Boullé, "Regularization and Averaging of the Selective Naive Bayes classifier," *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pp. 1680–1688, 2006.

[31] ——, "MODL: A Bayes optimal discretization method for continuous attributes," *Machine Learning*, vol. 65, no. 1, pp. 131–165, May 2006.

[32] ——, "A grouping method for categorical attributes having very large number of values," *Machine Learning and Data Mining in Pattern Recognition*, pp. 228–242, 2005.

[33] C. L. Blake and C. J. Merz, "UCI Repository of machine learning databases," 1998, http://archive.ics.uci.edu/ml/ last visited: 15/09/2010. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[34] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," *Machine Learning*, vol. 31, pp. 1–38, 2004.

[35] I. Guyon, G. Cawley, G. Dror, and V. Lemaire, "Results of the Active Learning Challenge," in *JMLR W&CP, Workshop on Active Learning and Experimental Design, collocated with AISTATS, Sardinia, Italy*, vol. 10, 2010, pp. 1–26.

[36] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Machine learning*, vol. 130, pp. 103–130, 1997.

[37] F. Cucker and S. Smale, "Best Choices for Regularization Parameters in Learning Theory: On the Bias-Variance Problem," *Foundations of Computational Mathematics*, vol. 2, no. 4, pp. 413–428, 2008.

[38] G. Bouchard and B. Triggs, "The tradeoff between generative and discriminative classifiers," in *IASC International Symposium on Computational Statistics (COMPSTAT)*, 2004, pp. 721–728.

[39] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine learning*, vol. 36, no. 1, pp. 105–139, 1999.

[40] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[41] A. Prinzie and D. Van den Poel, "Random multiclass classification: Generalizing random forests to random mnl and random nb," in *Database and Expert Systems Applications*. Springer - Lecture Notes in Computer Science, 2007, pp. 349–358.