Research Repository UCD

Title	Kalman Filter-based Heuristic Ensemble (KFHE): A new perspective on multi-class ensemble classification using Kalman filters
Authors(s)	Pakrashi, Arjun, MacNamee, Brian
Publication date	2019-06
Publication information	Pakrashi, Arjun, and Brian MacNamee. “Kalman Filter-Based Heuristic Ensemble (KFHE): A New Perspective on Multi-Class Ensemble Classification Using Kalman Filters.” Elsevier, June 2019. https://doi.org/10.1016/j.ins.2019.02.017.
Publisher	Elsevier
Item record/more information	http://hdl.handle.net/10197/9680
Publisher's statement	This is the author’s version of a work that was accepted for publication in Information Sciences. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Information Sciences (485 (2019)) https://doi.org/10.1016/j.ins.2019.02.017
Publisher's version (DOI)	10.1016/j.ins.2019.02.017

Downloaded 2026-05-01 23:36:18

The UCD community has made this article openly available. Please share how this access benefits you. Your story matters! (@ucd_oa)

© Some rights reserved. For more information

Kalman Filter-based Heuristic Ensemble (KFHE): A new perspective on multi-class ensemble classification using Kalman filters

Arjun Pakrashi*, Brian Mac Namee

Insight Centre for Data Analytics, School of Computer Science, University College Dublin, Ireland

Abstract

This paper introduces a new perspective on multi-class ensemble classification that considers training an ensemble as a state estimation problem. The new perspective considers the final ensemble classifier model as a static state, which can be estimated using a Kalman filter that combines noisy estimates made by individual classifier models. A new algorithm based on this perspective, the Kalman Filter-based Heuristic Ensemble (KFHE), is also presented in this paper which shows the practical applicability of the new perspective. Experiments performed on 30 datasets compare KFHE with state-of-the-art multi-class ensemble classification algorithms and show the potential and effectiveness of the new perspective and algorithm. Existing ensemble approaches trade off classification accuracy against robustness to class label noise, but KFHE is shown to be significantly better or at least as good as the state-of-the-art algorithms for datasets both with and without class label noise.

Keywords: Classification, Multi-class, Ensemble, Kalman filter, Heuristic

1. Introduction

An ensemble classification model is composed of multiple individual base classifiers, also known as component classifiers, the outputs of which are aggregated together into a single prediction. The classification accuracy of an ensemble model can be expected to exceed that of any of its individual base classifiers. The main motivation behind ensemble techniques is that a committee of experts working together on a problem are more likely to accurately solve it than a single expert working alone [23]. Although many existing ensemble

*Corresponding author

Email addresses: arjun.pakrashi@ucdconnect.ie, arjun.pakrashi@insight-centre.org (Arjun Pakrashi), brian.macnamee@ucd.ie (Brian Mac Namee)

techniques (e.g. [4, 16, 20, 45]) have been repeatedly shown in benchmark experiments to be effective (see [28, 29]), current approaches still have limitations. For example, methods based on *bagging*, although robust, may not lead to models as accurate as those learned by more sophisticated methods such as those based on *boosting* [28]. Methods based on boosting, however, are sensitive to class-label noise and the presence of outliers in training datasets [9].

To address the limitations of current multi-class classification ensemble algorithms, this paper presents a new perspective on ensemble model training, framing it as a state estimation problem that can be solved using a Kalman filter [22, 25]. Although Kalman filters are most commonly used to solve problems associated with time series data, this is not the case in this work. Rather, this work exploits the data fusion property of the Kalman filter to combine individual multi-class component classifier models to construct an ensemble.

The new perspective views the ensemble model to be trained as an unknown static state to be estimated. A Kalman filter can be used to estimate an unknown static state by combining multiple uncertain measurements of the state. This exploits the data fusion property of the Kalman filter. In the new perspective the measurements are the single component classifiers in the ensemble, and the uncertainties of these measurements are based on the classification errors of the single component classifiers. The Kalman filter is used to combine the component classifier models into an overall ensemble model. This new perspective on ensemble training provides a framework within which different algorithms can be formulated. This paper describes one such new algorithm, the Kalman Filter-based Heuristic Ensemble (KFHE). In an evaluation experiment KFHE is shown to out-perform methods based on boosting while maintaining the robustness of methods based on bagging. The contributions of this paper are:

1. A new perspective on training multi-class ensemble classifiers, which views it as a state estimation problem and solves it using a Kalman filter [22, 25].
2. A new multi-class ensemble classification algorithm, the Kalman Filter-based Heuristic Ensemble (KFHE).
3. Extensive experiments comparing KFHE with the state-of-the-art ensemble algorithms that demonstrate the effectiveness of KFHE in both scenarios of noise free and noisy class-labels.

The remainder of this paper is structured as follows. Section 2 discusses previous work on multi-class ensemble classification algorithms and provides a brief introduction to the Kalman filter. Section 3 introduces the new Kalman filter-based perspective on building multi-class classification ensembles. The Kalman Filter-based Heuristic Ensemble (KFHE) method based on this perspective is described in Section 4. The setup of an experiment to evaluate the performance of KFHE and the comparison method to state-of-the-art approaches on a selection of datasets is described in Section 5, and a detailed discussion of the results of this experiment is presented in Section 6. Finally, Section 7 reflects on the newly proposed perspective and explores directions for future work.

2. Background

This section first reviews existing multi-class ensemble classification methods. Relevant aspects of the Kalman filter approach for state estimation, which serve as a basis for the explanation of KFHE, are then introduced.

2.1. Ensemble methods

The advent of ensemble approaches in machine learning in the early 1990s was due mainly to works by Hansen and Salamon [19], and Schapire [34]. Hansen and Salamon [19] showed that multiple classifiers could be combined to achieve better performance than any individual classifier. Schapire [34] proved that the *learnability* of strong learners and weak learners are equivalent, and then showed how to boost weak learners to become strong learners. Since then many alternative and improved approaches to build ensembles have been introduced. Ensemble methods can still, however, be categorised into three fundamental types: *bagging*, *boosting*, and *stacking*.

Bagging [4] or bootstrap aggregation, trains several base classifiers on bootstrap samples of a training dataset and combines the outputs of these base classifiers using simple aggregation such as majority voting. Training models on different samples of the training set introduces diversity into the ensemble, which is key to making ensembles work effectively. *UnderBagging* [2] is a variation of bagging addressing imbalanced datasets that performs undersampling before every bagging iteration, but also keeps all minority class instances in every iteration. The *Random Forest* [5] is an extension to bagging in which base classifiers (usually decision trees) are trained using a bootstrap sample of the dataset that has also been reduced to only a small random sample of the input space. The *Rotational Forest* [32] is another extension that attempts to build base classifiers that are simultaneously accurate and diverse. The input dataset is transformed by applying PCA [21] on different subsets of the attributes of the dataset, and axis rotation is performed by combining the coefficient matrices found by PCA for each subset. This is repeated multiple times. *Local Linear Forests* modify random forests by considering random forests as an adaptive kernel method and combining it with local linear regression [14].

Boosting [45] approaches iteratively learn component classifiers such that each one specialises on specific types of training examples. Each component classifier is trained using a weighted sample from a training dataset such that at each iteration the ensemble emphasises training examples that were misclassified in the previous iteration. Since the introduction of the original boosting algorithm, *AdaBoost* [13], several new approaches to boosting have been proposed. In *LogitBoost* [15], the logistic loss function is minimised while combining the sub-classifiers in a binary classification context. A linear programming approach to boosting, *LPBoost* [7], was shown to be competitive with AdaBoost. This algorithm minimises the misclassification error and maximises the soft margin in the feature space generated by the predictions of the weak hypothesis components of the ensemble. A multi-class modification for binary class AdaBoost

was introduced in [13], and an improvement of it was proposed in [20]. *Rot-Boost* [43] is a direct extension of the rotational forest approach [32] to include boosting. The *Gradient Boosting Machine* (GBM) [16] is a sequential tree based ensemble method, where each tree corrects the errors of the previously trained trees. *Stochastic Gradient Boosting Machine* (S-GBM) [17] improves GBM by training the component trees on bootstrap samples.

AdaBoost is sensitive to noisy class labels and performs poorly as the level of noise increases [12]. This is mainly due to the exponential loss function AdaBoost uses to optimise the ensemble. If a training datapoint has noisy class-labels AdaBoost will increase its weight for the next iteration and keep on increasing the weight of the datapoint in a vain attempt to classify it correctly. Therefore, given enough such noisy class-labelled datapoints AdaBoost can learn classifiers with poor generalisation ability. Although the performance of bagging decreases in the presence of class-label noise, it does not do so as severely as it does with AdaBoost [9].

To overcome this problem with noisy class-labeled datasets, *MadaBoost* [10] was proposed. MadaBoost changes the standard AdaBoost weight update rule by capping the maximum value for the weight of a datapoint to be 1. Similarly *FilterBoost* [3] optimises the log loss function, leading to a weight update rule which caps the weight upper bound of a datapoint to 1 using a smooth function. *BrownBoost* [12] and *Noise Detection Based AdaBoost (ND_AdaBoost)* [6] make AdaBoost more robust to class label noise by explicitly identifying noisy examples and ignoring them. *Robust Multi-class AdaBoost* (*Rob_MulAda*) [37] is an extension to *ND_AdaBoost* for multi-class classification. *Vote-Boosting* [33], decides the weights of each datapoint while training based on the disagreement of the predictions of the component classifiers that exist at each iteration. For lower levels of class-label noise, the datapoints with higher disagreement rates are emphasised. Whereas for higher levels of class-label noise, datapoints which agree among different component classifiers are highlighted, in an attempt to achieve robustness to class-label noise. A comprehensive review and analysis of the different boosting variations can be found in [44].

Stacking [42, 38] is a two stage process in which the outputs of a collection of first stage base classifiers are combined by a second stage classifier to produce a final output. Seewald [35], empirically showed that the extension to stacking by Ting and Witten [38] does not perform well in the multi-class context, and proposed *StackingC* to overcome this drawback. In [26] the weaknesses of StackingC were highlighted and were shown to occur due to increasingly skewed class distributions because of the binarisation of the multi-class problem. Next, a three layered improved stacked method for multi-class classification, *Troika* [26], was proposed. The stacking approach to building ensembles has received much less research attention than approaches based on bagging and boosting.

2.2. The Kalman filter

The Kalman filter [22] is a mathematical tool for stochastic estimation of the state of a linear system based on noisy measurements. Let there be a system which evolves linearly over time, and assume that the state of the system,

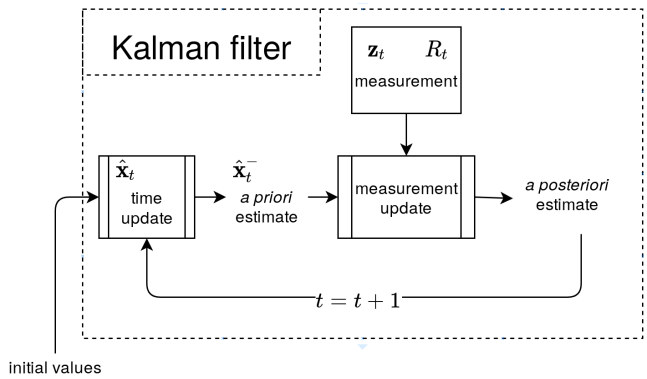


Figure 1: A high-level illustration of a Kalman filter

which is unobservable, has to be estimated at each time step, t. The state may be estimated in two ways. First, a linear model, which is used to update the state of the system from step t to step $t + 1$, can be used to get an *a priori* estimate of the state. This estimate will have a degree of uncertainty as the linear model is unlikely to fully capture the true nature of the system. Estimating the state using this type of linear model is commonly known as a *time update* step. Second, an external sensor can provide a state estimate. This estimate will also have an associated uncertainty, referred to as *measurement noise*, and introduced because of inaccuracies in the measurement process.

Given these two state estimates, and their related uncertainties, the Kalman filter combines the *a priori* estimate and the measurement to generate an *a posteriori* state estimate, such that the uncertainty of the *a posteriori* estimate is minimised. This combination of a sensor measurement with an *a priori* estimate is commonly known as the *measurement update* step. The process iterates using the *a posteriori* estimate calculated in a measurement update step as input to the time update step of the next iteration. A high-level illustration of the Kalman filter is shown in Figure 1. More formally, the time update step in a Kalman filter can be defined as:

$$\hat{\mathbf{x}}_t^- = A_t \hat{\mathbf{x}}_{t-1} + B_t \mathbf{u}_t \tag{1}$$

$$P_t^- = A_t P_{t-1} A_t^T + Q_t \tag{2}$$

where:

- $\hat{\mathbf{x}}_t^-$ is the *a priori* estimate at step t when the knowledge of the state in the previous step $(t-1)$ is given
- $\hat{\mathbf{x}}_{t-1}$ is the *a posteriori* estimate at step $(t-1)$, which is found through combining the *a priori* estimate and the measurement
- A_t is the state transition matrix which defines the linear relationship between $\hat{\mathbf{x}}_{t-1}$ and $\hat{\mathbf{x}}_t$

- $\mathbf{u}_t$ is the control input vector, containing inputs which changes the state based on some external effect
- B_t is the control input matrix applied to the control input vector
- P_t^- is the covariance matrix representing the uncertainty of the *a priori* estimate
- P_{t-1} is the covariance matrix representing the uncertainty of the *a posteriori* estimate at step $t-1$
- Q_t is the process noise covariance matrix, induced during the linear update

Similarly, the measurement update step can be defined as:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + K_t(\mathbf{z}_t - H_t\hat{\mathbf{x}}_t^-) \tag{3}$$

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + R_t)^{-1} \tag{4}$$

$$P_t = (I - K_t H_t) P_t^- \tag{5}$$

where

- $\mathbf{z}_t$ is the measurement of the system at time t
- R_t is the measurement noise covariance matrix
- H_t is a transformation matrix relating the state space to the measurement space (when they are the same space, then H_t can be the identity matrix)
- K_t is the *Kalman gain* which drives the weighted combination of the measurement and the *a priori* state
- I indicates the identity matrix

The Kalman filter iterates through the time update and the measurement update steps. In this work time steps are considered equidistant and discreet. Hence, from this point, "time step" and "iteration" will be used interchangeably. At $t = 0$, an initial estimate for $\hat{\mathbf{x}}_0$ and P_0 is used. Next, the time update step is performed using Eq. (1) and (2) to get $\hat{\mathbf{x}}_t^-$ and P_t^- respectively. The measurement $\mathbf{z}_t$ and its related uncertainty R_t are then obtained from a sensor or other appropriate source. These are combined with the a priori estimate using the measurement update step to find $\hat{\mathbf{x}}_t$ and P_t using Eq (4), (3) and (5), which are then used in the next iteration $(t+1)$. A detailed explanation of Kalman filters can be found in [22, 25], and an intuitive description in [40].

It should be emphasised here that, although a Kalman filter is used and Kalman filters are most commonly used with time series data, the proposed method *does not* perform time series prediction. Rather the focus is on multi-class classification and the data fusion property of the Kalman filter is used to combine the individual multi-class classifiers in the ensemble. Also, the term

"ensemble" in this work relates to multi-class ensemble classifiers, and should not be confused with Ensemble Kalman Filters (EnKF) [11].

Apart from their applications to time series data and sensor fusion, Kalman filters have been used previously in a small number of supervised and unsupervised machine learning applications. For example, [36], improves the predictions of a neural network using Kalman filters, although this method is essentially a post-processing of the results of a neural network output. Properties of a Kalman filter were used in combination with heuristics in population-based metaheuristic optimisation algorithms [39, 27], and in an unsupervised context in clustering [31, 30]. To the best of the authors' knowledge this is the first application of Kalman filters to training multi-class ensemble classifiers.

3. Training multi-class ensemble classifiers using a Kalman filter

This section introduces the new perspective on training multi-class ensemble classifiers using a Kalman filter. First, a toy example of static state estimation using a Kalman filter is presented, and then the new perspective is described.

3.1. A static state estimation problem: Estimating voltage level of a battery

Imagine that the exact voltage of a DC battery (which should remain constant) is unknown and needs to be estimated. A sensor is available to measure the voltage level of the battery. The measurements made by this sensor are unfortunately noisy, but the uncertainty associated with the measurements is known. This is a simple example of a static state estimation problem that can be solved by taking multiple noisy sensor measurements of the battery's voltage, and combining these into a single accurate estimate using a Kalman filter.

The Kalman filter can be applied in this scenario as follows. As it is known that the voltage of the battery does not change the state transition matrix, A_t, in Eq. (1) is the identity matrix; the control input matrix, B_t, in Eq. (2) is non-existent; and the process noise covariance matrix, Q_t, in Eq. (2) is considered to be zero. The voltage read by the sensor at a particular measurement, and the related uncertainty of the value due to the limited accuracy of the sensor, give $\mathbf{z}_t$ and R_t in Eq. (3) and (4) respectively. Given this information, the Kalman filter time update and measurement update steps can be performed to combine the current estimated voltage, $\hat{\mathbf{x}}_t^-$, and the measurement, $\mathbf{z}_t$, to get a new and better estimate of the voltage. The process can be repeated, where at each step, a new voltage measurement from the sensor is received, which is then combined with the current estimated voltage value using the measurement update step.

Note that, after t iterations, the estimated voltage is a combination of the sensor output values, where the Kalman gain, K_t in Eq (4) and (5), is controls the influence of each measurement in the combination. Therefore, after t iterations, the estimated voltage, $\hat{\mathbf{x}}_t$, can be seen as an ensemble of the values received from the sensor, which are optimally combined. This same idea can be applied to combine noisy base classifiers into a more accurate ensemble model.

3.2. Combining multi-class classifiers using the Kalman filter

A machine learning algorithm learns a hypothesis for a specific problem. Assume that all possible hypotheses make a *hypothesis space*[1], as described in [8]. Any point in the hypothesis space represents one hypothesis. For a specific problem, there is at least one ideal hypothesis within this hypothesis space which the learning algorithm tries to reach. Different hypotheses within the hypothesis space differ in their trainable parameters, and the machine learning algorithm modifies these parameters. Therefore, the training process can be seen as a search through the hypothesis space toward the ideal hypothesis.

The perspective presented in this paper views the ideal hypothesis as the static *state* to be estimated, and the hypothesis space as a *state space*. When an individual component classifier, h_t, is trained, it can be seen as a point in the hypothesis space. Here, h_t can be considered as an attempt to measure the ideal state with a related uncertainty indicated by the training error of h_t. The Kalman filter can be used to estimate the ideal state by combining these multiple noisy measurements. The combination of these noisy measurements leads to an estimation of the state that is expected to be more accurate than the individual measurements, and so an ensemble classification model that is more accurate than its component classifiers.

This is illustrated in Figure 2. The vertical axis is an abstract representation of the *hypothesis space* with each point along this axis representing a possible hypothesis. The star symbol on the vertical axis indicates the ideal hypothesis for a specific classification problem. The horizontal axis in Figure 2 represents training iterations proceeding from left to right. The circles are the estimates of the hypothesis at a time step t (the combination of all models added to the ensemble to this point in the training process), and the plus symbols represent the measurement of the hypothesis at a time step t (the last model added to the ensemble). The dashed and solid arrows connecting the state estimates indicate the combination of the measurement and the *a priori* estimate respectively. The goal of the process is to reach a hypothesis as close as possible to the ideal hypothesis (indicated by the horizontal line marked with a star) by combining multiple individual hypotheses using a Kalman filter.

To help with understanding the new perspective, the Kalman filter-based approach to ensemble training can be directly mapped back to the DC battery voltage estimation example described in Section 3.1. The ensemble model capturing the ideal hypothesis is equivalent to the actual voltage level of the DC battery. An individual component classifier, h_t, is analogous to the output from the voltage sensor. The classification error of the model h_t maps to the uncertainty related to the voltage sensor measurements. Just as the estimated voltage after t iterations can be thought as an *ensemble of sensor measurements* in the battery voltage estimation case; the trained individual classifiers, h_t combined

[1] The term hypothesis and hypothesis space is used to introduce the high level idea in connection with [8], but the terms model and model space will be used synonomoulsy throughout this text.

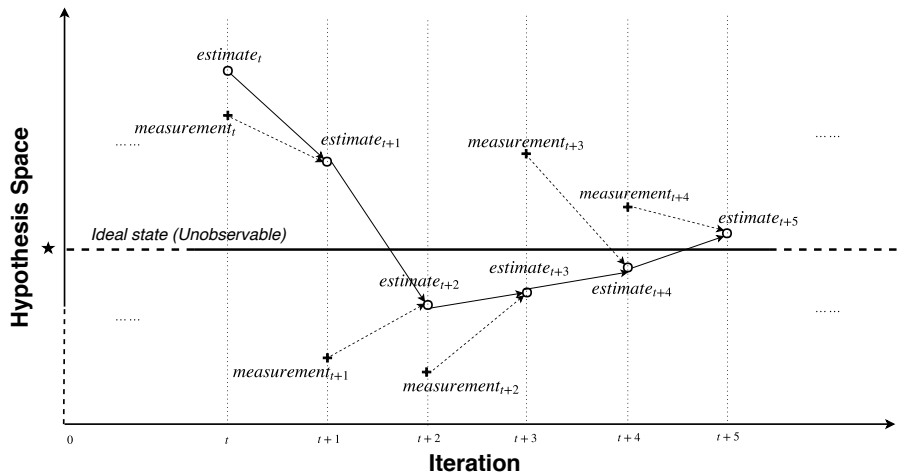


Figure 2: An illustration of the state estimation perspective on training a classification ensemble. The vertical axis indicates the hypothesis space and the horizontal axis represents iterations. Any point along the vertical axis represents a hypothesis. The horizontal line marked with the star indicates the ideal hypothesis. The circles are the estimates, and the plus symbols are the measurements of the hypothesis. The absolute distance of the circles and the plus symbols from the bold horizontal line indicating the ideal hypotheses represents the uncertainty of the estimates. The target of the training algorithm is to navigate through the hypothesis space to get as close to the ideal hypothesis as possible.

using the Kalman filter leads to an *ensemble of classifier models*.

4. Kalman Filter Heuristic-based Ensemble (KFHE)

This section provides a detailed description of the Kalman Filter-based Heuristic Ensemble (KFHE) algorithm, based on the new perspective proposed in Section 3. First, Section 4.1 presents an overview of the algorithm and connects the high-level concepts from Section 3. Sections 4.2, 4.3 and 4.4 then discuss the details of the algorithm.

4.1. Algorithm overview

In KFHE the Kalman filter used to estimate an ensemble classifier, as described in Section 3, is referred to as the *model Kalman filter*, abbreviated to *kf-m*. To implement *kf-m*, the following questions must be answered:

1. What should constitute a state?
2. How should the time update step be defined?
3. What should constitute a measurement?
4. How should measurement uncertainty be evaluated?

The *kf-m* state estimates are essentially the trained component classifiers. A model specification (for example the rules encoded in a decision tree or weight

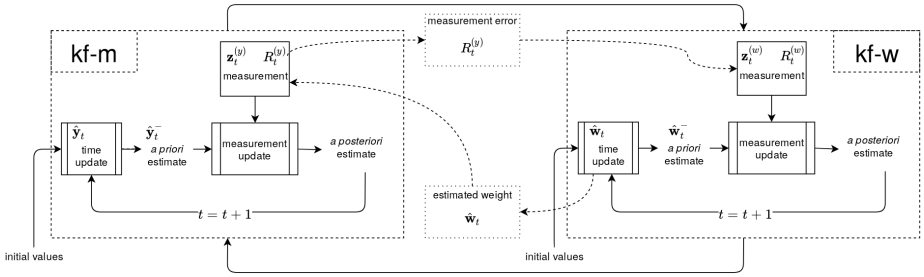


Figure 3: Overall dataflow between *kf-m* and *kf-w*

values in neural network) cannot be used directly as a state within the Kalman filter framework. Instead the predictions made by a component classifier for the instances in the training dataset are used as the representation of the state, as shown in Figure 4. This allows states to be combined using the equations in Section 2.2. This representation is explained in detail in Section 4.2.

Heuristics are used to address the remaining questions. The time update step is implemented as the identity function, as it can be assumed that the ideal state is static and does not change over time (as indicated by the horizontal line in Figure 2). The measurement is a function of the output of the multi-class classifier trained at the tth iteration. This model is trained using a weighted sample from the overall training dataset. The classification error of the model trained at the tth iteration, measured against its predictions for the full training set, is used as the uncertainty of the measurement.

A Kalman filter is then used to combine a measurement, which is the classification model at step t represented as shown in Figure 4, and the *a priori* estimate to get an *a posteriori* estimate. The *a posteriori* state estimate at the tth iteration is considered the ensemble classifier up to the tth iteration. This *a posteriori* estimate is used in the next iteration, and the process continues until a stopping condition is met. As the uncertainties of the estimates are represented as the classification errors, the process continues towards estimating states expected to yield lower classification errors.

The use of weighted samples from the training set to train component classifiers at each step of the *kf-m* process gives rise to another question: how should the weights for the weighted sampling of the training dataset be decided? In KFHE the answer is through another Kalman filter, which is referred to as the *weight Kalman filter* and abbreviated to *kf-w*. The *kf-w* Kalman filter works very similarly to *kf-m*, but estimates sampling weights for the training dataset instead of the overall model state. This is described in detail in Section 4.3.

The interactions between the model Kalman filter, *kf-m*, and the weight Kalman filter, *kf-w*, are illustrated in Figure 3. Essentially *kf-w* provides weights for the measurement step in *kf-m*, and *kf-m* provides measurement errors back to *kf-w* for its measurement step. The training process is summarised in Algorithm 1 and the following subsections describe the workings of *kf-m* and *kf-w* in detail.

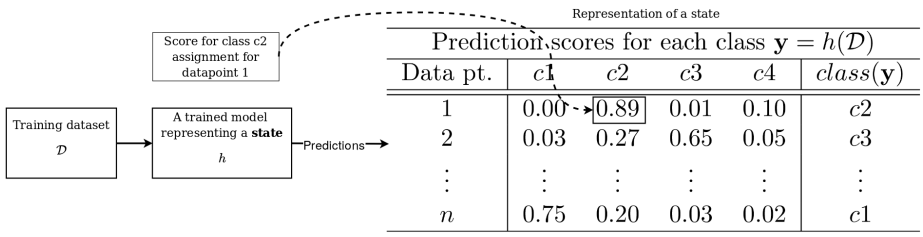


Figure 4: The representation of a state for *kf-m*.

4.2. The model Kalman filter: kf-m

The model Kalman filter, *kf-m,* estimates the ensemble classifier by combining component classifiers into a single ensemble classification model. This is a static estimation problem as the state to be estimated, the ideal ensemble classifier, does not change over time. For this reason the time update step for *kf-m* is the identity function and the *a posteriori* estimate of iteration $t-1$ is directly transferred to the *a priori* estimate at iteration t.

The trained base classifiers of the ensemble (the measurements) or the *a posteriori* state estimate (ensemble classifier) themselves are not directly usable as a state in the Kalman filter framework. Therefore a proxy numerical representation is required to perform the computations. The proxy representation of the state is shown in Figure 4 where each row represents a datapoint from the training set and the estimate scores for the classes for the corresponding datapoint. The class membership is determined by taking the class with the maximum score, and this membership is expressed as $class(\mathbf{y})$. For example in Figure 4, the first datapoint has the highest prediction score assigned to class-label $c2$, and thus the first datapoint is considered as a member of class $c2$. This representation of a model is used as the state in the Kalman filter framework.

Hence, the time update equations for *kf-m* are very simply defined as:

$$\hat{\mathbf{y}}_t^- = \hat{\mathbf{y}}_{t-1} \tag{6}$$

$$P_t^{(y)^-} = P_{t-1}^{(y)} \tag{7}$$

where

- $\hat{\mathbf{y}}_{t-1}$ is the *a posteriori* estimate from the previous iteration and $\hat{\mathbf{y}}_t^-$ represents the *a priori* estimate at the present iteration. These are the predictions of the ensemble model at the tth iteration in the representation shown in Figure 4. So, for example, $\mathbf{y}_t = [y_{t1}; y_{t2}; \ldots; y_{tn}]$ where y_{ti} denotes the prediction for the ith datapoint; and each y_{ti} is a vector of c prediction scores where c is the number of classes in the prediction problem.

- $P_t^{(y)^-}$ and $P_{t-1}^{(y)}$ are the uncertainties related to $\hat{\mathbf{y}}_t^-$ and $\hat{\mathbf{y}}_{t-1}$ respectively.

Eq. (6) is derived directly from Eq. (1) by setting A_k to the identity matrix, and assuming that $\mathbf{u}_k$ is non-existent (there is no control process involved in

KFHE). H_t in Eq. (4), (3), and (5) is set to the identity function. Also, it is assumed that no process noise is induced, hence Q_t in Eq. (2) is set to 0 to get Eq. (7). The superscript (y) throughout indicates that these parameters are related to the model Kalman filter, *kf-m*, estimating the state $\mathbf{y}_t$.

The *kf-m* measurement step is more interesting. At every tth iteration a new classification model h_t is trained with a weighted sample of the training dataset. The sampling is done with replacement, with the same number of datapoints as in the original training dataset. The weights are designed to highlight the points which were misclassified previously, as is common in boosting algorithms (although the weight updates are performed using the other Kalman filter *kf-w*). The measurement is taken as the average of the previous prediction, $\hat{\mathbf{y}}_{t-1}$, and the prediction of this tth model, h_t, as in Eq. (8). This effectively attempts to capture how much the trained model of the present iteration impacts the ensemble predictions until iteration $t-1$. Therefore the measurement step and its related error for *kf-m* becomes:

$$\mathbf{z}_t^{(y)} = \frac{1}{2}(\hat{\mathbf{y}}_{t-1} + h_t(\mathcal{D})) \tag{8}$$

$$R_t^{(y)} = \frac{1}{n}\sum_i^n (\mathcal{D}_{Y_i} \neq class(z_{ti}^{(y)})) \tag{9}$$

$$\hat{\mathbf{y}}_t = \hat{\mathbf{y}}_t^- + K_t^{(y)}(\mathbf{z}_t^{(y)} - \hat{\mathbf{y}}_t^-) \tag{10}$$

$$K_t^{(y)} = P_t^{(y)^-}(P_t^{(y)^-} + R_t^{(y)})^{-1} \tag{11}$$

$$P_t^{(y)} = (I - K_t^{(y)})P_t^{(y)^-} \tag{12}$$

where:

- $h_t = \mathcal{L}(\mathcal{D}, \hat{\mathbf{w}}_{t-1})$ is a model trained on dataset $\mathcal{D}$, using the learning algorithm $\mathcal{L}$, where the dataset is sampled using the weights $\hat{\mathbf{w}}_{t-1}$.
- $h_t(\mathcal{D})$ indicates the predictions made by the trained model h_t for the datapoints in the dataset $\mathcal{D}$.
- $\mathbf{z}_t^{(y)} = [z_{t1}; z_{t2}; \ldots; z_{tn}]$ represents the measurement heuristic, the representation of which is as explained in Figure 4.
- $R_t^{(y)}$ is the uncertainty related to $\mathbf{z}_t^{(y)}$ and is a misclassification rate calculated by comparing the class predictions made by the current ensemble, $class(z_{ti}^{(y)})$, with the ground truth classes, $\mathcal{D}_Y$.

The remaining steps of the Kalman filter process to compute the Kalman gain, the *a posteriori* state estimate, and the variance are as described for the standard Kalman filter framework but are repeated in Eq. (11), (10) and (12) for completeness. Note that, the uncertainty and the Kalman gain are scalars in the KFHE implementation, as the state to be estimated is one model and only one measurement is taken per iteration.

To initialise the *kf-m* processthe initial learner is trained as $h_0 = \mathcal{L}(\mathcal{D}, \hat{\mathbf{w}}_0)$ and $\hat{\mathbf{y}}_0 = h_0(\mathcal{D})$, where $\hat{\mathbf{w}}_0$ is a uniform distribution. Also, $P_0^{(y)}$ is set to 1, indicating that the initial *a priori* estimates are uncertain. After initialisation, the iteration starts at $t = 1$. The goal of the training phase is to compute and store the learned h_t models and the Kalman gain $K_t^{(y)}$ values for all t.

To avoid measurements with large errors, if the measurement error is more than $(1 - \frac{1}{c})$, where c is the number of classes, then the sampling weights, $\mathbf{w}_t$, are reset to a uniform distribution, which is a similar modification to that used in the AdaBoost implementation in [1].

4.3. The weight Kalman filter: kf-w

The previous description mentioned how a component learner h_t depends on a vector of sampling weights, $\hat{\mathbf{w}}_{t-1}$, which is estimated using *kf-w*. The purpose of $\hat{\mathbf{w}}_{t-1}$ is to give more weight to the datapoints which were not classified correctly in the previous iteration to encourage specialisation. The implementation of *kf-w* is very similar to the previous Kalman filter implementation. In this case the state estimated by the Kalman filter is a vector of real numbers representing weights. The time update step in this case is also the identity function:

$$\hat{\mathbf{w}}_t^{-} = \hat{\mathbf{w}}_{t-1} \tag{13}$$

$$P_t^{(w)^{-}} = P_{t-1}^{(w)} \tag{14}$$

To estimate the measurement of the weights the following equations are used:

$$\boldsymbol{z}_t^{(w)} = [z_{ti}^{(w)} | z_{ti}^{(w)} = \hat{w}_{ti} \times f(\mathcal{D}_{Y_i} \neq class(z_{ti}^{(y)}))\ \ 1 \leq i \leq n] \tag{15}$$

$$R_t^{(w)} = R_t^{(y)} \tag{16}$$

This heuristic derives the measurement $\mathbf{z}_t^{(w)}$ of *kf-w*, from the classification error, $R_t^{(y)}$, of the measurement $\mathbf{z}_t^{(y)}$ of *kf-m*, as shown in Figure 3. In Eq. (15), the function f can adjust the impact of misclassified datapoints on the weight vector. In the present work on KFHE, two options are explored: $f(x) = x$ and $f(x) = exp(x)$, where the second option places more emphasis on misclassified datapoints. We refer to the variant of KFHE using the first, linear definition for $f(x)$ as KFHE-l and the variant using the second, exponential definition as KFHE-e.

A trivial heuristic is used in this step to compute the measurement error, $R_t^{(w)}$, by setting it to $R_t^{(y)}$ (Eq. (16)). This assumes the measurement weight, $\mathbf{z}_t^{(w)}$, has an error at most equal to the last measurement error for *kf-m*, which assumes that the weights $\hat{\mathbf{w}}_t^{-}$ will lead to a model with an error no more than the last measurement by *kf-m*. The measurement update of *kf-w* becomes:

$$\hat{\mathbf{w}}_t = \hat{\mathbf{w}}_t^{-} + K_t^{(w)}(\mathbf{z}_t^{(w)} - \hat{\mathbf{w}}_t^{-}) \tag{17}$$

$$K_t^{(w)} = P_t^{(w)\text{-}}(P_t^{(w)\text{-}} + R_t^{(w)})^{-1} \tag{18}$$

$$P_t^{(w)} = (I - K_t^{(w)})P_t^{(w)\text{-}} \tag{19}$$

The superscript (w) indicates that these parameters are related to *kf-w*. Here $\mathbf{w}_t = [w_{t1}, w_{t2,\ldots,}w_{tn}]$ and $\boldsymbol{z}_t^{(w)} = [z_{t1}, z_{t2,\ldots,}z_{tn}]$ are vectors, with w_{ti} and z_{ti} representing the weight estimate and the weight measurement of the tth iteration for the ith datapoint.

The equations for *kf-w* to compute the Kalman gain, $K_t^{(w)}$; the *a posteriori* state estimate for the weights, $\hat{\mathbf{w}}_t$; and the variance, $P_t^{(w)}$, are shown in Eq. (18), (17) and (19). These are identical to those presented for *kf-m* in Section 4.2 (except for the superscripts), but are included here for completeness.

Initially, $\mathbf{w}_0$ is set to have equal weights for every datapoint in the training set, and $P_0^{(w)}$ is initialised to 1. Note that under this implementation the calculation of the measurement error for *kf-w* and the initialisation of $P_0^{(w)}$, makes the Kalman gain $K_t^{(w)}$ the same as $K_t^{(y)}$. No information from the *kf-w* process needs to be stored to support predictions from the ensmble.

4.4. Making predictions using KFHE

The goal of KFHE training is to calculate and store the trained base model, h_t, and Kalman gain, $K_t^{(y)}$, for each iteration, t, of the model Kalman filter, *kf-m*, process for $t = 0$ to $t = T$ (the total number of component classifiers trained). Once this is done, generating predictions is straight-forward. Given a new datapoint, $\mathbf{d}$, $\hat{\boldsymbol{y}}_0$ is found using the initial model h_0. Then Eq. (8) and (10) are iteratively applied to generate predictions from each model, h_t, which are combined using the appropriate Kalman gain values, $K_t^{(y)}$. The final $\hat{\boldsymbol{y}}_T$ value is taken as the ensemble prediction, and is a vector containing a prediction score for each class. Datapoints as classified as belonging to the class with the maximum score. Algorithm 2 summarises the prediction process for KFHE.

5. Experiments

This section describes the datasets, algorithms, experimental setup, and evaluation processes used in a set of experiments designed to evaluate the effectiveness of the KFHE algorithm. Two variants of KFHE, KFHE-e and KFHE-l (as described in Section 4.3), are evaluated and a set of state-of-the-art ensemble methods are used as benchmarks.

5.1. Datasets & performance measure

30 multi-class datasets (described in Table 1) from the UCI Machine Learning repository [24] are used. These datasets are frequently used in classifier benchmark experiments [9, 43, 6, 37], cover diverse domains, have numbers of classes ranging from 2 to 15, and exhibit varying amounts of class imbalance.

To evaluate the performance of each model, the macro-averaged F_1-score [23] was used. The F_1-score in a binary classifier context indicates how precise as well as how robust a classifier model is, and it can be easily extended to a

Algorithm 1 KFHE training

procedure KFHE_TRAIN($\mathcal{D}_{train}, T$) ▷
$\mathcal{D}_{train}$: training dataset,
T: Max iterations
  Initialise *kf-m*: h_0, $\hat{\mathbf{y}}_0$ and $P_0^{(y)}$ following Section 4.2
  Initialise *kf-w*: $\hat{\mathbf{w}}_0$ and $P_0^{(d)}$ following Section 4.3
  $t = 1$
  while $t \leq T$ **do**
    ▷ *kf-m* Section
    ***kf-m* time update**: Find $\hat{\mathbf{y}}^{-}$, Eq. (6) and related uncertainty $P_t^{(y)^{-}}$, Eq. (7)
    ***kf-m* measurement**: Train $h_t = \mathcal{L}(\mathcal{D}, \hat{\mathbf{w}}_{t-1})$, compute $\mathbf{z}_t^{(y)}$ and $R_t^{(y)}$, Eq. (8) and (9)
    if (misclassification rate of h_t more than $(1 - \frac{1}{c})$) **then**
      Reset $\hat{\mathbf{w}}_{t-1}$ and $P_{t-1}^{(w)}$ to initial values
      $t = t - 1$
      continue ▷ Repeat measurement step
    end if
    ***kf-m* measurement update**: Compute $\hat{\mathbf{y}}_t$, $K_t^{(y)}$ and $P_t^{(y)}$, using Eq. (10), (11) and (12)
    ▷ *kf-w* Section

    ***kf-w* time update**: Find $\hat{\mathbf{w}}^{-}$ and related uncertainty $P_t^{(w)^{-}}$ using Eq. (13) and (14)
    ***kf-w* measurement**: Compute $\mathbf{z}_t^{(w)}$ and $R_t^{(w)}$ using Eq. (15) and (16)
    ***kf-w* measurement update**: Compute $\hat{\mathbf{w}}_t$, $K_t^{(w)}$ and $P_t^{(w)}$ using Eq. (17), (18) and (19)

    $t = t + 1$
  end while
  return($\{h_t; \forall t\}, \{K_t^{(y)}; \forall t\}$) ▷ The learned classifier models, and the *kf-m* Kalman gain values
end procedure

multi-class scenario. The macro-averaged F_1-score will be denoted as $F_1^{(macro)}$, and is defined as:

$$F_1^{(macro)} = \frac{1}{c}\sum_{i=1}^{c} F_1^{(i)} = \frac{1}{c}\sum_{i=1}^{c} 2 \times \frac{precision^{(i)} \times recall^{(i)}}{precision^{(i)} + recall^{(i)}}$$

where $precision^{(i)}$ and $recall^{(i)}$ are the precision and recall values for the ith class, where c is the number of classes. $F_1^{(macro)}$ is appropriate for this experiment because the datasets used exhibit different levels of class imbalance.

Algorithm 2 KFHE prediction

procedure KFHE_PREDICTION($\mathbf{d}, \{h_t; \forall t\}, \{K_t^{(y)}; \forall t\}, T$) ▷
$\mathbf{d}$: test datapoint,
h_t: The tth sub-classifier,
$K_t^{(y)}$: The tth Kalman gain,
T: max training iterations $0 \leq t \leq T$
  $\hat{\mathbf{y}}_0 = h_0(\mathbf{d})$ ▷ Initial *a posteriori* estimate
  $t = 1$
  while $t \leq T$ **do**
    Compute $\hat{\mathbf{y}}_t^-$, the time update for *kf-m* using Eq. (6)
    Compute $\mathbf{z}_t^{(y)}$, the measurement for *kf-m*, using the h_t using Eq. (8)
    Compute the *a posteriori* estimate $\hat{\mathbf{y}}_t$ using Eq. (10)
    $t = t + 1$
  end while
  return $(\hat{\mathbf{y}}_T)$ ▷ Return class-wise prediction scores
end procedure

5.2. Experimental setup

The state-of-the-art methods used as benchmarks are AdaBoost [45], Bagging [4], Gradient Boosting Machine (GBM) [16] and Stochastic Gradient Boosting Machine (S-GBM) [17]. This set covers the different fundamental ensemble classifier types described in Section 2. For all algorithms, including KFHE-e and KFHE-l, the component learners are CART models [23]. The performance of a single CART model is also included as a baseline to compare against the ensemble methods. The number of ensemble components is set to 100 for all algorithms (initial experiments showed that for all datasets there were no significant improvements in performance beyond 100 components).

All implementations and evaluations were performed in R[2]. The AdaBoost and Bagging implementations were from the package adabag [1], and the GBM and S-GBM implementations were from the package gbm [41]. As multi-class datasets were used in this experiment, the multi-class variant of AdaBoost, AdaBoost.SAMME [45], was used for this experiment (this will be described just as AdaBoost in the remainder of the paper). For the KFHE experiments the training was stopped when the value of $K_t^{(y)}$ reached 0, which can be interpreted as an indication that the state estimated by *kf-m* has no uncertainty.

The experiments were divided into two parts. First, to evaluate the effectiveness of KFHE-e and KFHE-l and to compare these to the state-of-the-art methods, the performance of all algorithms is assessed using the datasets listed in Table 1. Second, the robustness of the different algorithms to class-label noise is compared. For both sets of experiments, for each algorithm-dataset pair, a

[2] A version of KFHE is available at https://github.com/phoxis/kfhe

Table 1: The datasets used in this paper

dataset names	#datapoints	#dimensions	#classes
mushroom	8124	22	2
iris	150	5	3
glass	214	10	6
car_eval	1728	7	4
cmc	1473	10	3
tvowel	871	4	6
balance_scale	625	5	3
breasttissue	106	10	6
german	1000	21	2
ilpd	579	11	2
ionosphere	351	34	2
knowledge	403	6	4
vertebral	310	7	2
sonar	208	61	2
diabetes	145	4	3
skulls	150	5	5
physio	464	37	3
flags	194	30	8
bupa	345	7	2
cleveland	303	14	5
haberman	306	4	2
hayes-roth	132	6	3
monks	432	7	2
newthyroid	432	7	3
yeast	1484	9	10
spam	4601	58	2
lymphography	148	19	4
movement_libras	360	91	15
SAheart	462	10	2
zoo	101	17	7

20 times 4-fold cross-validation experiment was performed, and the mean of the $F_1^{(macro)}$-scores across the folds are measured.

For the second set of experiments, class-label noise was introduced synthetically into each of the datasets in Table 1. To induce noise a fraction of the datapoints from the training set was sampled randomly and the class of each selected datapoint was randomly changed, following a uniform distribution, to a different one. For each dataset in Table 1, datasets with 5%, 10%, 15% and 20% noise were generated. For each of these noisy datasets, a 20 times 4-fold cross-validation experiment was performed. For each fold, the noisy class labels were used in training, but the $F_1^{(macro)}$-scores were computed with respect to the original unchanged dataset labels.

Table 2: Average ranks over all datasets for different levels of class label noise summarised from Tables A.4 to A.8 in AppendixA. Lower ranks are better, best ranks are highlighted in boldface. Percent values in the rows indicate class-label noise.

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
0%	**2.78**	3.33	2.98	3.70	4.30	4.82	6.08
5%	**3.07**	**3.07**	3.77	3.27	4.08	4.62	6.13
10%	3.70	**2.70**	4.77	3.37	3.50	4.10	5.87
15%	3.87	**2.68**	4.83	3.48	3.27	3.75	6.12
20%	4.33	**2.70**	5.40	3.37	3.18	3.10	5.92

6. Results

The experiment results comparing the performance of KFHE-e and KFHE-l to the other methods are shown first. Next, the results of the experiments comparing the performance of the different methods in the presence of noisy class-labels are presented. Statistical significance tests that analyse the differences between the proposed and other methods are also presented.

6.1. Performance comparison of the methods

The relative performance of each algorithm, based on the average $F_1^{(macro)}$-scores achieved in the cross validation experiments, on each of the 30 datasets was ranked (from 1 to 7, where 1 implies best performance). The first row of Table 2 (labelled 0%) shows the average rank of each algorithm across the datasets (detailed performance results for each algorithm on each dataset are shown in Table A.4 in AppendixA). These average ranks are also visualised in the first column of Figure 5 (also labelled 0%).

The average ranking shows that KFHE-e was able to attain the best average rank 2.78, AdaBoost was very close with average rank 2.98, followed by KFHE-l with the average rank 3.33. It is clear that KFHE-e outperformed GBM, S-GBM, Bagging and CART. Also, KFHE-l performs better overall than GBM, S-GBM, Bagging and CART. It was concluded that KFHE-l performed slightly less well than KHFE-e and AdaBoost due to the lack of emphasis on misclassified points in the weight measurement step. In Section 6.3, a statistical significance test will be performed to uncover significant differences between methods on datasets with non-noisy class-labels.

The evolution of the key parameters of KFHE (the measurement error, $R_t^{(y)}$; the posteriori variance, $P_t^{(y)}$; the Kalman gain, $K_t^{(y)}$; and the training misclassification rate of the *kf-m* component) with respect to the number of ensemble iterations t, for a selection of datasets (*knowledge*, *diabetes*, *car_eval* and *lymphography*) are plotted in Figure 6. The plots show the results of the first 100 iterations after which, for most of the datasets, the error reduces to 0. The plots for all of the datasets are given in AppendixD.

The plots in Figure 6 show that in all cases the value of $P_t^{(y)}$ decreases monotonically, which can be interpreted as the system becoming more confident on

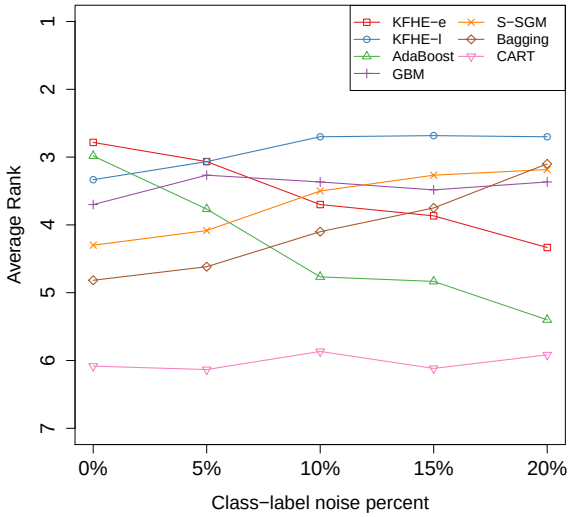
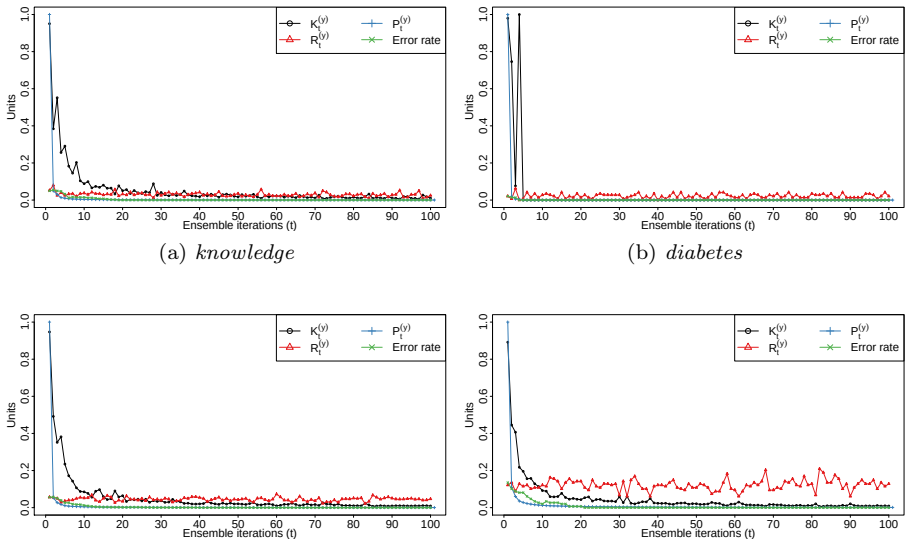


Figure 5: Changes in average rank with noisy class-labels, over the datasets used (y-axis is inverted to highlight lower average ranks are better).

the *a posteriori* estimate, and therefore that the values of $K_t^{(y)}$ reduce and stabilise, implying less impact for subsequent measurements. This is because of the way the *time step* update was formulated in Section 4.2: no uncertainty induced during the time update step, and no process noise is assumed. Therefore, in effect the steepness of $P_t^{(y)}$ controls how much of the measurement is combined through Eq. (10) and (11). Also, it is interesting to note the similarity and the rate of change of the error rate of the ensemble with the $P_t^{(y)}$ value. For most of the datasets they show a similar trend. The value of $K_t^{(y)}$ indicates the fraction of the measurement which will be incorporated into the ensemble. A measurement with less error is incorporated more into the final model.

6.2. Performance for the noisy class-label case

The relative performance of each algorithm, based on the average $F_1^{(macro)}$-scores achieved in the cross validation experiments, on each of the 30 datasets was ranked (from 1 to 7, where 1 implies best performance). this was performed separately for datasets with 5%, 10%, 15% and 20% induced class label noise. Table 2 shows the average rank of each algorithm for each level of noise (detailed performance results for each algorithm on each dataset are shown in Tables A.5 to A.8 in AppendixA). These average ranks are also visualised in Figure 5. For ease of reading, the vertical axis in Figure 5 is inverted to highlight that the lower average ranks are better.



(a) *knowledge* (b) *diabetes*

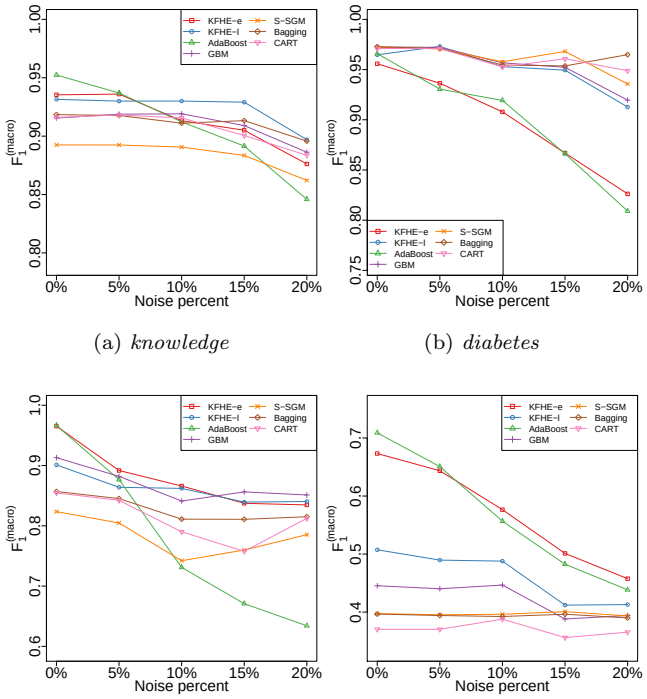
(c) *car_ eval* (d) *lymphography*

Figure 6: Changes in the parameters and the misclassification rate for the training sets for KFHE-e, for the *knowledge*, *diabetes*, *car_ eval*, and *lymphography* datasets.

Out of the algorithms tested the KFHE-l algorithm performs most consistently in the presence of class-label noise. At the 5% noise level KFHE-e and KFHE-l had the same rank, and as the class-noise level increases to 10%, 15% and 20%, KFHE-l attains the best average rank over the datasets. Along with KFHE-l, S-GBM and Bagging also improve their relative ranking. As the fraction of mislabelled datapoints increased in the training set, the average rank of AdaBoost degrades sharply. The performance of AdaBoost and Bagging in the presence of noisy class labels is studied in [9], where a similar result was found. On the other hand the change in the relative rank for GBM, and CART was consistently stable.

It should be noted that the degradation of performance with respect to class-noise in AdaBoost is more severe than KFHE-e, although both of them use the *exp* function to highlight the weights of the misclassified datapoints. This is due to the smoothing effect in the KFHE algorithm, which makes KFHE-e less sensitive to noise than AdaBoost. On the other hand, KFHE-l does not use *exp* in Eq. (15) for the weight measurement step, which makes it more robust to noise and allows it achieve high performance across all noise levels.

Figure 7 shows the change in $F_1^{(macro)}$-score for each algorithm on the *knowledge*, *diabetes*, *car_ eval*, and *lymphography* datasets (similar plots for all datasets are given in AppendixC), as the amount of class-label noise increases (note that to highlight changes in performance the vertical axes in these charts are scaled to narrow ranges of possible $F_1^{(macro)}$-scores). These plots are derived



(a) *knowledge* (b) *diabetes*

(c) *car_ eval* (d) *lymphography*

Figure 7: Changes in $F_1^{(macro)}$-score with increased induced class-label noise for the *knowledge*, *diabetes*, *car_ eval*, and *lymphography* datasets.

from Tables A.4-A.8. With few exceptions the performances of KFHE-l, GBM, S-GBM and Bagging are not impacted as much as the other approaches by noise. Although KFHE-e is generally better than the other approaches when there is no class-label noise present, as the induced noise increases, the $F_1^{(macro)}$-score for KFHE-e decreases—albeit less severely than in the case of AdaBoost.

6.3. Statistical significance testing

This section presents two types of statistical significance tests that compare the performance of the different algorithms tested. First, to assess the overall differences in performance a multiple classifier comparison test was performed following the recommendations of [18]. Second, a comparison of each pair of algorithms in isolation is performed using the Wilcoxon's Signed Rank Sum test [18].

6.3.1. Multiple classifier comparison

To understand the overall effectiveness of the variants of KFHE (KFHE-e and KFHE-l), following the recommendations of García et. al. [18], a multi-

ple classifier comparison significance test was performed (separate tests were performed on the performance of algorithms at each noise level). First, a Friedman Aligned Rank test was performed. This indicated that, at all noise levels, the null hypothesis that the performance of all algorithms is similar can be rejected, with $p < 0.001$. To further investigate these differences, post-hoc pairwise Friedman Aligned Rank tests along with the Finner p-value adjustment [18] were performed. Rank plots describing the results of the post-hoc tests (with a significance level of $\alpha = 0.05$) are shown in Figure 8.

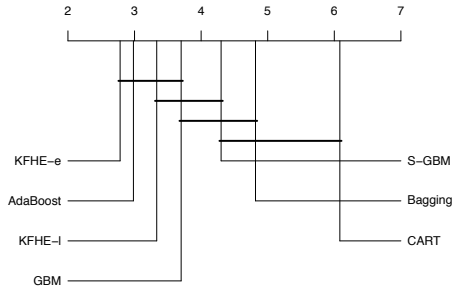
When no class-label noise is present, the results indicate that KFHE-e (avg. rank 2.78) was significantly better than S-GBM (avg. rank 4.3), Bagging (avg. rank 4.82) and CART (avg. rank 6.08) with $\alpha = 0.01$; and that KFHE-l (avg. rank 3.33) was significantly better than S-GBM, Bagging and CART with $\alpha = 0.01$. Although KFHE-e attained a better average rank, 2.78, than AdaBoost, 2.98, the null-hypothesis could not be rejected, and so it cannot be determined that the performances of KFHE-e and AdaBoost are significantly different. Similarly, KFHE-l attained a worse average rank, 3.33, than AdaBoost, but tests did not identify a statistically significant difference.

The results of the experiment for the datasets with class-label noise indicate that, as the noise continues to increase, the relative performance of KFHE-l improves, but the relative performance of KFHE-e decreases. This is as expected, because of the chosen weight measurement heuristic for the two variants of KFHE as explained in Section 4.3. KFHE-l was found to be statistically significantly better than S-GBM, and Bagging at all noise levels except 20%. KFHE-l was also found to be statistically significantly better than AdaBoost at the 10%, 15% and 20% noise levels. Although the performance of KFHE-e decreases with increasing class-label noise, it does not decrease as sharply as AdaBoost. The complete details of the tests are given in Table B.9 in AppendixB.

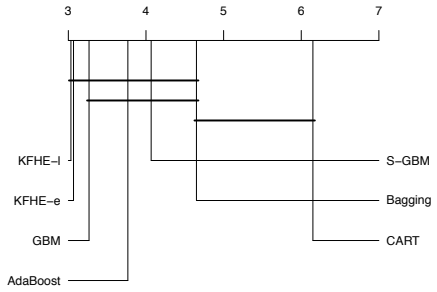
Overall these tests confirm that when no class-label noise is present KFHE-e performs as well as AdaBoost and GBM, but significantly better than S-GBM, Bagging and CART. KFHE-e, however, is not as robust to class label noise as the other approaches. KFHE-l, on the other hand, is robust to noise and performs very well in all class-label noise settings.

6.3.2. Isolated algorithm pairs comparison

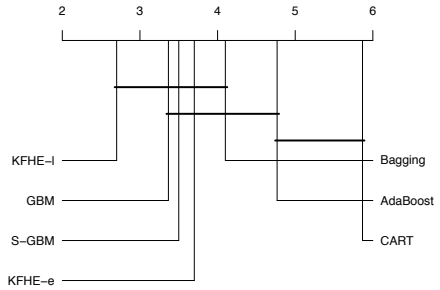
To further understand how individual algorithm pairs compare with each other, ignoring other algorithms, a two tailed Wilcoxon's Signed Rank Sum test for each pair of algorithms was performed. It must be emphasised that the Wilcoxon's rank sum test *cannot* be used to perform multiple classifier comparison without introducing Type I error (rejecting the null hypothesis when it cannot be rejected), as it does not control the Family Wise Error Rate (FWER) [18]. Therefore, the p-values for each pair from this experiment should *only* be interpreted in isolation from any other algorithms. Table 3 shows the results of these tests for the datasets without any class-label noise (Tables B.10a to B.10e in AppendixB show the results for the noisy cases). The cells in the lower diagonal show the p-values of the Wilcoxon's Signed Rank Sum test for corresponding algorithm pair and the cells in the upper diagonal show the pairwise



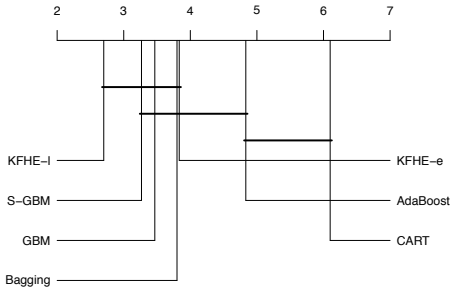
(a) Rank chart for no induced class-label noise



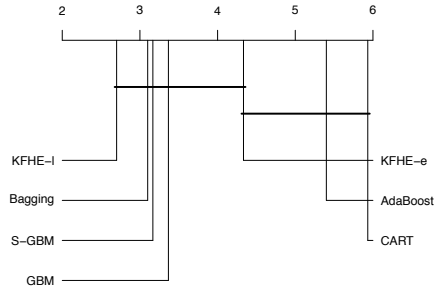
(b) Rank chart for 5% induced class-label noise



(c) Rank chart for 10% induced class-label noise



(d) Rank chart for 15% induced class-label noise



(e) Rank chart for 20% induced class-label noise

Figure 8: Rank plots from post-hoc Friedman Aligned Rank test with the Finner p-value adjustment, using a significance level of 0.05. Algorithms connected with the horizontal bars in the sub-plots are similar based on this test.

win/lost/tie counts.

The results in Table 3 show that without class-label noise when compared in isolation KFHE-e performs significantly better than any other method, except AdaBoost. In the noise free case KFHE-l performs significantly better than S-GBM, Bagging and CART. Similarly, the test results at different noise levels (described in AppendixB) show that as class-label noise increases, the performance of KFHE-e starts to become significantly better than AdaBoost,

Table 3: Result of pairwise Wilcoxon's Signed Rank Sum test over the different datasets when no class-label noise is present. Upper diagonal: win/lose/tie. Lower diagonal: Wilcoxon's Signed Rank Sum Test p-values. * $\alpha = 0.1$, ** $\alpha = 0.05$ and *** $\alpha = 0.01$.

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
KFHE-e		(19/11/0)	(13/16/1)	(23/7/0)	(21/9/0)	(24/6/0)	(26/4/0)
KFHE-l	0.009519 ***		(12/18/0)	(16/14/0)	(20/10/0)	(25/5/0)	(26/4/0)
AdaBoost	0.491795	0.028548 **		(19/11/0)	(20/10/0)	(22/8/0)	(25/5/0)
GBM	0.003018 ***	0.144739	0.013515 **		(22/8/0)	(20/10/0)	(25/5/0)
S-GBM	0.001128 ***	0.004921 ***	0.002834 ***	0.003418 ***		(19/11/0)	(25/5/0)
Bagging	0.000210 ***	0.000034 ***	0.000415 ***	0.004108 ***	0.336640		(25/4/1)
CART	0.000010 ***	0.000006 ***	0.000019 ***	0.000055 ***	0.002057 ***	0.000016 ***	

although it is worse than other methods. When compared in isolation KFHE-l performs significantly better than almost all other methods at all noise levels.

7. Conclusion and future work

This paper introduces a new perspective on training multi-class ensemble classification models. The ensemble classifier model is viewed as a state to be estimated, and this state is estimated using a Kalman filter. Unlike more common applications of Kalman filters to time series data, this work exploits the sensor fusion property of the Kalman filter to combine multiple individual multi-class classifiers to build a multi-class ensemble classifier algorithm. Based on this new perspective a new multi-class ensemble classification algorithm, the Kalman Filter-based Heuristic Ensemble (KFHE), is proposed.

Detailed experiments on two slight variants of KFHE, KFHE-e and KFHE-l, were performed. KFHE-e is more effective on non-noisy class-labels, as it emphasises the misclassified training datapoints from one iteration of the training algorithm to the next, and KFHE-l is more effective on noisy class-labels as it does not emphasise misclassified training datapoints as much. Experiments show that KFHE-e and KFHE-l perform at least as well as, and in many cases, better than Bagging, S-GBM, GBM and AdaBoost. For datasets with noisy class labels, KFHE-l performed significantly better than all other methods across different levels of class-label noise. For these datasets KFHE-e performed more poorly than KFHE-l, GBM, and S-GBM, but better than AdaBoost.

KFHE can be seen as a hybrid ensemble approach mixing the benefits of both bagging and a boosting. Bagging weighs each of the component learner's votes equally, whereas boosting finds the optimum weights, using which the component learners are combined. KFHE does not find the optimum weights analytically as AdaBoost does, but attempts to combine the classifiers based on how well the measurement is in a given iteration.

Given the new perspective, other implementations that expand upon KFHE can also be designed following the framework and methods described in Sections 3 and 4. In future, it would be interesting to pursue the following studies:

- The effect when process noise and a linear time update step are introduced.

- Multiple and different types of measurements can also be performed. That is, instead of having one component classifier model per iteration, more than one classifier model could be used. This is analogous to having multiple noisy sensors measuring the DC voltage level of the toy example presented in Section 3.1.

- To further study the effects of other types of noise (class-wise label noise, noise in input space, etc.), higher levels of noise induced in the class-label assignments, and performance on imbalanced class datasets.

Acknowledgements

This research was supported by Science Foundation Ireland (SFI) under Grant number SFI/12/RC/2289. The authors would like to thank Gevorg Poghosyan, PhD Research Student at Insight Centre for Data Analytics, School of Computer Science, University College Dublin, for feedback and discussions which led to the state space representation in Figure 2. Also, the authors would like to thank the unnamed reviewers for their detailed and constructive comments which helped to significantly improve the quality of the paper.

AppendixA. Complete experiment results

Table A.4: Each cell in the table shows the mean and standard deviation of the $F_1^{(macro)}$-score (higher value is better) for the 20 times 4-fold cross-validation experiment for each algorithm and each of the datasets listed in Table 1. The values in parenthesis are the relative rankings of the algorithms on the dataset in the corresponding row (lower ranks are better).

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
mushroom	1.0000 ± 0.00 (1.5)	0.9968 ± 0.00 (5)	1.0000 ± 0.00 (1.5)	0.9997 ± 0.00 (3)	0.9990 ± 0.00 (4)	0.9941 ± 0.00 (6.5)	0.9941 ± 0.00 (6.5)
iris	0.9433 ± 0.03 (4)	0.9487 ± 0.03 (1)	0.9448 ± 0.03 (2)	0.9403 ± 0.04 (5)	0.9437 ± 0.03 (3)	0.9376 ± 0.03 (6)	0.9298 ± 0.03 (7)
glass	0.7125 ± 0.08 (3)	0.7153 ± 0.07 (1)	0.7144 ± 0.07 (2)	0.6666 ± 0.08 (4)	0.5695 ± 0.09 (6)	0.5965 ± 0.09 (5)	0.5466 ± 0.06 (7)
car_eval	0.9653 ± 0.02 (2)	0.9011 ± 0.03 (4)	0.9665 ± 0.02 (1)	0.9131 ± 0.04 (3)	0.8236 ± 0.04 (7)	0.8569 ± 0.04 (5)	0.8546 ± 0.04 (6)
cmc	0.5222 ± 0.02 (5)	0.5280 ± 0.03 (2)	0.5038 ± 0.02 (7)	0.5270 ± 0.02 (4)	0.5275 ± 0.02 (3)	0.5291 ± 0.03 (1)	0.5187 ± 0.03 (6)
tvowel	0.8451 ± 0.02 (2)	0.8283 ± 0.03 (3)	0.8279 ± 0.02 (4)	0.8458 ± 0.03 (1)	0.8236 ± 0.03 (5)	0.8004 ± 0.03 (6)	0.7855 ± 0.03 (7)
balance_scale	0.6345 ± 0.03 (1)	0.5984 ± 0.01 (4)	0.6186 ± 0.03 (2)	0.5935 ± 0.02 (5)	0.6045 ± 0.01 (3)	0.5861 ± 0.02 (6)	0.5412 ± 0.02 (7)
flags	0.3059 ± 0.05 (3)	0.2771 ± 0.05 (4)	0.3187 ± 0.06 (2)	0.3236 ± 0.06 (1)	0.2602 ± 0.03 (5)	0.2525 ± 0.03 (6)	0.2439 ± 0.03 (7)
german	0.6907 ± 0.03 (2)	0.6960 ± 0.02 (1)	0.6837 ± 0.03 (5)	0.6860 ± 0.03 (3)	0.6852 ± 0.03 (4)	0.6826 ± 0.02 (6)	0.6550 ± 0.03 (7)
ilpd	0.6126 ± 0.04 (2)	0.5797 ± 0.04 (5)	0.6153 ± 0.04 (1)	0.5809 ± 0.04 (4)	0.5733 ± 0.04 (6)	0.5675 ± 0.04 (7)	0.5865 ± 0.04 (3)
ionosphere	0.9238 ± 0.03 (2)	0.9157 ± 0.03 (4)	0.9298 ± 0.02 (1)	0.9179 ± 0.03 (3)	0.9105 ± 0.03 (5)	0.9004 ± 0.03 (6)	0.8617 ± 0.03 (7)
knowledge	0.9354 ± 0.03 (2)	0.9315 ± 0.03 (3)	0.9524 ± 0.02 (1)	0.9155 ± 0.03 (6)	0.8925 ± 0.04 (7)	0.9184 ± 0.03 (4)	0.9160 ± 0.03 (5)
vertebral	0.8036 ± 0.04 (4)	0.8090 ± 0.04 (2)	0.8001 ± 0.04 (6)	0.8030 ± 0.04 (5)	0.8130 ± 0.05 (1)	0.8042 ± 0.04 (3)	0.7857 ± 0.04 (7)
sonar	0.8072 ± 0.05 (2)	0.7836 ± 0.06 (4)	0.8371 ± 0.05 (1)	0.7893 ± 0.06 (3)	0.7797 ± 0.06 (5)	0.7766 ± 0.06 (6)	0.7021 ± 0.05 (7)
skulls	0.2358 ± 0.06 (5)	0.2380 ± 0.06 (3)	0.2362 ± 0.06 (4)	0.2514 ± 0.08 (1)	0.2436 ± 0.06 (2)	0.2300 ± 0.06 (7)	0.2309 ± 0.07 (6)
diabetes	0.9558 ± 0.03 (7)	0.9647 ± 0.03 (6)	0.9658 ± 0.03 (5)	0.9725 ± 0.02 (2)	0.9722 ± 0.02 (3)	0.9727 ± 0.03 (1)	0.9710 ± 0.03 (4)
physio	0.9069 ± 0.02 (4)	0.9109 ± 0.03 (2)	0.9079 ± 0.02 (3)	0.9046 ± 0.02 (5)	0.9136 ± 0.03 (1)	0.8959 ± 0.03 (6)	0.8847 ± 0.03 (7)
breasttissue	0.6766 ± 0.08 (1)	0.6711 ± 0.08 (2)	0.6606 ± 0.08 (4)	0.6605 ± 0.07 (5)	0.6347 ± 0.08 (6)	0.6653 ± 0.08 (3)	0.6338 ± 0.08 (7)
bupa	0.7027 ± 0.04 (2)	0.7114 ± 0.04 (1)	0.6926 ± 0.04 (6)	0.7018 ± 0.04 (3)	0.6944 ± 0.04 (5)	0.6954 ± 0.04 (4)	0.6433 ± 0.05 (7)
cleveland	0.2975 ± 0.05 (2)	0.2845 ± 0.04 (5)	0.3058 ± 0.04 (1)	0.2938 ± 0.04 (3)	0.2865 ± 0.04 (4)	0.2736 ± 0.04 (7)	0.2766 ± 0.04 (6)
haberman	0.5504 ± 0.05 (6)	0.5743 ± 0.05 (5)	0.5465 ± 0.05 (7)	0.5751 ± 0.05 (4)	0.5996 ± 0.04 (1)	0.5757 ± 0.05 (3)	0.5772 ± 0.05 (2)
hayes_roth	0.8602 ± 0.05 (1)	0.8491 ± 0.05 (3)	0.8510 ± 0.04 (2)	0.6094 ± 0.08 (6)	0.5683 ± 0.10 (7)	0.7418 ± 0.10 (4)	0.6691 ± 0.10 (5)
monks	0.9997 ± 0.00 (2)	0.9981 ± 0.01 (3)	1.0000 ± 0.00 (1)	0.9671 ± 0.06 (4)	0.9114 ± 0.06 (5)	0.9002 ± 0.06 (6)	0.8178 ± 0.09 (7)
newthyroid	0.3973 ± 0.04 (4)	0.3867 ± 0.04 (6)	0.3972 ± 0.04 (5)	0.3742 ± 0.04 (7)	0.4162 ± 0.04 (2)	0.4275 ± 0.04 (1)	0.4087 ± 0.11 (3)
yeast	0.5339 ± 0.05 (1)	0.4701 ± 0.03 (4)	0.5209 ± 0.05 (3)	0.5225 ± 0.05 (2)	0.4359 ± 0.02 (5)	0.4187 ± 0.03 (6)	0.4069 ± 0.03 (7)
spam	0.9477 ± 0.01 (2)	0.9256 ± 0.01 (4)	0.9508 ± 0.01 (1)	0.9309 ± 0.01 (3)	0.9225 ± 0.01 (5)	0.9029 ± 0.01 (6)	0.8870 ± 0.01 (7)
lymphography	0.6733 ± 0.19 (2)	0.5074 ± 0.13 (3)	0.7089 ± 0.18 (1)	0.4454 ± 0.10 (4)	0.3973 ± 0.03 (5)	0.3966 ± 0.03 (6)	0.3704 ± 0.04 (7)
movement_libras	0.7772 ± 0.04 (1)	0.7488 ± 0.05 (3)	0.7679 ± 0.04 (2)	0.6434 ± 0.05 (5)	0.6190 ± 0.05 (6)	0.6715 ± 0.05 (4)	0.5176 ± 0.05 (7)
SAheart	0.6214 ± 0.04 (6)	0.6408 ± 0.04 (4)	0.6090 ± 0.04 (7)	0.6436 ± 0.04 (3)	0.6509 ± 0.03 (1)	0.6466 ± 0.04 (2)	0.6237 ± 0.05 (5)
zoo	0.8548 ± 0.12 (2)	0.8490 ± 0.11 (3)	0.8740 ± 0.11 (1)	0.8450 ± 0.10 (4)	0.5455 ± 0.11 (7)	0.5922 ± 0.09 (5)	0.5840 ± 0.05 (6)
Average rank	**2.78**	3.33	2.98	3.7	4.3	4.82	6.08

Table A.5: Noise Level: 5% . Each cell in the table shows the $F_1^{(macro)}$-measure (higher value is better) from the 20 times 4-fold cross-validation experiment with 5% noise induced on the class labels. The values in parenthesis is the relative ranking of the algorithm on the dataset in the corresponding row (lower ranks are better).

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
mushroom	0.9972 ± 0.00 (4)	0.9941 ± 0.00 (6)	0.9993 ± 0.00 (2)	0.9997 ± 0.00 (1)	0.9990 ± 0.00 (3)	0.9941 ± 0.00 (6)	0.9941 ± 0.00 (6)
iris	0.9205 ± 0.05 (7)	0.9413 ± 0.03 (3)	0.9282 ± 0.04 (6)	0.9359 ± 0.03 (4)	0.9486 ± 0.03 (1)	0.9436 ± 0.03 (2)	0.9335 ± 0.03 (5)
glass	0.6818 ± 0.09 (2)	0.6969 ± 0.07 (1)	0.6597 ± 0.08 (3)	0.6498 ± 0.08 (4)	0.5532 ± 0.08 (6)	0.5876 ± 0.10 (5)	0.5367 ± 0.06 (7)
car_eval	0.8918 ± 0.03 (1)	0.8639 ± 0.03 (4)	0.8765 ± 0.03 (3)	0.8820 ± 0.04 (2)	0.8047 ± 0.04 (7)	0.8451 ± 0.03 (5)	0.8423 ± 0.03 (6)
cmc	0.5223 ± 0.02 (5)	0.5285 ± 0.02 (3)	0.5047 ± 0.03 (7)	0.5303 ± 0.02 (2)	0.5274 ± 0.02 (4)	0.5330 ± 0.02 (1)	0.5197 ± 0.03 (6)
tvowel	0.8346 ± 0.03 (2)	0.8275 ± 0.03 (4)	0.7903 ± 0.03 (6)	0.8435 ± 0.03 (1)	0.8329 ± 0.03 (3)	0.7961 ± 0.03 (5)	0.7805 ± 0.04 (7)
balance_scale	0.5989 ± 0.03 (1)	0.5940 ± 0.02 (3)	0.5917 ± 0.03 (4)	0.5912 ± 0.02 (5)	0.5982 ± 0.02 (2)	0.5799 ± 0.02 (6)	0.5418 ± 0.02 (7)
flags	0.3113 ± 0.06 (3)	0.2988 ± 0.05 (4)	0.3193 ± 0.06 (1)	0.3185 ± 0.06 (2)	0.2678 ± 0.04 (5)	0.2518 ± 0.03 (6)	0.2420 ± 0.04 (7)
german	0.6732 ± 0.03 (2)	0.6765 ± 0.03 (1)	0.6690 ± 0.03 (4)	0.6695 ± 0.03 (3)	0.6655 ± 0.03 (5)	0.6608 ± 0.03 (6)	0.6348 ± 0.04 (7)
ilpd	0.6130 ± 0.04 (2)	0.5874 ± 0.04 (3)	0.6220 ± 0.04 (1)	0.5815 ± 0.04 (4)	0.5755 ± 0.04 (6)	0.5724 ± 0.04 (7)	0.5759 ± 0.04 (5)
ionosphere	0.9093 ± 0.03 (2)	0.9087 ± 0.03 (4)	0.9090 ± 0.03 (3)	0.9018 ± 0.03 (6)	0.9103 ± 0.03 (1)	0.9023 ± 0.03 (5)	0.8507 ± 0.04 (7)
knowledge	0.9360 ± 0.02 (2)	0.9300 ± 0.02 (3)	0.9369 ± 0.02 (1)	0.9188 ± 0.03 (4)	0.8924 ± 0.03 (7)	0.9177 ± 0.03 (6)	0.9181 ± 0.03 (5)
vertebral	0.7928 ± 0.05 (5)	0.8073 ± 0.05 (3)	0.7740 ± 0.05 (6)	0.8024 ± 0.05 (4)	0.8163 ± 0.04 (1.5)	0.8163 ± 0.05 (1.5)	0.7736 ± 0.05 (7)
sonar	0.7900 ± 0.05 (2)	0.7759 ± 0.05 (3)	0.8116 ± 0.05 (1)	0.7719 ± 0.06 (4)	0.7708 ± 0.05 (5)	0.7610 ± 0.05 (6)	0.6867 ± 0.06 (7)
skulls	0.2462 ± 0.07 (3)	0.2226 ± 0.06 (6)	0.2275 ± 0.06 (5)	0.2550 ± 0.07 (1)	0.2510 ± 0.07 (2)	0.2295 ± 0.06 (4)	0.1935 ± 0.06 (7)
diabetes	0.9364 ± 0.04 (6)	0.9731 ± 0.03 (1)	0.9305 ± 0.04 (7)	0.9722 ± 0.02 (2)	0.9705 ± 0.02 (5)	0.9710 ± 0.03 (3)	0.9709 ± 0.03 (4)
physio	0.8781 ± 0.03 (5)	0.9092 ± 0.02 (2)	0.8712 ± 0.03 (6)	0.8995 ± 0.02 (3)	0.9113 ± 0.02 (1)	0.8955 ± 0.03 (4)	0.8658 ± 0.03 (7)
breasttissue	0.6557 ± 0.07 (2)	0.6709 ± 0.07 (1)	0.6511 ± 0.08 (3)	0.6502 ± 0.08 (4)	0.6285 ± 0.08 (6)	0.6416 ± 0.08 (5)	0.5927 ± 0.08 (7)
bupa	0.6852 ± 0.04 (2)	0.6962 ± 0.04 (1)	0.6625 ± 0.05 (6)	0.6839 ± 0.04 (4)	0.6846 ± 0.04 (3)	0.6795 ± 0.05 (5)	0.6309 ± 0.05 (7)
cleveland	0.2895 ± 0.05 (4)	0.2906 ± 0.05 (3)	0.3076 ± 0.05 (1)	0.2922 ± 0.05 (2)	0.2883 ± 0.05 (5)	0.2793 ± 0.04 (7)	0.2864 ± 0.05 (6)
haberman	0.5429 ± 0.05 (6)	0.5554 ± 0.06 (5)	0.5342 ± 0.05 (7)	0.5665 ± 0.06 (3)	0.5793 ± 0.06 (1)	0.5643 ± 0.06 (4)	0.5738 ± 0.06 (2)
hayes_roth	0.8022 ± 0.07 (2)	0.8289 ± 0.06 (1)	0.7815 ± 0.07 (3)	0.5869 ± 0.09 (6)	0.5208 ± 0.09 (7)	0.7145 ± 0.10 (4)	0.6695 ± 0.10 (5)
monks	0.9644 ± 0.02 (2)	0.9985 ± 0.00 (1)	0.9311 ± 0.03 (5)	0.9473 ± 0.06 (3)	0.9268 ± 0.06 (6)	0.9379 ± 0.06 (4)	0.8498 ± 0.09 (7)
newthyroid	0.3972 ± 0.04 (5)	0.3962 ± 0.04 (6)	0.4039 ± 0.04 (4)	0.3960 ± 0.04 (7)	0.4475 ± 0.04 (1)	0.4395 ± 0.03 (2)	0.4086 ± 0.11 (3)
yeast	0.4797 ± 0.06 (2)	0.4354 ± 0.03 (4)	0.4521 ± 0.07 (3)	0.4829 ± 0.05 (1)	0.4312 ± 0.03 (5)	0.4176 ± 0.02 (6)	0.4021 ± 0.03 (7)
spam	0.9325 ± 0.01 (1)	0.9265 ± 0.01 (4)	0.9311 ± 0.01 (2)	0.9294 ± 0.01 (3)	0.9219 ± 0.01 (5)	0.9039 ± 0.01 (6)	0.8866 ± 0.01 (7)
lymphography	0.6436 ± 0.16 (2)	0.4896 ± 0.12 (3)	0.6507 ± 0.15 (1)	0.4402 ± 0.09 (4)	0.3954 ± 0.04 (5)	0.3941 ± 0.04 (6)	0.3704 ± 0.05 (7)
movement_libras	0.7365 ± 0.04 (1)	0.7138 ± 0.05 (3)	0.7362 ± 0.04 (2)	0.6064 ± 0.06 (5)	0.5868 ± 0.06 (6)	0.6517 ± 0.06 (4)	0.5008 ± 0.05 (7)
SAheart	0.6120 ± 0.04 (5)	0.6252 ± 0.04 (4)	0.6020 ± 0.04 (7)	0.6255 ± 0.04 (3)	0.6446 ± 0.03 (1)	0.6405 ± 0.04 (2)	0.6106 ± 0.05 (6)
zoo	0.7765 ± 0.11 (4)	0.8025 ± 0.13 (2)	0.7841 ± 0.12 (3)	0.8058 ± 0.12 (1)	0.5566 ± 0.11 (7)	0.5823 ± 0.09 (5)	0.5704 ± 0.07 (6)
Average rank	**3.07**	**3.07**	3.77	3.27	4.08	4.62	6.13

Table A.6: Noise Level: 10% . Each cell in the table shows the $F_1^{(macro)}$-measure (higher value is better) from the 20 times 4-fold cross-validation experiment with 10% noise induced on the class labels. The values in parenthesis is the relative ranking of the algorithm on the dataset in the corresponding row (lower ranks are better).

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
mushroom	0.9942 ± 0.00 (4)	0.9941 ± 0.00 (6)	0.9970 ± 0.00 (3)	0.9988 ± 0.00 (1)	0.9987 ± 0.00 (2)	0.9941 ± 0.00 (6)	0.9941 ± 0.00 (6)
iris	0.8749 ± 0.06 (6)	0.9384 ± 0.04 (3)	0.8569 ± 0.06 (7)	0.9319 ± 0.05 (5)	0.9487 ± 0.03 (1)	0.9433 ± 0.03 (2)	0.9380 ± 0.03 (4)
glass	0.6850 ± 0.09 (2)	0.6990 ± 0.08 (1)	0.6801 ± 0.07 (3)	0.6253 ± 0.08 (4)	0.5658 ± 0.08 (6)	0.6189 ± 0.09 (5)	0.5641 ± 0.10 (7)
car_eval	0.8660 ± 0.04 (1)	0.8621 ± 0.03 (2)	0.7311 ± 0.04 (7)	0.8411 ± 0.05 (3)	0.7421 ± 0.05 (6)	0.8111 ± 0.04 (4)	0.7901 ± 0.04 (5)
cmc	0.5133 ± 0.02 (6)	0.5232 ± 0.02 (1)	0.4886 ± 0.02 (7)	0.5220 ± 0.02 (2)	0.5212 ± 0.02 (4)	0.5217 ± 0.02 (3)	0.5175 ± 0.03 (5)
tvowel	0.8264 ± 0.03 (2)	0.8187 ± 0.03 (4)	0.7747 ± 0.04 (7)	0.8383 ± 0.03 (1)	0.8238 ± 0.03 (3)	0.7961 ± 0.03 (5)	0.7791 ± 0.03 (6)
balance_scale	0.6016 ± 0.03 (2)	0.5939 ± 0.02 (3)	0.5929 ± 0.03 (4)	0.5912 ± 0.02 (5)	0.6024 ± 0.02 (1)	0.5757 ± 0.02 (6)	0.5332 ± 0.02 (7)
flags	0.2716 ± 0.05 (4)	0.2544 ± 0.03 (6)	0.2860 ± 0.04 (2)	0.2885 ± 0.06 (1)	0.2763 ± 0.03 (3)	0.2577 ± 0.03 (5)	0.2484 ± 0.04 (7)
german	0.6698 ± 0.03 (4)	0.6786 ± 0.03 (1)	0.6611 ± 0.03 (6)	0.6695 ± 0.03 (5)	0.6756 ± 0.03 (2)	0.6706 ± 0.03 (3)	0.6348 ± 0.04 (7)
ilpd	0.5836 ± 0.04 (2)	0.5782 ± 0.04 (3)	0.5849 ± 0.05 (1)	0.5737 ± 0.04 (4)	0.5696 ± 0.04 (5)	0.5650 ± 0.04 (7)	0.5694 ± 0.05 (6)
ionosphere	0.8922 ± 0.04 (5)	0.9098 ± 0.03 (1)	0.8867 ± 0.04 (6)	0.9048 ± 0.03 (4)	0.9095 ± 0.03 (2)	0.9093 ± 0.03 (3)	0.8486 ± 0.04 (7)
knowledge	0.9132 ± 0.03 (4)	0.9300 ± 0.02 (1)	0.9122 ± 0.03 (5)	0.9191 ± 0.03 (2)	0.8906 ± 0.03 (7)	0.9111 ± 0.02 (6)	0.9156 ± 0.03 (3)
vertebral	0.7904 ± 0.05 (5)	0.7987 ± 0.04 (3)	0.7749 ± 0.04 (7)	0.7949 ± 0.05 (4)	0.8099 ± 0.05 (1)	0.8059 ± 0.05 (2)	0.7838 ± 0.05 (6)
sonar	0.7818 ± 0.05 (2)	0.7719 ± 0.06 (3)	0.7947 ± 0.05 (1)	0.7588 ± 0.06 (5)	0.7702 ± 0.06 (4)	0.7505 ± 0.06 (6)	0.6677 ± 0.06 (7)
skulls	0.2396 ± 0.06 (4)	0.2362 ± 0.07 (5)	0.2275 ± 0.05 (6)	0.2547 ± 0.06 (3)	0.2586 ± 0.06 (1)	0.2557 ± 0.07 (2)	0.2247 ± 0.07 (7)
diabetes	0.9079 ± 0.05 (7)	0.9529 ± 0.04 (4)	0.9193 ± 0.05 (6)	0.9567 ± 0.04 (2)	0.9578 ± 0.04 (1)	0.9545 ± 0.04 (3)	0.9527 ± 0.04 (5)
physio	0.8736 ± 0.03 (6)	0.9114 ± 0.03 (1)	0.8458 ± 0.04 (7)	0.8977 ± 0.03 (4)	0.9100 ± 0.03 (2)	0.8988 ± 0.03 (3)	0.8822 ± 0.03 (5)
breasttissue	0.6136 ± 0.08 (5)	0.6489 ± 0.10 (2)	0.6150 ± 0.09 (4)	0.6721 ± 0.09 (1)	0.5737 ± 0.10 (7)	0.6470 ± 0.09 (3)	0.5890 ± 0.07 (6)
bupa	0.6737 ± 0.05 (5)	0.6894 ± 0.04 (3)	0.6670 ± 0.05 (6)	0.6823 ± 0.05 (4)	0.6901 ± 0.04 (1)	0.6898 ± 0.05 (2)	0.6241 ± 0.05 (7)
cleveland	0.2805 ± 0.05 (4)	0.2849 ± 0.05 (2)	0.2828 ± 0.05 (3)	0.2889 ± 0.06 (1)	0.2654 ± 0.05 (6)	0.2608 ± 0.04 (7)	0.2700 ± 0.04 (5)
haberman	0.5436 ± 0.06 (6)	0.5729 ± 0.06 (5)	0.5326 ± 0.05 (7)	0.5819 ± 0.05 (3)	0.5975 ± 0.05 (1)	0.5858 ± 0.06 (2)	0.5796 ± 0.07 (4)
hayes_roth	0.7349 ± 0.07 (2)	0.7971 ± 0.07 (1)	0.7291 ± 0.09 (3)	0.5641 ± 0.09 (6)	0.5330 ± 0.11 (7)	0.7036 ± 0.09 (4)	0.6459 ± 0.08 (5)
monks	0.9336 ± 0.02 (2)	0.9835 ± 0.02 (1)	0.8884 ± 0.03 (6)	0.9058 ± 0.08 (5)	0.9069 ± 0.05 (4)	0.9162 ± 0.05 (3)	0.8280 ± 0.09 (7)
newthyroid	0.3922 ± 0.04 (7)	0.4255 ± 0.04 (4)	0.4015 ± 0.04 (6)	0.4178 ± 0.04 (5)	0.4601 ± 0.04 (3)	0.4641 ± 0.04 (2)	0.4903 ± 0.08 (1)
yeast	0.5061 ± 0.05 (1)	0.4508 ± 0.03 (3)	0.4499 ± 0.05 (4)	0.4756 ± 0.05 (2)	0.4382 ± 0.03 (5)	0.4099 ± 0.03 (6)	0.3958 ± 0.03 (7)
spam	0.9215 ± 0.01 (4)	0.9251 ± 0.01 (2)	0.9128 ± 0.01 (5)	0.9279 ± 0.01 (1)	0.9231 ± 0.01 (3)	0.8993 ± 0.01 (6)	0.8848 ± 0.01 (7)
lymphography	0.5765 ± 0.15 (1)	0.4878 ± 0.12 (3)	0.5567 ± 0.12 (2)	0.4466 ± 0.10 (4)	0.3963 ± 0.04 (5)	0.3924 ± 0.03 (6)	0.3878 ± 0.05 (7)
movement_libras	0.7025 ± 0.06 (1)	0.6890 ± 0.05 (3)	0.6957 ± 0.05 (2)	0.5678 ± 0.05 (6)	0.5818 ± 0.05 (5)	0.6224 ± 0.05 (4)	0.4789 ± 0.05 (7)
SAheart	0.6259 ± 0.04 (5)	0.6410 ± 0.04 (3)	0.6137 ± 0.05 (7)	0.6359 ± 0.05 (4)	0.6503 ± 0.04 (1)	0.6436 ± 0.04 (2)	0.6141 ± 0.05 (6)
zoo	0.7423 ± 0.11 (2)	0.7612 ± 0.11 (1)	0.7238 ± 0.10 (3)	0.6636 ± 0.10 (4)	0.5182 ± 0.08 (6)	0.5183 ± 0.08 (5)	0.5104 ± 0.06 (7)
Average rank	3.70	**2.70**	4.77	3.37	3.50	4.10	5.87

Table A.7: Noise Level: 15% . Each cell in the table shows the $F_1^{(macro)}$-measure (higher value is better) from the 20 times 4-fold cross-validation experiment with 15% noise induced on the class labels. The values in parenthesis is the relative ranking of the algorithm on the dataset in the corresponding row (lower ranks are better).

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
mushroom	0.9941 ± 0.00 (4.5)	0.9941 ± 0.00 (4.5)	0.9967 ± 0.00 (3)	0.9992 ± 0.00 (1)	0.9990 ± 0.00 (2)	0.9934 ± 0.00 (6.5)	0.9934 ± 0.00 (6.5)
iris	0.8619 ± 0.06 (6)	0.9379 ± 0.04 (3)	0.8438 ± 0.06 (7)	0.9317 ± 0.05 (4)	0.9499 ± 0.03 (1)	0.9463 ± 0.04 (2)	0.9299 ± 0.03 (5)
glass	0.5976 ± 0.09 (2)	0.6240 ± 0.08 (1)	0.5901 ± 0.09 (4)	0.5909 ± 0.09 (3)	0.5076 ± 0.08 (6)	0.5345 ± 0.08 (5)	0.4721 ± 0.08 (7)
car_eval	0.8374 ± 0.04 (3)	0.8394 ± 0.04 (2)	0.6708 ± 0.04 (7)	0.8563 ± 0.04 (1)	0.7599 ± 0.05 (5)	0.8108 ± 0.05 (4)	0.7577 ± 0.07 (6)
cmc	0.5182 ± 0.03 (5)	0.5199 ± 0.02 (4)	0.4889 ± 0.03 (7)	0.5221 ± 0.02 (3)	0.5245 ± 0.03 (2)	0.5258 ± 0.03 (1)	0.4949 ± 0.04 (6)
tvowel	0.8261 ± 0.03 (3)	0.8208 ± 0.03 (4)	0.7552 ± 0.03 (7)	0.8274 ± 0.03 (2)	0.8311 ± 0.03 (1)	0.7924 ± 0.03 (5)	0.7758 ± 0.03 (6)
balance_scale	0.5908 ± 0.03 (3)	0.5948 ± 0.03 (2)	0.5808 ± 0.04 (5)	0.5871 ± 0.02 (4)	0.5949 ± 0.02 (1)	0.5674 ± 0.02 (6)	0.5326 ± 0.03 (7)
flags	0.3032 ± 0.06 (1)	0.2998 ± 0.05 (3)	0.2958 ± 0.06 (4)	0.3021 ± 0.06 (2)	0.2463 ± 0.03 (5)	0.2451 ± 0.03 (6)	0.2352 ± 0.04 (7)
german	0.6488 ± 0.03 (3)	0.6522 ± 0.03 (1)	0.6376 ± 0.03 (5)	0.6491 ± 0.03 (2)	0.6432 ± 0.03 (4)	0.6275 ± 0.03 (6)	0.6202 ± 0.04 (7)
ilpd	0.5645 ± 0.04 (3)	0.5699 ± 0.04 (1)	0.5698 ± 0.04 (2)	0.5592 ± 0.04 (4)	0.5564 ± 0.04 (5)	0.5557 ± 0.04 (6)	0.5517 ± 0.04 (7)
ionosphere	0.8572 ± 0.04 (5)	0.8892 ± 0.04 (3)	0.8416 ± 0.05 (6)	0.8714 ± 0.04 (4)	0.9025 ± 0.04 (1)	0.8995 ± 0.04 (2)	0.8043 ± 0.06 (7)
knowledge	0.9050 ± 0.03 (4)	0.9291 ± 0.03 (1)	0.8915 ± 0.03 (6)	0.9090 ± 0.03 (3)	0.8835 ± 0.03 (7)	0.9133 ± 0.03 (2)	0.9006 ± 0.04 (5)
vertebral	0.7463 ± 0.04 (5)	0.7790 ± 0.05 (3)	0.7275 ± 0.05 (6)	0.7641 ± 0.05 (4)	0.7997 ± 0.05 (1)	0.7866 ± 0.05 (2)	0.7267 ± 0.05 (7)
sonar	0.7548 ± 0.06 (2)	0.7451 ± 0.06 (5)	0.7462 ± 0.06 (4)	0.7330 ± 0.06 (6)	0.7643 ± 0.06 (1)	0.7485 ± 0.07 (3)	0.6356 ± 0.08 (7)
skulls	0.2545 ± 0.06 (3)	0.2635 ± 0.06 (1)	0.2587 ± 0.06 (2)	0.2530 ± 0.05 (4.5)	0.2408 ± 0.08 (6)	0.2530 ± 0.06 (4.5)	0.2183 ± 0.06 (7)
diabetes	0.8668 ± 0.06 (6)	0.9494 ± 0.04 (5)	0.8662 ± 0.08 (7)	0.9523 ± 0.04 (4)	0.9680 ± 0.03 (1)	0.9536 ± 0.03 (3)	0.9608 ± 0.03 (2)
physio	0.8496 ± 0.03 (6)	0.8964 ± 0.02 (3)	0.8155 ± 0.04 (7)	0.8893 ± 0.03 (4)	0.9052 ± 0.02 (1)	0.8995 ± 0.02 (2)	0.8734 ± 0.03 (5)
breasttissue	0.6072 ± 0.08 (6)	0.6283 ± 0.08 (2)	0.6202 ± 0.09 (3)	0.6139 ± 0.07 (5)	0.6161 ± 0.08 (4)	0.6500 ± 0.07 (1)	0.5652 ± 0.08 (7)
bupa	0.6553 ± 0.04 (6)	0.6779 ± 0.05 (2)	0.6565 ± 0.05 (5)	0.6640 ± 0.05 (4)	0.6828 ± 0.05 (1)	0.6776 ± 0.05 (3)	0.6089 ± 0.05 (7)
cleveland	0.2917 ± 0.05 (3.5)	0.2869 ± 0.05 (5)	0.2989 ± 0.05 (2)	0.2998 ± 0.05 (1)	0.2770 ± 0.04 (6)	0.2917 ± 0.05 (3.5)	0.2758 ± 0.05 (7)
haberman	0.5448 ± 0.05 (6)	0.5606 ± 0.06 (5)	0.5404 ± 0.05 (7)	0.5625 ± 0.06 (4)	0.5819 ± 0.07 (1)	0.5645 ± 0.06 (3)	0.5677 ± 0.07 (2)
hayes_roth	0.7335 ± 0.08 (2)	0.7653 ± 0.08 (1)	0.6615 ± 0.09 (4)	0.5390 ± 0.08 (6)	0.4691 ± 0.10 (7)	0.6970 ± 0.09 (3)	0.6558 ± 0.10 (5)
monks	0.8757 ± 0.04 (4)	0.9623 ± 0.03 (1)	0.8265 ± 0.04 (6)	0.8671 ± 0.07 (5)	0.9045 ± 0.05 (3)	0.9134 ± 0.06 (2)	0.7957 ± 0.08 (7)
newthyroid	0.3816 ± 0.04 (7)	0.4305 ± 0.04 (3)	0.3825 ± 0.04 (6)	0.4224 ± 0.04 (4)	0.4700 ± 0.04 (1)	0.4646 ± 0.04 (2)	0.4175 ± 0.10 (5)
yeast	0.4700 ± 0.05 (1)	0.4462 ± 0.04 (3)	0.4198 ± 0.05 (4)	0.4695 ± 0.05 (2)	0.4194 ± 0.03 (5)	0.4115 ± 0.03 (6)	0.4018 ± 0.03 (7)
spam	0.9154 ± 0.01 (4)	0.9256 ± 0.01 (1)	0.8968 ± 0.02 (6)	0.9236 ± 0.01 (2)	0.9206 ± 0.01 (3)	0.9013 ± 0.01 (5)	0.8808 ± 0.01 (7)
lymphography	0.5011 ± 0.13 (1)	0.4120 ± 0.08 (3)	0.4825 ± 0.13 (2)	0.3881 ± 0.03 (6)	0.4008 ± 0.03 (4)	0.3964 ± 0.03 (5)	0.3561 ± 0.04 (7)
movement_libras	0.6955 ± 0.05 (2)	0.6914 ± 0.06 (3)	0.6995 ± 0.05 (1)	0.5758 ± 0.05 (6)	0.5934 ± 0.05 (5)	0.6325 ± 0.06 (4)	0.4584 ± 0.06 (7)
SAheart	0.6105 ± 0.04 (5)	0.6279 ± 0.05 (4)	0.5929 ± 0.04 (7)	0.6322 ± 0.04 (2)	0.6368 ± 0.04 (1)	0.6290 ± 0.05 (3)	0.6081 ± 0.04 (6)
zoo	0.6541 ± 0.12 (4)	0.7949 ± 0.12 (1)	0.6618 ± 0.12 (3)	0.7736 ± 0.11 (2)	0.4358 ± 0.10 (7)	0.5625 ± 0.07 (6)	0.5627 ± 0.07 (5)
Average rank	3.87	**2.68**	4.83	3.48	3.27	3.75	6.12

Table A.8: Noise Level: 20% . Each cell in the table shows the $F_1^{(macro)}$-measure (higher value is better) from the 20 times 4-fold cross-validation experiment with 20% noise induced on the class labels. The values in parenthesis is the relative ranking of the algorithm on the dataset in the corresponding row (lower ranks are better).

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
mushroom	0.9939 ± 0.00 (5)	0.9943 ± 0.00 (4)	0.9964 ± 0.00 (3)	0.9981 ± 0.00 (2)	0.9984 ± 0.00 (1)	0.9912 ± 0.01 (7)	0.9914 ± 0.00 (6)
iris	0.8457 ± 0.07 (6)	0.9208 ± 0.05 (4)	0.7999 ± 0.06 (7)	0.9199 ± 0.05 (5)	0.9560 ± 0.03 (1)	0.9537 ± 0.03 (2)	0.9369 ± 0.04 (3)
glass	0.6253 ± 0.08 (2)	0.6242 ± 0.08 (3)	0.5804 ± 0.09 (5)	0.5985 ± 0.09 (4)	0.5692 ± 0.07 (6)	0.6284 ± 0.08 (1)	0.5329 ± 0.09 (7)
car_eval	0.8347 ± 0.03 (3)	0.8403 ± 0.04 (2)	0.6342 ± 0.04 (7)	0.8512 ± 0.04 (1)	0.7853 ± 0.04 (6)	0.8151 ± 0.05 (4)	0.8124 ± 0.04 (5)
cmc	0.5183 ± 0.02 (4)	0.5167 ± 0.02 (5)	0.4847 ± 0.03 (7)	0.5202 ± 0.02 (3)	0.5284 ± 0.02 (2)	0.5285 ± 0.02 (1)	0.5006 ± 0.03 (6)
tvowel	0.8244 ± 0.03 (1)	0.8198 ± 0.03 (4)	0.7136 ± 0.04 (7)	0.8226 ± 0.03 (3)	0.8237 ± 0.03 (2)	0.7897 ± 0.03 (5)	0.7669 ± 0.03 (6)
balance_scale	0.5856 ± 0.04 (4)	0.5951 ± 0.03 (2)	0.5442 ± 0.03 (7)	0.5960 ± 0.03 (1)	0.5892 ± 0.03 (3)	0.5660 ± 0.03 (5)	0.5471 ± 0.04 (6)
flags	0.2893 ± 0.06 (3)	0.2824 ± 0.05 (4)	0.2934 ± 0.05 (2)	0.2964 ± 0.05 (1)	0.2609 ± 0.04 (5)	0.2598 ± 0.05 (6)	0.2243 ± 0.04 (7)
german	0.6311 ± 0.03 (5)	0.6613 ± 0.03 (2)	0.6158 ± 0.03 (7)	0.6496 ± 0.03 (4)	0.6671 ± 0.03 (1)	0.6535 ± 0.03 (3)	0.6267 ± 0.04 (6)
ilpd	0.5794 ± 0.04 (5)	0.5836 ± 0.04 (4)	0.5667 ± 0.04 (7)	0.5843 ± 0.05 (3)	0.5909 ± 0.04 (1)	0.5848 ± 0.04 (2)	0.5713 ± 0.04 (6)
ionosphere	0.8458 ± 0.04 (4)	0.8682 ± 0.04 (3)	0.8260 ± 0.04 (6)	0.8397 ± 0.04 (5)	0.8731 ± 0.04 (2)	0.8786 ± 0.04 (1)	0.8053 ± 0.06 (7)
knowledge	0.8762 ± 0.04 (5)	0.8970 ± 0.03 (1)	0.8459 ± 0.04 (7)	0.8862 ± 0.03 (3)	0.8621 ± 0.03 (6)	0.8957 ± 0.04 (2)	0.8836 ± 0.04 (4)
vertebral	0.7763 ± 0.05 (5)	0.8039 ± 0.05 (3)	0.7563 ± 0.05 (7)	0.7847 ± 0.04 (4)	0.8211 ± 0.05 (1)	0.8104 ± 0.05 (2)	0.7608 ± 0.05 (6)
sonar	0.7274 ± 0.07 (5)	0.7423 ± 0.07 (1)	0.7340 ± 0.07 (4)	0.7104 ± 0.07 (6)	0.7410 ± 0.07 (2)	0.7384 ± 0.07 (3)	0.6257 ± 0.08 (7)
skulls	0.2173 ± 0.06 (6)	0.2390 ± 0.07 (2)	0.2315 ± 0.06 (4)	0.2385 ± 0.06 (3)	0.2299 ± 0.06 (5)	0.2402 ± 0.06 (1)	0.1998 ± 0.05 (7)
diabetes	0.8262 ± 0.07 (6)	0.9127 ± 0.05 (5)	0.8091 ± 0.07 (7)	0.9196 ± 0.05 (4)	0.9357 ± 0.04 (3)	0.9649 ± 0.04 (1)	0.9488 ± 0.06 (2)
physio	0.7971 ± 0.05 (7)	0.9012 ± 0.03 (2)	0.8001 ± 0.05 (6)	0.8879 ± 0.03 (4)	0.9142 ± 0.03 (1)	0.8983 ± 0.03 (3)	0.8655 ± 0.04 (5)
breasttissue	0.5644 ± 0.09 (7)	0.6337 ± 0.09 (1)	0.5745 ± 0.10 (5)	0.6187 ± 0.08 (2)	0.6051 ± 0.08 (4)	0.6183 ± 0.07 (3)	0.5737 ± 0.09 (6)
bupa	0.6388 ± 0.05 (5)	0.6600 ± 0.05 (3)	0.6236 ± 0.05 (6)	0.6418 ± 0.05 (4)	0.6710 ± 0.04 (1)	0.6606 ± 0.05 (2)	0.5981 ± 0.05 (7)
cleveland	0.2819 ± 0.05 (6)	0.2958 ± 0.05 (2)	0.3033 ± 0.05 (1)	0.2858 ± 0.05 (5)	0.2913 ± 0.05 (4)	0.2941 ± 0.05 (3)	0.2761 ± 0.05 (7)
haberman	0.5216 ± 0.06 (6)	0.5520 ± 0.06 (4)	0.5172 ± 0.05 (7)	0.5619 ± 0.05 (3)	0.5956 ± 0.05 (1)	0.5742 ± 0.05 (2)	0.5442 ± 0.06 (5)
hayes_roth	0.6835 ± 0.10 (2)	0.6840 ± 0.09 (1)	0.6451 ± 0.09 (3)	0.5240 ± 0.08 (6)	0.4190 ± 0.09 (7)	0.6130 ± 0.09 (4)	0.5557 ± 0.10 (5)
monks	0.8150 ± 0.04 (5)	0.8836 ± 0.04 (1)	0.7689 ± 0.04 (7)	0.8402 ± 0.05 (3)	0.8253 ± 0.04 (4)	0.8428 ± 0.05 (2)	0.8036 ± 0.06 (6)
newthyroid	0.3845 ± 0.05 (6)	0.4459 ± 0.04 (4)	0.3749 ± 0.05 (7)	0.4467 ± 0.05 (3)	0.4724 ± 0.05 (2)	0.4835 ± 0.04 (1)	0.4137 ± 0.10 (5)
yeast	0.4512 ± 0.05 (2)	0.4216 ± 0.03 (3)	0.3847 ± 0.04 (6)	0.4534 ± 0.05 (1)	0.4164 ± 0.03 (4)	0.4004 ± 0.03 (5)	0.3846 ± 0.03 (7)
spam	0.9089 ± 0.01 (4)	0.9244 ± 0.01 (1)	0.8884 ± 0.02 (6)	0.9211 ± 0.01 (2)	0.9206 ± 0.01 (3)	0.8989 ± 0.01 (5)	0.8681 ± 0.01 (7)
lymphography	0.4576 ± 0.13 (1)	0.4130 ± 0.07 (3)	0.4384 ± 0.11 (2)	0.3941 ± 0.04 (4)	0.3933 ± 0.04 (5)	0.3899 ± 0.03 (6)	0.3654 ± 0.04 (7)
movement_libras	0.6627 ± 0.05 (1)	0.6585 ± 0.05 (3)	0.6612 ± 0.05 (2)	0.5284 ± 0.05 (6)	0.5559 ± 0.06 (5)	0.6263 ± 0.05 (4)	0.4527 ± 0.05 (7)
SAheart	0.5972 ± 0.05 (5)	0.6205 ± 0.04 (3)	0.5762 ± 0.05 (7)	0.6039 ± 0.04 (4)	0.6368 ± 0.04 (1)	0.6349 ± 0.05 (2)	0.5806 ± 0.05 (6)
zoo	0.6653 ± 0.13 (4)	0.7603 ± 0.12 (1)	0.6714 ± 0.12 (3)	0.7389 ± 0.12 (2)	0.5562 ± 0.10 (6.5)	0.5573 ± 0.08 (5)	0.5562 ± 0.07 (6.5)
Average rank	4.33	**2.70**	5.40	3.37	3.18	3.10	5.92

AppendixB. Complete statistical test results

Table B.9: Result of post-hoc Friedman Aligned Rank test with Finner p-value adjustment over the different datasets using with different induced noise in the class-labels. Lower diagonal: post-hoc Friedman Aligned Rank Test p-values after the Finner adjustment. * $\alpha = 0.1$, ** $\alpha = 0.05$ and *** $\alpha = 0.01$

(a) No induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging
KFHE-l	0.296371					
AdaBoost	0.717099	0.485252				
GBM	0.101786	0.573588	0.201563			
S-GBM	0.000174 ***	0.009166 ***	0.000696 ***	0.042330 **		
Bagging	0.000015 ***	0.001233 ***	0.000072 ***	0.009166 ***	0.573588	
CART	0.000000 ***	0.000001 ***	0.000000 ***	0.000013 ***	0.022192 **	0.089728 *

(b) 5% induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging
KFHE-l	0.783059					
AdaBoost	0.286251	0.412077				
GBM	0.300240	0.430539	0.952742			
S-GBM	0.020310 **	0.042946 **	0.247823	0.236744		
Bagging	0.004712 ***	0.011182 **	0.087277 *	0.082096 *	0.601616	
CART	0.000000 ***	0.000000 ***	0.000011 ***	0.000011 ***	0.003113 ***	0.013687 **

(c) 10% induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging
KFHE-l	0.195755					
AdaBoost	0.195755	0.007206 ***				
GBM	0.963993	0.195755	0.194619			
S-GBM	0.505287	0.043617 **	0.505287	0.505287		
Bagging	0.505287	0.042152 **	0.505287	0.498988	0.959589	
CART	0.000621 ***	0.000001 ***	0.043617 **	0.000621 ***	0.007206 ***	0.007206 ***

(d) 15% induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging
KFHE-l	0.065548 *					
AdaBoost	0.099521 *	0.000256 ***				
GBM	0.813436	0.085647 *	0.074584 *			
S-GBM	0.925528	0.056191 *	0.118441	0.77674		
Bagging	0.854408	0.076564 *	0.074584 *	0.938200	0.811715	
CART	0.000334 ***	0.000000 ***	0.074584 *	0.000256 ***	0.000436 ***	0.000256 ***

(e) 20% induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging
KFHE-l	0.009541 ***					
AdaBoost	0.057691 *	0.000007 ***				
GBM	0.384210	0.113135	0.003655 ***			
S-GBM	0.354782	0.127197	0.003080 ***	0.933339		
Bagging	0.109001	0.395131	0.000261 ***	0.461039	0.479407	
CART	0.007492 ***	0.000000 ***	0.474921	0.000261 ***	0.000261 ***	0.000011 ***

Table B.10: Result of pairwise Wilcoxon's Signed Rank Sum test over the different datasets using with different induced noise in the class-labels, to compare individual isolated pair of algorithms. Upper diagonal: win/lose/tie. Lower diagonal: Wilcoxon's Signed Rank Sum Test p-values. *: $\alpha = 0.1$, **: $\alpha = 0.05$ and ***: $\alpha = 0.01$. The p-values for a pair of algorithms are to be interpreted in isolation and not to be combined with more than one methods.

(a) No induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
KFHE-e		(19/11/0)	(13/16/1)	(23/7/0)	(21/9/0)	(24/6/0)	(26/4/0)
KFHE-l	0.009519 ***		(12/18/0)	(16/14/0)	(20/10/0)	(25/5/0)	(26/4/0)
AdaBoost	0.491795	0.028548 **		(19/11/0)	(20/10/0)	(22/8/0)	(25/5/0)
GBM	0.003018 ***	0.144739	0.013515 **		(22/8/0)	(20/10/0)	(25/5/0)
S-GBM	0.001128 ***	0.004921 ***	0.002834 ***	0.003418 ***		(19/11/0)	(25/5/0)
Bagging	0.000210 ***	0.000034 ***	0.000415 ***	0.004108 ***	0.336640		(25/4/1)
CART	0.000010 ***	0.000006 ***	0.000019 ***	0.000055 ***	0.002057 ***	0.000016 ***	

(b) 5% induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
KFHE-e		(15/15/0)	(20/10/0)	(16/14/0)	(19/11/0)	(22/8/0)	(26/4/0)
KFHE-l	0.344180		(16/14/0)	(18/12/0)	(19/11/0)	(22/7/1)	(27/2/1)
AdaBoost	0.020351 **	0.085688 *		(14/16/0)	(17/13/0)	(18/12/0)	(24/6/0)
GBM	0.299968	0.092311 *	0.314422		(21/9/0)	(22/8/0)	(27/3/0)
S-GBM	0.028548 **	0.020862 **	0.118468	0.016213 **		(19/10/1)	(24/6/0)
Bagging	0.005874 ***	0.000320 ***	0.049937 **	0.012819 **	0.289329		(25/4/1)
CART	0.000020 ***	0.000005 ***	0.000095 ***	0.000041 ***	0.000386 ***	0.000007 ***	

(c) 10% induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
KFHE-e		(11/19/0)	(22/8/0)	(12/18/0)	(14/16/0)	(17/13/0)	(23/7/0)
KFHE-l	0.015399 **		(24/6/0)	(20/10/0)	(18/12/0)	(20/9/1)	(27/2/1)
AdaBoost	0.001053 ***	0.000518 ***		(8/22/0)	(12/18/0)	(14/16/0)	(19/11/0)
GBM	0.471305	0.034357 **	0.065296 *		(15/15/0)	(17/13/0)	(27/3/0)
S-GBM	0.229510	0.067954 *	0.329165	0.180019		(20/10/0)	(24/6/0)
Bagging	0.122595	0.002125 ***	0.258524	0.106679	0.19383		(25/4/1)
CART	0.000916 ***	0.000020 ***	0.029918 **	0.000142 ***	0.002341 ***	0.000030 ***	

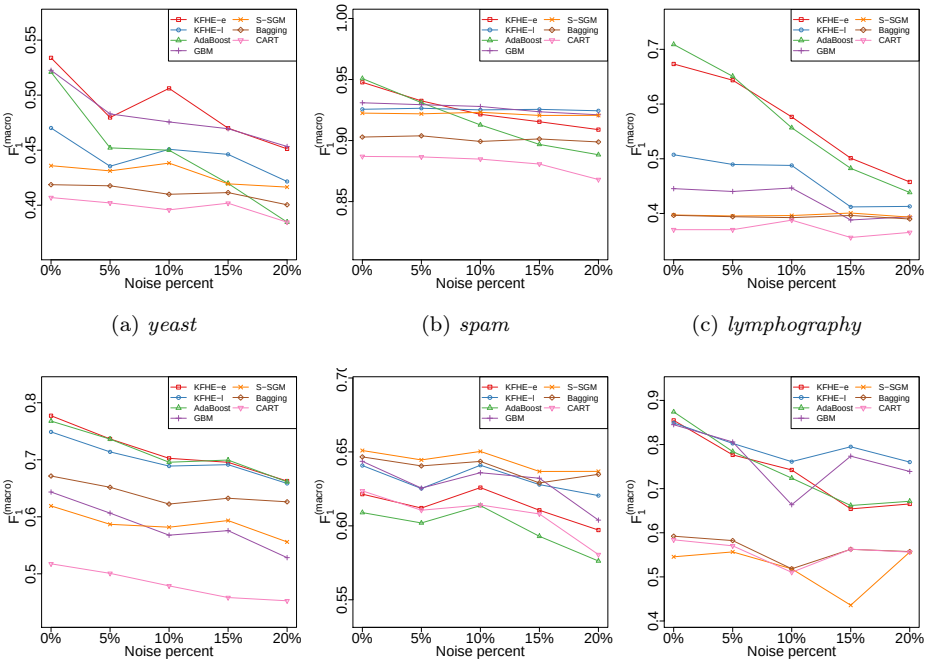
(d) 15% induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
KFHE-e		(7/22/1)	(21/9/0)	(11/19/0)	(13/17/0)	(16/13/1)	(25/5/0)
KFHE-l	0.001529 ***		(25/5/0)	(20/10/0)	(16/14/0)	(18/12/0)	(28/2/0)
AdaBoost	0.000285 ***	0.000047 ***		(7/23/0)	(12/18/0)	(12/18/0)	(20/10/0)
GBM	0.069314 ***	0.003418 ***	0.006987 ***		(13/17/0)	(13/16/1)	(27/3/0)
S-GBM	0.406508	0.046838 **	0.133351	0.479495		(19/11/0)	(27/3/0)
Bagging	0.386690	0.007829 ***	0.051005 *	0.459044	0.495897		(26/3/1)
CART	0.000518 ***	0.000002 ***	0.039323 **	0.000029 ***	0.000285 ***	0.000002 ***	

(e) 20% induced class-label noise

	KFHE-e	KFHE-l	AdaBoost	GBM	S-GBM	Bagging	CART
KFHE-e		(7/23/0)	(22/8/0)	(7/23/0)	(10/20/0)	(11/19/0)	(23/7/0)
KFHE-l	0.000464 ***		(25/5/0)	(19/11/0)	(16/14/0)	(18/12/0)	(28/2/0)
AdaBoost	0.000227 ***	0.000007 ***		(5/25/0)	(8/22/0)	(7/23/0)	(15/15/0)
GBM	0.015802 **	0.006226 ***	0.001479 ***		(13/17/0)	(10/20/0)	(27/3/0)
S-GBM	0.159245	0.174673	0.007398 ***	0.344180		(16/14/0)	(25/4/1)
Bagging	0.116442	0.067954 *	0.000854 ***	0.092311 *	0.406508		(29/1/0)
CART	0.005874 ***	0.000005 ***	0.430606	0.000019 ***	0.000084 ***	0.000001 ***	

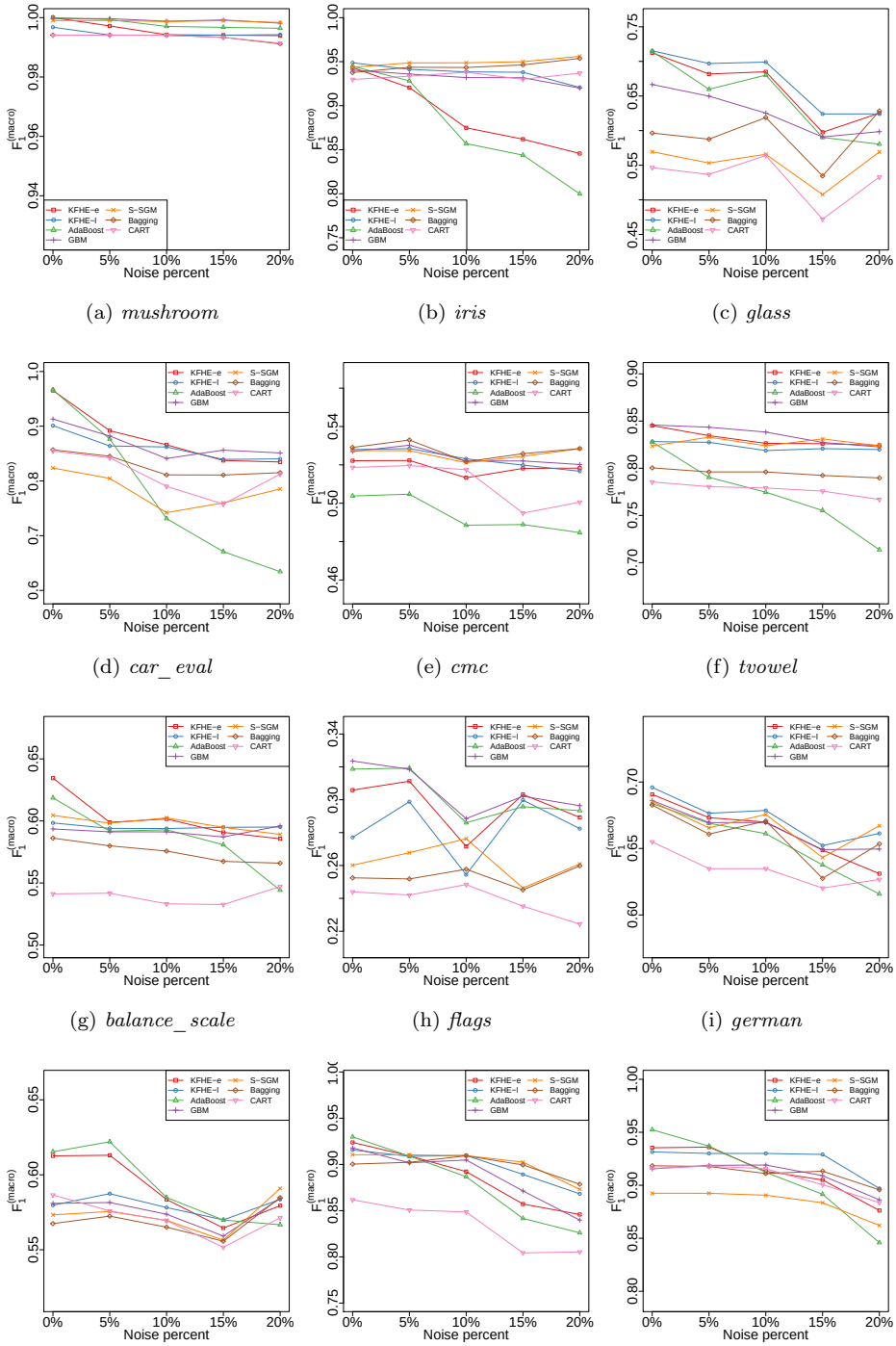
AppendixC. Plots of how model performance changes with noise



(a) *yeast* (b) *spam* (c) *lymphography*

(d) *movement_libras* (e) *SAheart* (f) *zoo*

Figure C.9: Changes in $F_1^{(macro)}$-score with the induced noise in class from datasets *yeast* to *zoo* labels



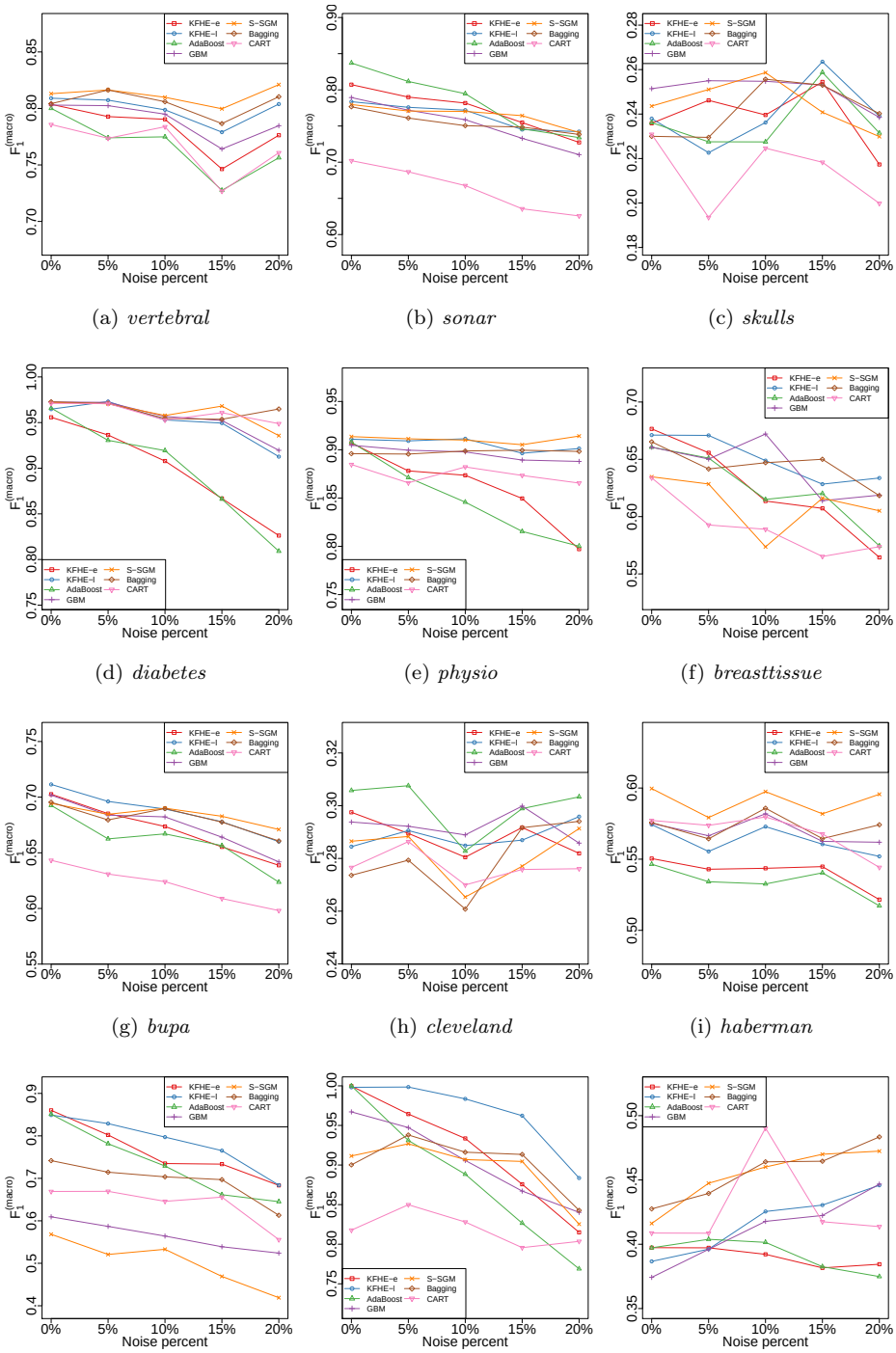
(a) *mushroom* (b) *iris* (c) *glass*

(d) *car_ eval* (e) *cmc* (f) *tvowel*

(g) *balance_ scale* (h) *flags* (i) *german*

(j) *ilpd* (k) *ionosphere* (l) *knowledge*

Figure C.10: Changes in $F_1^{(macro)}$-score with the induced noise in class from datasets *mushroom* to *knowledge* labels



(a) *vertebral* (b) *sonar* (c) *skulls*

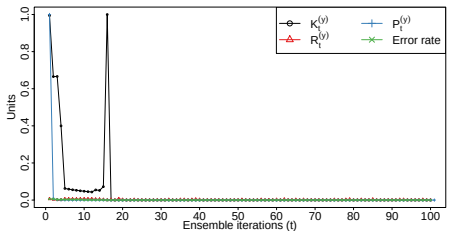
(d) *diabetes* (e) *physio* (f) *breasttissue*

(g) *bupa* (h) *cleveland* (i) *haberman*

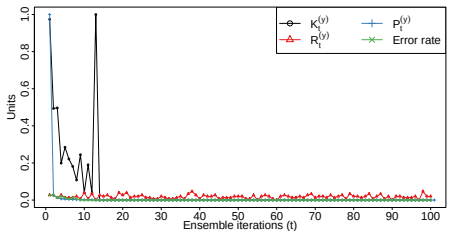
(j) *hayes_roth* (k) *monks* (l) *newthyroid*

Figure C.11: Changes in $F_1^{(macro)}$-score with the induced noise in class from datasets *vertebral* to *newthyroid* labels

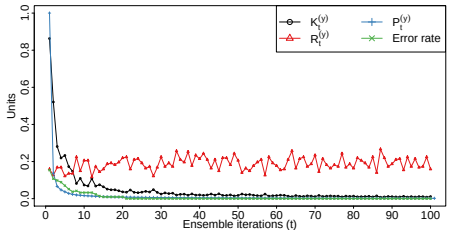
AppendixD. Plots of how model parameters change during training



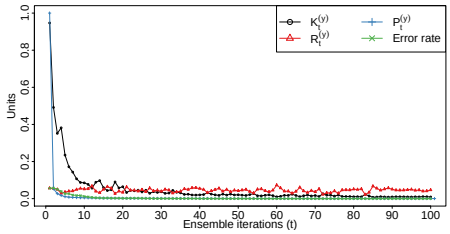
(a) *mushroom*



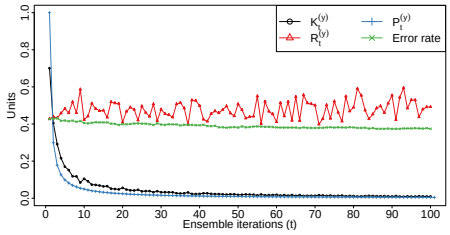
(b) *iris*



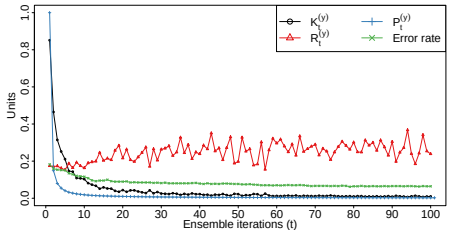
(c) *glass*



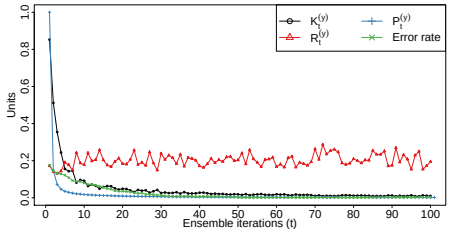
(d) *car_ eval*



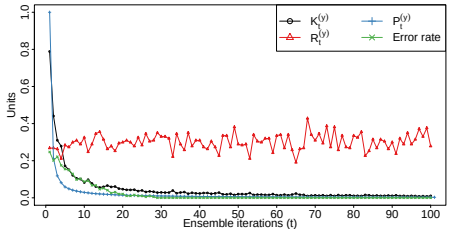
(e) *cmc*



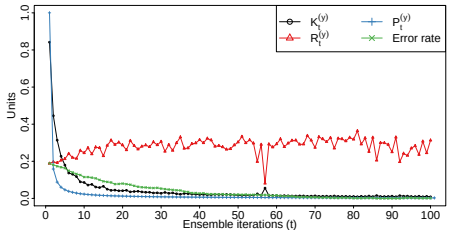
(f) *tvowel*



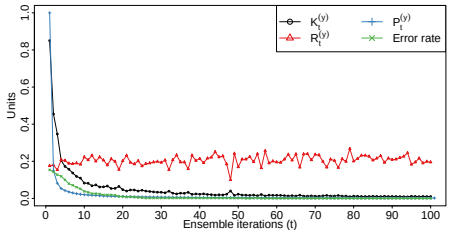
(g) *balance_ scale*



(h) *flags*

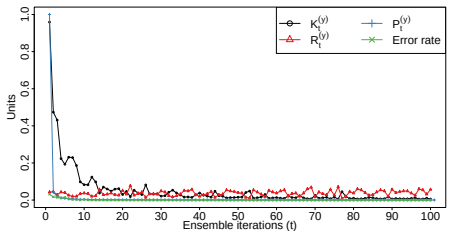


(i) *german*

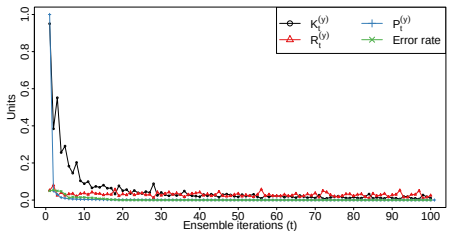


(j) *ilpd*

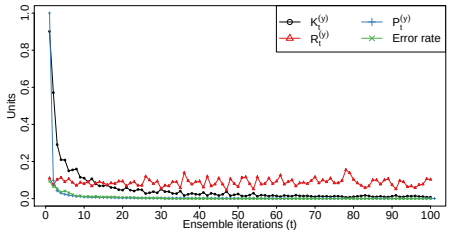
Figure D.12: Changes in the parameters and the misclassification rate for the training sets for KFHE. Datasets *mushroom* to *ilpd*



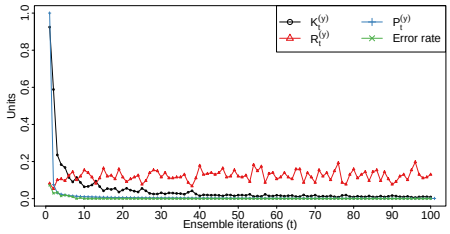
(a) *ionosphere*



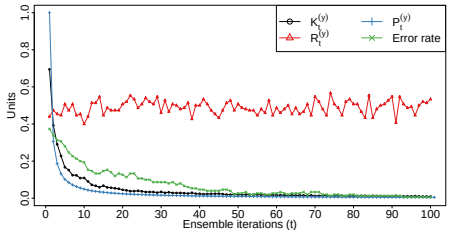
(b) *knowledge*



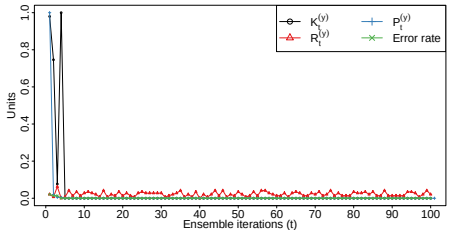
(c) *vertebral*



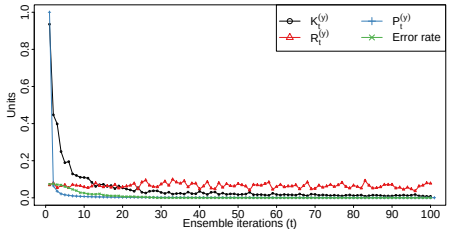
(d) *sonar*



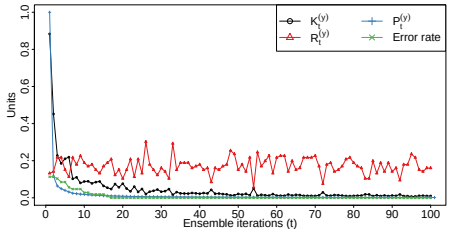
(e) *skulls*



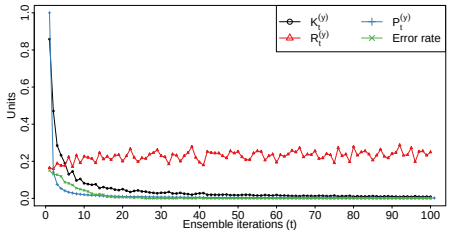
(f) *diabetes*



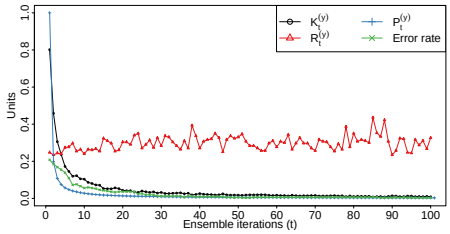
(g) *physio*



(h) *breasttissue*

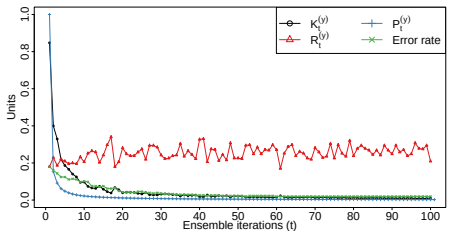


(i) *bupa*

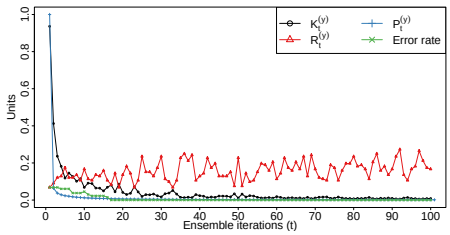


(j) *cleveland*

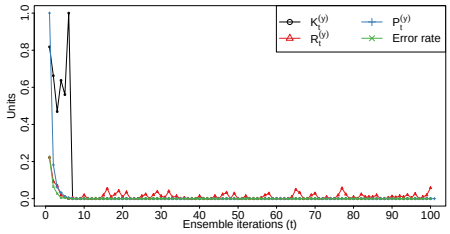
Figure D.13: Changes in the parameters and the misclassification rate for the training sets for KFHE. Datasets *ionosphere* to *cleveland*



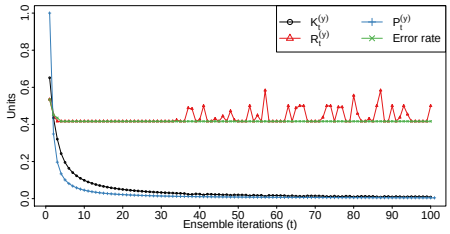
(a) *haberman*



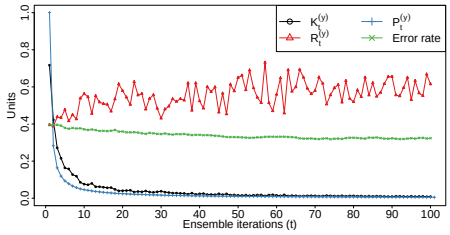
(b) *hayes_roth*



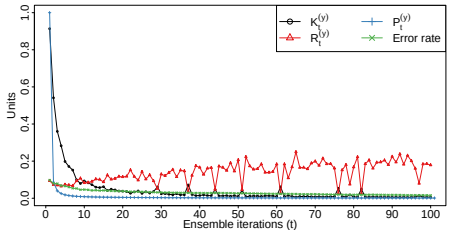
(c) *monks*



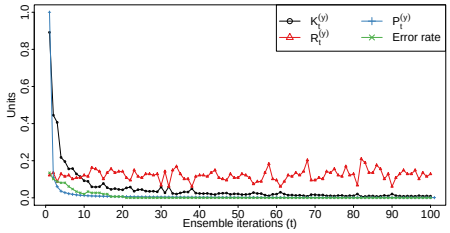
(d) *newthyroid*



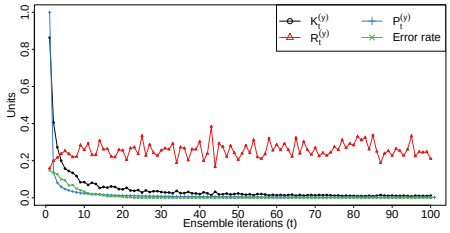
(e) *yeast*



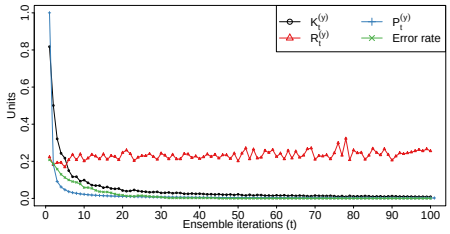
(f) *spam*



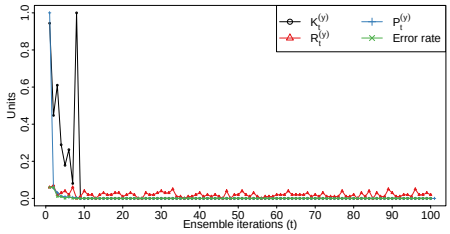
(g) *lymphography*



(h) *movement_libras*



(i) *SAheart*



(j) *zoo*

Figure D.14: Changes in the parameters and the misclassification rate for the training sets for KFHE. Datasets *haberman* to *zoo*

[1] *Esteban Alfaro, Matías Gámez, and Noelia García. adabag: An R package for classification with boosting and bagging.* Journal of Statistical Software, *54(2):1–35, 2013.* 4.2, 5.2

[2] *R. Barandela, R.M. Valdovinos, and J.S. Sánchez. New applications of ensembles of classifiers.* Pattern Analysis & Applications, *6(3):245–256, Dec 2003.* 2.1

[3] *Joseph K Bradley and Robert E Schapire. Filterboost: Regression and classification on large datasets. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors,* Advances in Neural Information Processing Systems 20, *pages 185–192. Curran Associates, Inc., 2008.* 2.1

[4] *Leo Breiman. Bagging predictors.* Machine Learning, *24(2):123–140, Aug 1996.* 1, 2.1, 5.2

[5] *Leo Breiman. Random forests.* Machine Learning, *45(1):5–32, Oct 2001.* 2.1

[6] *Jingjing Cao, Sam Kwong, and Ran Wang. A noise-detection based adaboost algorithm for mislabeled data.* Pattern Recognition, *45(12):4451 – 4465, 2012.* 2.1, 5.1

[7] *Ayhan Demiriz, Kristin P Bennett, and John Shawe-Taylor. Linear programming boosting via column generation.* Machine Learning, *46(1):225–254, 2002.* 2.1

[8] *Thomas G. Dietterich. Ensemble methods in machine learning. In* Multiple Classifier Systems, *pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.* 3.2, 1

[9] *Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization.* Machine Learning, *40(2):139–157, Aug 2000.* 1, 2.1, 5.1, 6.2

[10] *Carlos Domingo and Osamu Watanabe. Madaboost: A modification of adaboost. In* Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, *COLT '00, pages 180–189, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.* 2.1

[11] *Geir Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation.* Ocean dynamics, *53(4):343–367, 2003.* 2.2

[12] *Yoav Freund. An adaptive version of the boost by majority algorithm.* Machine Learning, *43(3):293–318, Jun 2001.* 2.1

[13] *Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In* European conference on computational learning theory, *pages 23–37. Springer, 1995.* 2.1

[14] *Rina Friedberg, Julie Tibshirani, Susan Athey, and Stefan Wager. Local linear forests.* arXiv preprint arXiv:1807.11408*, 2018.* 2.1

[15] *Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors).* The annals of statistics*, 28(2):337–407, 2000.* 2.1

[16] *Jerome H. Friedman. Greedy function approximation: A gradient boosting machine.* Annals of Statistics*, 29:1189–1232, 2000.* 1, 2.1, 5.2

[17] *Jerome H. Friedman. Stochastic gradient boosting.* Computational Statistics & Data Analysis*, 38(4):367 – 378, 2002. Nonlinear Methods and Data Mining.* 2.1, 5.2

[18] *Salvador García, Alberto Fernández, Julián Luengo, and Francisco Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power.* Information Sciences*, 180(10):2044 – 2064, 2010.* 6.3, 6.3.1, 6.3.2

[19] *L. K. Hansen and P. Salamon. Neural network ensembles.* IEEE Trans. Pattern Anal. Mach. Intell.*, 12(10):993–1001, October 1990.* 2.1

[20] *Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost.* Statistics and its Interface*, 2(3):349–360, 2009.* 1, 2.1

[21] *Trevor Hastie, Robert Tibshirani, and Jerome Friedman.* The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.* 2.1

[22] *R. E. Kalman. A new approach to linear filtering and prediction problems.* ASME Journal of Basic Engineering*, 1960.* 1, 1, 2.2, 2.2

[23] *John D Kelleher, Brian Mac Namee, and Aoife D'Arcy. Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies, 2015.* 1, 5.1, 5.2

[24] *M. Lichman. UCI machine learning repository, 2013.* 5.1

[25] *Peter S Maybeck.* Stochastic models, estimation, and control*, volume 3. Academic press, 1982.* 1, 1, 2.2

[26] *Eitan Menahem, Lior Rokach, and Yuval Elovici. Troika - an improved stacking schema for classification tasks.* Information Sciences*, 179(24):4097 – 4122, 2009.* 2.1

[27] *Christopher K. Monson and Kevin D. Seppi. The kalman swarm: A new approach to particle motion in swarm optimization. In* in Swarm Optimization, "Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, pages 140–150, 2004.* 2.2

[28] *Anil Narassiguin, Mohamed Bibimoune, Haytham Elghazel, and Alex Aussem. An extensive empirical comparison of ensemble learning methods for binary classification.* Pattern Analysis and Applications, *19(4):1093–1128, Nov 2016.* 1

[29] *David Opitz and Richard Maclin. Popular ensemble methods: An empirical study.* J. Artif. Int. Res., *11(1):169–198, July 1999.* 1

[30] *Arjun Pakrashi. A new hybrid clustering approach based on heuristic kalman algorithm. In* Swarm, Evolutionary, and Memetic Computing, *pages 445–455, Cham, 2015. Springer International Publishing.* 2.2

[31] *Arjun Pakrashi and Bidyut B. Chaudhuri. A kalman filtering induced heuristic optimization based partitional data clustering.* Information Sciences, *369(Supplement C):704 – 717, 2016.* 2.2

[32] *Juan J. Rodriguez, Ludmila I. Kuncheva, and Carlos J. Alonso. Rotation forest: A new classifier ensemble method.* IEEE Trans. Pattern Anal. Mach. Intell., *28(10):1619–1630, October 2006.* 2.1

[33] *Maryam Sabzevari, Gonzalo Martínez-Muñoz, and Alberto Suárez. Vote-boosting ensembles.* Pattern Recognition, *83:119 – 133, 2018.* 2.1

[34] *Robert E. Schapire. The strength of weak learnability.* Machine Learning, *5(2):197–227, Jun 1990.* 2.1

[35] *Alexander K. Seewald. How to make stacking better and faster while also taking care of an unknown weakness. In* Proceedings of the Nineteenth International Conference on Machine Learning, *ICML '02, pages 554–561, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.* 2.1

[36] *Joko Siswantoro, Anton Satria Prabuwono, Azizi Abdullah, and Bahari Idrus. A linear model based on kalman filter for improving neural network classification performance.* Expert Systems with Applications, *49(Supplement C):112 – 122, 2016.* 2.2

[37] *Bo Sun, Songcan Chen, Jiandong Wang, and Haiyan Chen. A robust multi-class adaboost algorithm for mislabeled noisy data.* Knowledge-Based Systems, *102:87 – 102, 2016.* 2.1, 5.1

[38] *Kai Ming Ting and Ian H. Witten. Stacked generalization: when does it work? In* in Procs. International Joint Conference on Artificial Intelligence, *pages 866–871. Morgan Kaufmann, 1997.* 2.1

[39] *R. Toscano and P. Lyonnet. A new heuristic approach for non-convex optimization problems.* Information Sciences, *180(10):1955 – 1966, 2010. Special Issue on Intelligent Distributed Information Systems.* 2.2

[40] *Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, Chapel Hill, NC, USA, 1995.* 2.2

[41] *Greg Ridgeway with contributions from others.* gbm: Generalized Boosted Regression Models, *2017. R package version 2.1.3.* 5.2

[42] *David H. Wolpert. Stacked generalization.* Neural Networks, *5(2):241 – 259, 1992.* 2.1

[43] *Chun-Xia Zhang and Jiang-She Zhang. Rotboost: A technique for combining rotation forest and adaboost.* Pattern Recognition Letters, *29(10):1524 – 1536, 2008.* 2.1, 5.1

[44] *Z.H. Zhou.* Ensemble Methods: Foundations and Algorithms. *Chapman & Hall/CRC machine learning & pattern recognition series. CRC Press, 2012.* 2.1

[45] *Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost.* Ann Arbor, *1001(48109):1612, 2006.* 1, 2.1, 5.2