Received: 24 August 2017 | Revised: 6 December 2017 | Accepted: 23 January 2018

DOI: 10.1002/widm.1249

WIREs DATA MINING AND KNOWLEDGE DISCOVERY WILEY

ADVANCED REVIEW

Ensemble learning: A survey

Omer Sagi | Lior Rokach

Department of Software and Information Systems Engineering, Ben-Gurion University, Beersheba, Israel

Correspondence
Lior Rokach, Department of Software and Information Systems Engineering, Ben-Gurion University, Beersheba, Israel.
Email: liorrk@bgu.ac.il

Ensemble methods are considered the state-of-the art solution for many machine learning challenges. Such methods improve the predictive performance of a single model by training multiple models and combining their predictions. This paper introduce the concept of ensemble learning, reviews traditional, novel and state-of-the-art ensemble methods and discusses current challenges and trends in the field.

This article is categorized under:
Algorithmic Development > Ensemble Methods
Technologies > Machine Learning
Technologies > Classification

KEYWORDS

boosting, classifier combination, ensemble models, machine-learning, mixtures of experts, multiple classifier system, random forest

1 | INTRODUCTION

Ensemble learning is an umbrella term for methods that combine multiple inducers to make a decision, typically in supervised machine learning tasks. An inducer, also referred as a base-learner, is an algorithm that takes a set of labeled examples as input and produces a model (e.g., a classifier or regressor) that generalizes these examples. By using the produced model, predictions can be drawn for new unlabeled examples. An ensemble inducer can be of any type of machine learning algorithm (e.g., decision tree, neural network, linear regression model, etc.). The main premise of ensemble learning is that by combining multiple models, the errors of a single inducer will likely be compensated by other inducers, and as a result, the overall prediction performance of the ensemble would be better than that of a single inducer.

Ensemble learning is usually regarded as the machine learning interpretation for the wisdom of the crowd. This concept can be illustrated through the story of Sir Francis Galton (1822–1911) who was an English philosopher and statistician that conceived the basic concept of standard deviation and correlation. While visiting a livestock fair, Galton conducted a simple weight guessing contest. The participants were asked to guess the weight of an ox. Hundreds of people participated in this contest, but no one succeeded in guessing the weight: 1,198 pounds. Much to his surprise, Galton found that the average of all guesses came quite close to the exact weight: 1,198 pounds. In this experiment, Galton revealed the power of combining many predictions in order to obtain an accurate prediction. Ensemble methods manifest this concept in machine learning challenges, where they result in improved predictive performance compared to a single model. In addition, when the computational cost of the participating inducers is low (e.g., decision tree), ensemble models are often very efficient.

Today, ensembles are considered as the state-of-the art approach for solving a plethora of machine learning challenges as shown in an extensive comparison of 179 classifiers from 17 families using 121 datasets from UCI, as well as various real life challenges (Fernández-Delgado, Cernadas, Barro, & Amorim, 2014). Recently, several comprehensive surveys on ensemble learning have been published in the literature (Gomes, Barddal, Enembreck, & Bifet, 2017; Kulkarni & Sinha, 2013; Rokach, 2016; Woźniak, Graña, & Corchado, 2014). While each of these surveys emphasizes a specific subresearch field, they do not present an holistic view of ensemble methods. Other extensive works (Polikar, 2006; Ren, Zhang, & Suganthan, 2016; Rokach, 2010) that provide a broad view of ensemble models fail to include several recently presented notable

developments and findings. This paper aims to serve as an introduction to ensemble models, and survey the state-of-the-art methods in the field. We also provide a thorough review of subjects that are currently under intense study such as the integration of ensemble learning in deep neural networks, distributed algorithms for training ensemble models, and transforming ensemble models into simpler models. The remainder of the paper is organized as follows: First, we introduce the foundations and origins of ensemble learning, explain how ensemble learning work and present its advantages. The next section presents different general approaches for building an ensemble model and combining inducer outputs. Then, we review some of the most commonly used methods for building an ensemble learner. Finally, we present consensus clustering and discuss different approaches used to simplify ensemble models.

2 | ENSEMBLE LEARNING FOUNDATIONS

Ensemble learning refers to the generation and combination of multiple inducers to solve a particular machine learning task. The intuitive explanation for the ensemble methodology stems from human nature and the tendency to gather different opinions and weigh and combine them to make a complex decision. The main idea is that weighing and aggregating several individual opinions will be better than choosing the opinion of one individual. An example for such a decision is matching a medical treatment to a disease (Polikar, 2006).

2.1 | Background

Marie Jean Antoine Nicolas de Caritat (1743–1794) was a French mathematician who presented the well-known Condorcet's jury theorem in 1785. The theorem refers to a jury of voters that are required to reach a decision regarding a binary outcome (e.g., whether or not to convict a defendant). If each voter has a probability $p > .5$ of being correct (each voter is more likely to be correct than a random guess), then adding more voters increases the probability of making the correct decision, assuming that votes are independent. This theorem has two major limitations: The assumption that the votes are independent and that there are only two possible outcomes. In his 2004 book, *The Wisdom of Crowds: Why the Many Are Smarter than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, James Michael Surowiecki elaborates upon this notion. He describes how combining information from different sources results in decisions that are often better than those of a single individual. Naturally, not all crowds are wise (e.g., greedy investors in a stock market bubble). According to Surowiecki, the wisdom of the crowd is likely to surpass a single decision maker when meeting the following criteria:

1. *Independence*: One's opinion is not affected by other opinions.
2. *Decentralization*: One is capable of specializing and making conclusions based on local information.
3. *Diversity of opinions*: One should hold private information, even if it is just an eccentric interpretation of the known facts.
4. *Aggregation*: Some mechanism exists for turning private judgments into a collective decision.

The manifestation of those concepts in the context of supervised learning has been explored since the 1970s when Tukey (1977) presented an ensemble of two linear regression models. Tukey suggested fitting the first linear regression model to the original data and fitting the second linear model to the residuals. Two years later, Dasarathy and Sheela (1979) suggested the composition of a classiffier system using two or more classifiers of different categories. The 1990s brought a significant stride in this field, as depicted in Figure 1. Hansen and Salamon (1990)) showed, for the first time that the generalization error of neural networks could be reduced when invoking ensembles of neural networks. Around the same time, Schapire (1990) described how combining several weak learners in a probably approximately correct (PAC) sense may outperform one strong learner. This work laid the foundation for the well-known AdaBoost (adaptive boosting) algorithm. Since then, it has been repeatedly shown that ensemble models improve the predictive performance of single models. Oza and Tumer (2008) reviewed application of ensemble methods in remote sensing, person recognition, one versus all recognition, and medicine. Research performed in the past few years presented the successful use of ensemble methods in object-detection (Paisitkriangkrai, Shen, & van den Hengel, 2016), education (Beemer, Spoon, He, Fan, & Levine, 2017), and malware detection (Idrees, Rajarajan, Conti, Chen, & Rahulamathavan, 2017).

2.2 | Why ensemble methods work?

There are several reasons why ensemble methods often improve predictive performance (Dietterich, 2002; Polikar, 2006):

- *Overfitting avoidance*: When just a small amount of data is available, a learning algorithm is prone to finding many different hypothesis that predict all of the training data perfectly while making poor predictions for unseen instances.

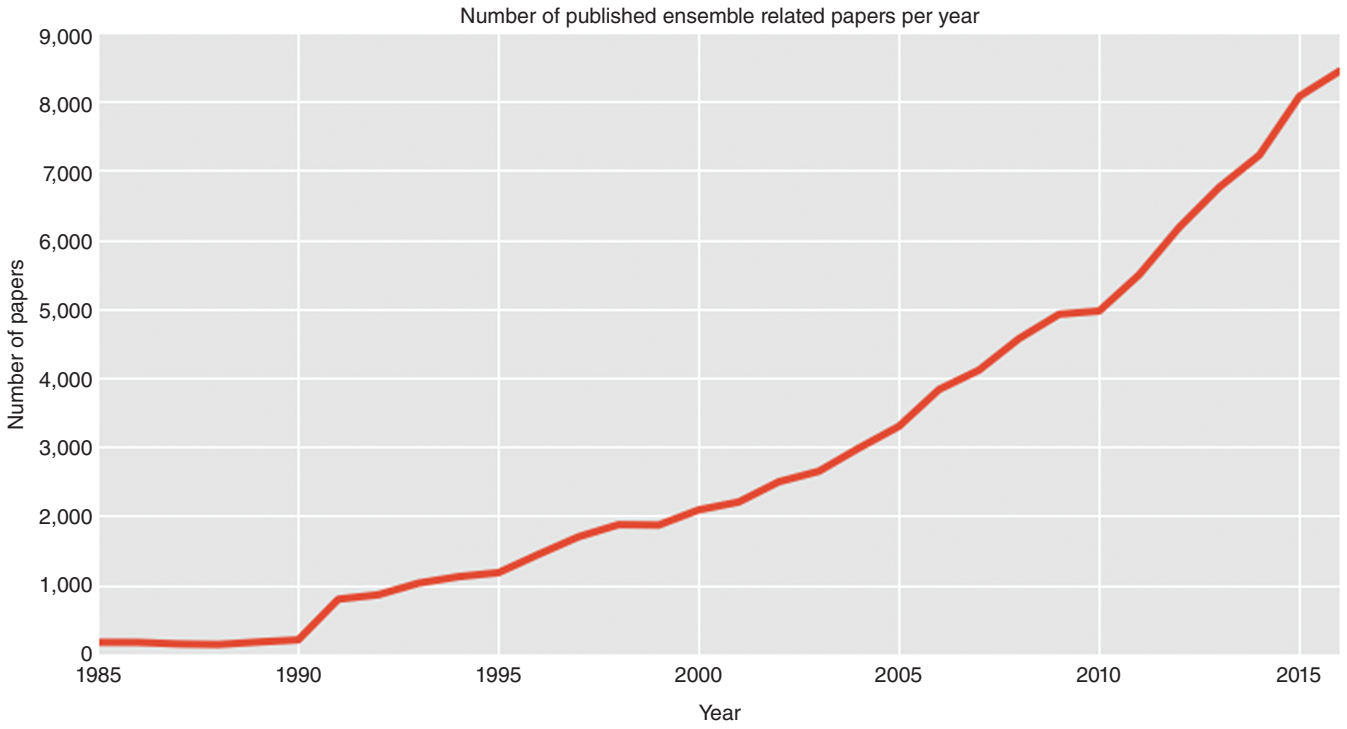

FIGURE 1 Number of published papers per year, based on searching the terms "ensemble" together with "machine learning" in the "web of science" database

Averaging different hypothesis reduces the risk of choosing an incorrect hypothesis and therefore, improves the overall predictive performance.

- *Computational advantage*: Single learners that conduct local searches may get stuck in local optima. By combining several learners, ensemble methods decrease the risk of obtaining a local minimum.
- *Representation*: The optimal hypothesis may be outside the space of any single model. By combining different models, the search space may be extended and hence, a better fit to the data space is achieved. Figure 2 presents the training set of a simple binary classification task using only two attributes. The goal is to induce a classifier that can classify new emails to either spam (sad face) or nonspam (smiley face) based on the email-length (x-axis) and number of recipients (y-axis). In this example we use a decision stump as a model. Decision stump is a weak classifier, consisting of a single split. In Figure 2a we can see that each of the decision stumps still misclassifies some of the training instances. Nevertheless, Figure 2b shows that by combining all three decision stumps, we obtain a perfect fit to the training instances.

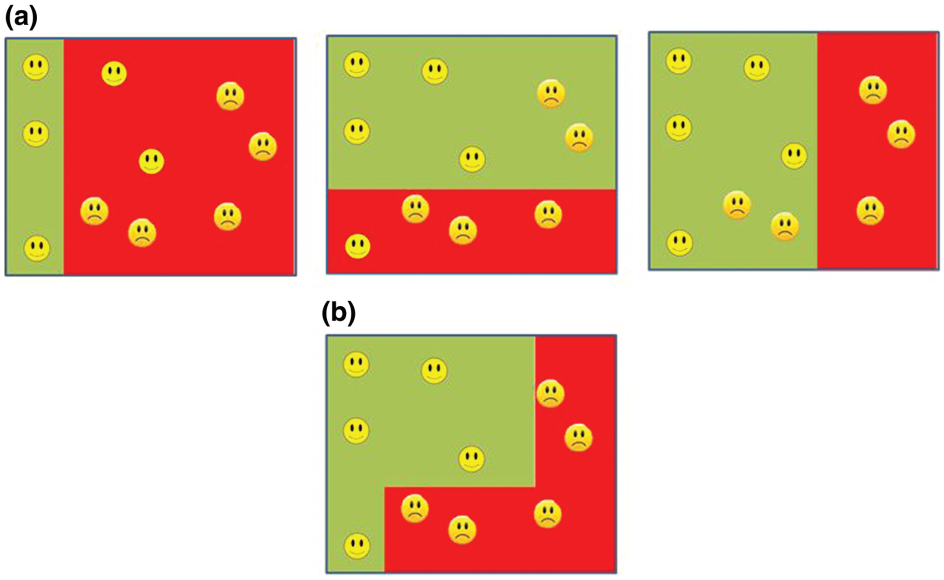

FIGURE 2 A forest of decision stumps provides a perfect classification

2.3 | Ensemble learning and unique machine learning challenges

There are several settings that pose nontrivial challenges to machine learning algorithms. Ensemble methods may be used to mitigate these challenges as described below:

- *Class imbalance*: There are many machine learning problems for which one class has substantially more examples than other classes (Japkowicz & Stephen, 2002). In such cases, machine learning algorithms may develop a preference for the majority class while ignoring minority classes. Ensemble methods may be applied in a way that mitigates the class imbalance problem. One example is to create an ensemble in which each of the inducers is trained using a balanced subsample of the data (Nikulin & Ng, 2009). Another work showed how combining random under-sampling techniques with ensemble techniques such as bagging or boosting may significantly improve predictive performance for problems with class imbalance (Galar, Fernandez, Barrenechea, Bustince, & Herrera, 2012). The prediction performance can be further improved by using the EUSBoost method which leverages evolutionary under-sampling to promote diversity among individual inducers (Galar, Fernández, Barrenechea, & Herrera, 2013).
- *Concept drift*: In many real-time machine learning applications, the distribution of features and the labels tend to change over time. This phenomenon frequently affects the predictive performance of the model over time. Ensemble based approaches often serve as a remedy for this problem. For example, in the dynamic weighted majority (DWM) method (Kolter & Maloof, 2003), individual decision trees are dynamically created and deleted according to changes in predictive performance. Another work examined how different diversity levels of ensembles affect their generalization for new concepts (Minku, White, & Yao, 2010). This work also shows that diversity alone may reduce the initial increase in error caused by a drift. The diversity for dealing with drifts (DDD) approach (Minku & Yao, 2012) manifests this concept by maintaining two ensembles of decision trees (decision forests) with two different levels of diversity. In this approach, the low-diversity forest is used before a drift detection takes place. After detecting the drift, new low-diversity and high-diversity forests are trained and predictions of new instances are drawn based on a weighted majority voting of (a) the old high-diversity forest, (b) the new low-diversity forest, and (c) the old low-diversity forest. ADWIN is another approach in which ensemble members can be reset when their accuracy degrades significantly (Bifet, Frank, Holmes, & Pfahringer, 2010). This method may also be integrated with online bagging ensembles (Bifet, Holmes, Pfahringer, Kirkby, & Gavaldà, 2009).
- *Curse of dimensionality*: Increasing the number of features fed into a machine learning model usually exponentially increases the search space and hence, the probability of fitting models that cannot be generalized. This phenomenon is known as the curse of dimensionality. Certain ensemble learning methods can be used to lessen the impact of the phenomenon. Bryll, Gutierrez-Osuna, and Quek (2003) introduced attribute bagging (AB), an approach in which an inducer is trained using a randomly selected subset of features. AB also tries to determine the appropriate size for the feature subset by evaluating the accuracy gained by different subset sizes. Improved decision forest (IDF; Huang, Fang, & Fan, 2010) is a decision forest algorithm that addresses high-dimensional datasets by applying a designated feature selection mechanism. It is suitable for gene expression data which is comprised of thousands of genes. Rokach (2008) presented a genetic algorithm that searches for the best mutually exclusive partition of the feature set, replacing the random partition. The optimal feature set partitioning (OFSP) method creates a fixed number of feature views in advanced towards reducing dimensionality and optimizing accuracy (Kumar & Minz, 2016).

3 | BUILDING AN ENSEMBLE MODEL

Given a dataset of n examples and m features $D = \{(x_i, y_i)\}$ ($|D| = n$, $x_i \in R^m$, $y_i \in R$), an ensemble learning model φ uses an aggregation function G that aggregates K inducers, $\{f_1, f_2, ..., f_k\}$ towards predicting a single output as follows:

$$\widehat{y_i} = \varphi(x_i) = G(f_1, f_2, \ldots, f_k) \tag{1}$$

where $\widehat{y_i} \in R$ for regression problems and $\widehat{y_i} \in Z$ for classification problems. Given this general framework, building an ensemble model involves the selection of a methodology for training the participating models and choosing a suitable process for combining the inducers' outputs.

3.1 | Model training

There are several ways to train an ensemble model to reach a desired outcome. However, some key principles must be considered when generating an ensemble model:

1. *Diversity*: The superior performance of ensemble models is achieved mainly due to the use of various "inductive biases" (Deng, Runger, Tuv, & Vladimir, 2013).Therefore, the participating inducers should be sufficiently diverse in order to gain a desired predictive performance.
2. *Predictive performance*: The individual inducer's predictive performance should be as high as possible and at least as good as a random model.

The two principles may seem to contradict each other at first glance. Furthermore, ensembles with diverse inducers do not always improve the predictive performance (Bi, 2012). The main idea in combining these two principles is to combine predictive inducers with uncorrelated errors in an ensemble, since it has been empirically and theoretically shown that the predictive performance of the entire ensemble has a positive correlation with the degree to which the errors made by individual inducers are uncorrelated (Ali & Pazzani, 1995). There are several approaches used to achieve the goal of including diverse inducers:

- *Input manipulation*: In this approach each base model is fitted using a different training subset so that a variety of inputs are used for the different base models. This method has been found to be effective for cases where small changes in the training set may result in a completely different model. In the most simple implementation of this approach, each model is trained using a slightly different sample. Class distribution among the different inducers may be random or determined according to the class distribution in the entire dataset (Chan & Stolfo, 1995a).
- *Manipulated learning algorithm*: In this approach, the use of each base model is altered. One way of doing this is to manipulate the way in which the base model traverses the hypothesis space. This is done by leading the base model to different convergence paths (Brown, Wyatt, & Tiňo, 2005). For example, when building an ensemble of decision trees, we can inject randomness by selecting one out of k best splitting attributes at each split. Distributing neighbors (Maudes, Rodríguez, & García-Osorio, 2009) is a method that expands the feature space by generating different combinations of the original features. Another way to create diversity in this manner is to train the base models with varied hyperparameter values (Lin & Chen, 2012). For example, training a neural-network inducer with different learning rates, varied layers and attributes, and so on.
- *Partitioning*: Diversity can be achieved by dividing the original dataset into smaller subsets and then using each subset to train a different inducer. In *horizontal partitioning*, we divide the original dataset into several sets that include the entire feature-set so that inducers differed only by their instances (Chawla, Hall, Bowyer, & Kegelmeyer, 2004). *Vertical partitioning* works in the opposite way as each inducer uses the same instances but with different features (Rokach, 2008). The feature subspace aggregating method (Feating; Ting, Wells, Tan, Teng, & Webb, 2011) divides the feature-space into mutually exclusive local regions that are defined over a fixed number of attributes. The user specifies the number of features used for dividing the subspace, and by that, determines the level of localization. Ordinal consistency driven feature subspace aggregating (Feating; Błaszczyński, Stefanowski, & Słowiński, 2016) subdivides attribute space into local regions as well. This method prioritizes partitions that result in consistent regions. CoFeating has been shown to perform well on datasets with a large number of attributes.
- *Output manipulation*: This approach refers to techniques that combine numerous binary classifiers into a single multiclass classifier. Error-correcting output codes (ECOC) is a succesful example of this approach (Dietterich & Bakiri, 1995). In this method, each class is encoded as an L-bit code-word where L is the number of classifiers participating in the ensemble. The purpose of each classifier is to predict a bit L of the code-word. Classifiers are then applied for new instances to generate L-bit strings that represent the predictions. The chosen class to be predicted for a given instance is the class whose code-word is the closest to the instance string. Closeness can be measured using different methods such as euclidean decoding and Hamming distance. Adaptive ECOC (AECOC) extends ECOC by adding conventionality reduction procedures when training the different binary classifiers (Zhong & Cheriet, 2013). N-ary error correcting coding scheme is a recently developed method that breaks the main multiclass classifier into simpler multiclass subproblems (Zhou, Tsang, Ho, & Muller, 2016).
- *Ensemble hybridization:* This approach combines at least two strategies when building the ensemble. The random forest algorithm is probably the most well-known manifestation of the hybridization approach as it manipulates the instances when building each tree, in addition to manipulating the learning algorithm by choosing randomally a subset of features at each node. RotBoost is an example of a hybrid of the rotation forest and AdaBoost algorithms (Zhang & Zhang, 2008). In each iteration a new rotation matrix is generated and used to create a dataset. The AdaBoost ensemble is induced from this dataset. In empirical testing RotBoost outperformed rotation forest and AdaBoost in terms of accuracy. The dynamic random forest algorithm (Bernard, Adam, & Heutte, 2012) relies upon an adaptive tree induction procedure. The training of each tree is guided by how it is complement the trees that have already been trained. This is done by

using resampling methods and randomization processes that relate to random forest. Nonlinear boosting projection (NLBP) combines boosting and subspace methods. In each iteration of inducer training, we construct a projection that takes into account instances that were miss-classified by previous classifiers (GarcÃa-Pedrajas, GarcÃa-Osorio, & Fyfe, 2007). In random rotation ensembles (Blaser & Fryzlewicz, 2016), the feature space is rotated when training each base-learner. This method was shown to be effective for problems that include many continuous features.

3.2 | Output fusion

Output fusion refers to the process of integrating the base model outputs into a single output. The following section summarizes the main approaches for combining the outputs.

- *Weighting methods*: Base model outputs can be combined by assigning weights to each base model. The weighting approach is most suitable for cases where the performance of the base models is comparable. Majority voting is the simplest weighting method for classification problems as the selected class is the one with the most votes. For regression problems, this concept is manifested by averaging the outputs across the inducers (Mendes-Moreira, Soares, Jorge, & Sousa, 2012). The idea of the distribution summation method in ensemble classifiers is to add the conditional probability vector obtained from each base model. The selected class is chosen based on the highest value in the total vector as follows:

$$\hat{y} = \operatorname{argmax}(\varphi(x_i)) \quad \text{where} \quad \varphi(x_i) = \sum_{k=1}^{K} f_k(x_i) \tag{2}$$

Another weighting approach is to assign a weight that is proportional to the inducers' strengths. This is commonly done in accordance with the predictive performance on a validation set (Opitz & Shavlik, 1996). In the Bayesian combination method, weighting is based on the probabilistic likelihood of obtaining the model given the entire dataset (Duan, Ajami, Gao, & Sorooshian, 2007). Schapire and Singer (1998) presented a weighting methodology that considers prediction confidence when aggregating the final decision. They apply the method in decision tree ensembles where prediction confidence is assigned according to the number of instances that have been reached to the corresponding leaf. The variance optimized bagging and bogging methods aim at optimizing a linear combination of the trees guided by variance reduction and preserving a prescribed accuracy (Derbeko, El-Yaniv, & Meir, 2002). Weights can be also assigned by using an objective function in which the weights are subject to $\sum w_i = 1$ and $-1 < w_i < 1$ where the target is minimizing the error (Mao et al., 2015). A recent study proposed the use of Matthews correlation coefficient (MCC), calculated over 10-fold cross-validation, as the quality measure of weights selection (Haque, Noman, Berretta, & Moscato, 2016).

- *Meta-learning methods*: Meta-learning is a process of learning from learners. Meta-learning models differ from standard machine learning models since they include more than one learning stage. In the meta-learning paradigm, the individual inducer outputs serve as an input to the meta-learner that generates the final output. Meta-learning methods work well in cases where certain base models have different performances on different subspaces. For example, when base models consistently correctly classify or consistently misclassify certain instances. Stacking is probably the most-popular meta-learning technique (Wolpert, 1992). By using a meta-learner, this method tries to induce which base models are reliable and which are not. In stacking we create a meta-dataset that contains the same number of instances found in the original dataset. However, instead of using the original input attributes, it uses the base models outputs as the input features. The predicted attribute remains the same as in the original dataset. A test instance is first predicted by each of the individual inducers. These predictions are then fed into a meta-level set that is fed into the meta-model which provides the final output. It is recommended that the original dataset be divided into two subsets. The first subset is reserved to form the meta-dataset, and the second subset is used to build the base learners. There are several modifications of the basic stacking algorithm. Troika (Menahem, Rokach, & Elovici, 2009) for example, aimed at improving the predictive performance specifically in multiclass datasets. Arbiter trees, combiner trees and grading (Chan & Stolfo, 1995b; Seewald & Fürnkranz, 2001) are additional meta-learning algorithms developed for integrating the base learners' outputs. Weighted bagging (Shieh & Kamm, 2009) features a preprocessing procedure in which probability weights are assigned to points based on their closeness to the target class using a kernel density estimator. This procedure reduces the effect of outliers on the model built. The mixture of experts (ME) is a very popular variant of meta-learning which relies on the idea of divide-and-conquer as the problem space is divided between a few experts. ME uses error functions in order to locate the most suitable base models for different distributions of the data space (Jacobs, Jordan, Nowlan, & Hinton, 1991). Subspaces may have soft "boundaries," namely, subspaces are allowed to overlap. Then, for each subspace one base model is selected to be responsible for providing the prediction.

TABLE 1 Method categories

Method name	Fusion method	Dependency	Training approach
AdaBoost	Weightning	Dependent	Input manipulation
Bagging	Weightning	Independent	Input manipulation
Random forest	Weightning	Independent	Ensemble hybridization
Random subspace methods	Weightning	Independent	Ensemble hybridization
Gradient boosting machines	Weightning	Dependent	Output manipulation
Error-correcting output codes	Weightning	Independent	Output manipulation
Rotation forest	Weightning	Independent	Manipulated learning
Extremely randomized trees	Weightning	Independent	Partitioning
Stacking	Meta-learning	Indpendent	Manipulated learning

In order to choose the best base model, another learning algorithm is usually used. Omari and Figueiras-Vidal (2015) introduced a postaggregation procedure for combining a large ensemble of decision trees. This method leverages the advantages of meta-learning with the advantages of weighting. It consists of two steps. In the first step, a weighting method (such as distribution summation) is used to fuse the individual trees' outputs and to create an initial aggregation. In the second step, a postaggregation is obtained by applying a machine learning algorithm on the original training instances along with their initial aggregation. Masoudnia and Ebrahimpour (2014) recently published a comprehensive survey of ME methods.

4 | ENSEMBLE METHODS

Ensemble methods can be divided into two main frameworks: the *dependent framework* and the *independent framework.* In the dependent framework, the output of each inducer affects the construction of the next inducer. In this framework knowledge generated in previous iterations guides the learning in the next iteration. In the independent framework each inducer is built independently from other inducers. In the following subsections we provide a brief review of the most popular ensemble methods of both frameworks. Table 1 summarizes the connection between popular methods to their associated framework as well as to relevant training techniques.

4.1 | AdaBoost

AdaBoost (Freund & Schapire, 1995) is the most well-known dependent algorithm for building an ensemble model. The main idea of AdaBoost is to focus on instances that were previously misclassified when training a new inducer. The level of focus given is determined by a weight that is assigned to each instance in the training set. In the first iteration, the same weight is assigned to all of the instances. In each iteration, the weights of misclassified instances are increased, while the weights of correctly classified instances are decreased. In addition, weights are also assigned to the individual base learners based on their overall predictive performance. The pseudocode of the AdaBoost algorithm is provided in Algorithm 1. The algorithm assumes a dataset of n instances, labeled as -1 or $+1$. The classification of a new instance is made by voting on all classifiers M_t, each having a weight of α_t. This iterative procedure provides a series of base learners that complement one another. There are several notable variations of AdaBoost. Soft margin AdaBoost (Rätsch, Onoda, & Müller, 2001) uses regularization in the algorithm to alleviate outlier effects. Modest AdaBoost (Vezhnevets & Vezhnevets, 2005) adds regularization that improves the generalization of the model but impairs training accuracy. SpatialBoost (Avidan, 2006) is a method that incorporates spatial reasoning in the original algorithm. Parallelizing of boosting-based methods like AdaBoost is not straightforward due to its inherent sequential nature. Palit and Reddy (2012) leveraged the MapReduce paradigm to develop two parallel boosting methods: AdaBoost.PL and LogitBoost.PL. These methods require only one cycle of MapReduce. Each mapper runs an AdaBoost algorithm on their own subset of the data to induce the set of inducers. Then, the inducers are sorted and transmitted along with their weights to the reducers. The reducer averages the weights to derive the weights of the final ensemble. A recent work presented boosting limitations in achieving good performance in difficult classification tasks such as speech and image recognition (Cortes, Mohri, & Syed, 2014).

Algorithm 1: The AdaBoost algorithm

Input: I(a weak inducer),T(the number of iterations), S(training set)

Output: M_t,α_t;$\forall t = 1, ..., T$

for *each* t *in* $1, ..., T$ **do**

Build classifier M_t using I and distribution D_t

$\varepsilon_t \longleftarrow \sum_{i:M_t(x_i)\neq y_i} D_t(i)$

if $\varepsilon_t > 0.5$ **then**

$T \longleftarrow t-1$

exit loop

end

else

$\alpha_t \longleftarrow \frac{1}{2} ln(\frac{1-\varepsilon_t}{\varepsilon_t})$

$D_{t+1}(i) = D_t(i) \cdot e^{-\alpha_t y_t M_t(x_i)}$

Normalize D_{t+1} to be a proper distribution

$t++$

end

end

4.2 | Bagging

Bagging (bootstrap aggregating; Breiman, 1996) is a simple yet effective approach for generating an ensemble of independent models in which each inducer is trained using a sample of instances taken from the original dataset as a replacement. In order to ensure a sufficient amount of instances per inducer, each sample usually contains the same number of instances as in the original dataset. Majority voting of the inducers' predictions is performed to determine the final prediction of an unseen instance (Kuncheva, 2004). Since sampling is done with replacements, some of the original instances are likely to appear more than once when training the same inducer while other instances may not be included at all. Since the inducers are independently trained, bagging can easily be implemented in a parallel manner by training each inducer using different computational units. One variation of the bagging algorithm is the improved bagging algorithm (IBA; Jiang, Li, Zheng, & Sun, 2011). In IBA, the resampling process is refined by marking each sample with information entropy. An empirical evaluation of IBA shows that it can improve bagging's generalization in certain machine learning challenges. There are several other popular improvements and variations of the bagging algorithm: The online bagging (and boosting) method (Oza, 2005) preserves bagging's accuracy while running much faster. Wagging (weight aggregation; Bauer & Kohavi, 1999) is a variation that assigns random weights to the instances. In double-bagging two classifiers are trained in each iteration using out-of-bag examples (Hothorn & Lausen, 2003). Bagging has recently been shown to be effective in facial recognition challenges when using extreme learning machines (ELMs) as base learners (Ghimire & Lee, 2014). It is also worth noting that bagging decision trees can be viewed as a particular instance of the random forest technique (Breiman, 2001).

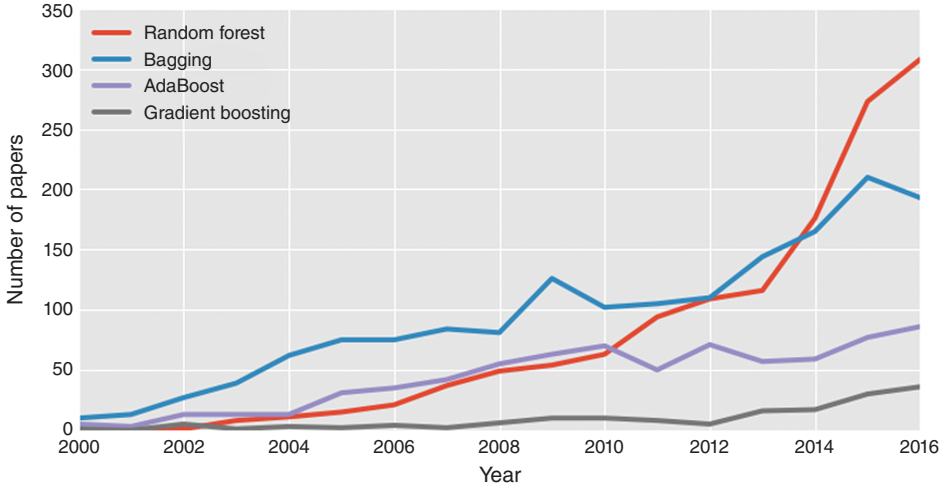

FIGURE 3 Number of published papers per method over time. Based on "web of science" database

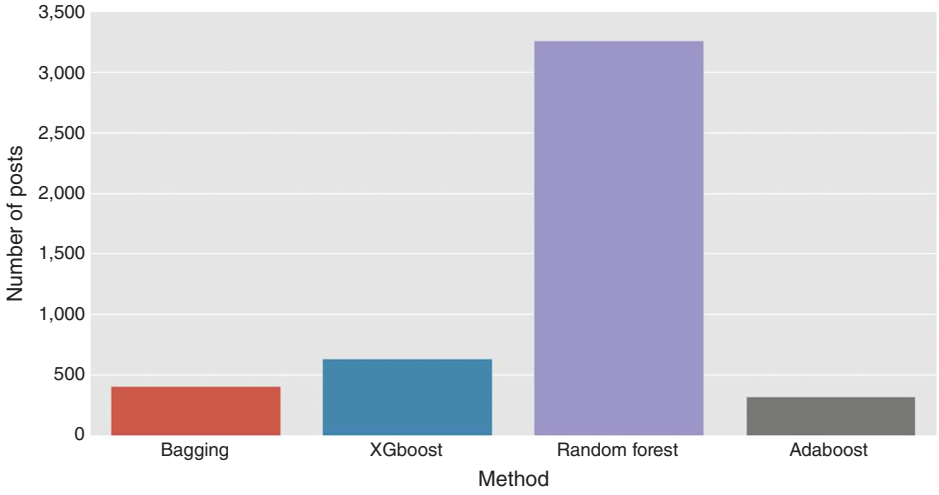

FIGURE 4 Number of technical posts per method between 2014 and 2017. Posts were collected from stackexchange statistical and data science forums. We consider messages that contain the name of the method

4.3 | Random forest and random subspace methods

Probably the most popular ensemble method developed (see Figures 3 and 4), random forest was presented independently by both Ho (1995) and Amit and Geman (1994) at about the same time. A few years later it was elaborated and popularized by Breiman (2001) in a paper that has been cited almost 30,000 times according to Google Scholar (as of June, 2017). Random forest's popularity continues to increase, primarily due to its simplicity and predictive performance. In addition, random forest is considered an easy to tune method compared to other methods (e.g., GBM) that require careful tuning. Random forest uses a large number of independent unpruned decision trees. The individual trees are constructed using the procedure presented in Algorithm 2. The input parameter N represents the number of input variables that are used to determine the decision at a node of the tree. In order to grow a random tree which is still sufficiently accurate, randomness is injected into the decision tree inducers using two randomization processes: (a) training each tree on a different sample of instances and (b) rather than choosing the best split at each node, the inducer randomaly samples a subset of the attributes and chooses the best split among them. The second randomization process can be implemented differently. Instead of selecting the best attribute at each node (using, e.g., an information gain measure), the attribute is selected randomly such that its probability of being selected is proportional to its measured value. A similar idea has been implemented for a randomized C4.5 decision tree (Dietterich, 2000). Instead of selecting the best attribute at each stage, it selects an attribute from the set of the best n features randomly (with equal probability). Random forest also has a real-time version (Saffari, Leistner, Santner, Godec, & Bischof, 2009) which is capable of addressing concept drift in sequential data. Specifically, this version adds a temporal weighting scheme that adaptively excludes some trees based on their errors in diffent time intervals and then grows new trees to replace the excluded ones. The random forest algorithm was originally developed for using decision trees as base learners, largely because of the process of choosing different feature subsets when splitting the nodes. However, this step can easily be replaced by using the broader random subspace method (RSM; Ho, 1998) which can be applied with other types of inducers. There are several other ways to inject randomness into an inducer besides the random forest procedure. First, instead of using all of the instances when selecting a splitting feature, a subsample of the instances can be used (Kamath & Cantu-Paz, 2001). Since the split made at a node is likely to vary with the sample selected, this technique results in different trees which can be combined in ensembles. Kulkarni and Sinha (2013) reviewed different versions of random forest. They presented a specific taxonomy of random forest and compared the existing random forest methods. Recent research found random forest to be more robust and stable than extreme learning machines, SVM, and neural networks, especially with small training sets (Han, Jiang, Zhao, Wang, & Yin, 2018).

Algorithm 2: The Random Forest algorithm

Input: IDT(a decision tree inducer),T(the number of iterations), S(training set), μ(the subsample size), N(number of attributes used in each node)

Output: M_t:$\forall t = 1, ..., T$

for *each* t *in* $1, ..., T$ **do**

$S_t \leftarrow$ Sample μ instances from S with replacement.

Build classifier M_t using $IDT(N)$ on S_t

$t++$

end

4.4 | Gradient boosting machines

In Gradient boosting machines (GBM; Friedman, 2001), the training of each inducer is dependent on inducers that have already been trained. The main difference between GBM and other techniques is that in GBM optimization is applied in the function space. It includes a learning procedure in which the goal is to construct the base learners so that they are maximally correlated with the negative gradient of the loss function, associated with the whole ensemble (Natekin & Knoll, 2013).More specifically, in GBM a sequence of regression trees is computed, where each successive tree predicts the pseudo-residuals of the preceding trees given an arbitrary differentiable loss function. An arbitrary loss function requires the specification of the loss function by the user, in addition to the function that calculates the corresponding negative gradient. Predictions are aggregated in an additive manner in which each added model is trained so it will minimize the loss function. It is important to note that a GBM model usually has many shallow trees, as opposed to random forest which has fewer (but deeper) trees. Choosing the right number of trees (i.e., number of iterations) is very important when training a gradient boosting model. Setting it too high can lead to overfitting, while setting it too low may result in underfitting. The selection of the most suitable number of iterations is usually done by using a validation set to evaluate the overall predictive performance. Overfitting can be reduced by applying a stochastic gradient boosting method (Friedman, 2002) in which trees are consecutively trained with small subsets, sampled from the original dataset. XGBoost (Chen & Guestrin, 2016) is a scalable machine learning system for tree boosting that gained popularity among machine learning practitioners in recent years as depicted in Figure 4. The success of XGBoost in numerous machine learning tasks is reflected in the fact that in 2015, 17 (of 29) Kaggles' winning solutions used XGBoost. XGBoost added several algorithmic optimizations and refinements to GBM in order to increase scalability. First, it features split finding algorithms that (a) handle sparse data with nodes' default directions, (b) address weighted data using merge and prune operations, and (c) enumerate efficiently over all possible splits so the splitting threshold is optimized. Another important refinement in XGBoost is that it adds a regularization component to the loss function presented in GBM, aimed at creating ensembles that are simpler and more generative. Finally, XGBoost can run faster than other models as it supports distributed platforms such as Apache Hadoop and can be distributed across multiple machines. LightGBM is another gradient boosting method recently developed by Microsoft (developed by Microsoft Research by Guolin Ke et al.). It uses histogram based algorithms to reduce duration and memory consumption when training the model. It also leverages network communication algorithms to optimize parallel learning (Zhu et al., 2017).

4.5 | Rotation forest

Rotation forest (Rodriguez, Kuncheva, & Alonso, 2006) is a method that generates diversity among decision tree inducers by training each inducer on the whole dataset in a rotated feature space. When creating the training data for the base classifier, the features are randomly split into K subsets and principal component analysis (PCA) is applied to each subset. The idea of PCA is to orthogonally transform any possible correlated features into a set of linearly uncorrelated features (called principal components). Each component is a linear combination of the original. The transformation guarantees that the first principal component has the largest possible variance. Every subsequent component has the highest variance possible under the constraint that it is orthogonal to the previous components. This transformation, in the context of ensembles, effectively forms new features for each base learner as different feature partitions will lead to different sets of transformed features, and thus different classification trees. In addition to encouraging diversity among the trees, rotating trees also alleviate the decision tree constraint of only splitting the input space into hyperplanes that are parallel to the original feature axes. Despite having been evaluated as outperforming random forest in terms of accuracy, rotation forest has two main drawbacks. First, calculating the PCA at each induction creates computational overhead that does not exist when training random forests. Second, the inducers' interpretability is impaired as the nodes of the obtained trees use transformed features rather than the original features. Schclar and Rokach (2009) suggested using random projections for creating the individual trees rather than using the PCA projection. This reduces the computational complexity burden that arises from the calculation of the principle components. Every derived feature that is used for training the tree is a random linear combination of the original features. Random projection random discretization ensembles (RPRDE) is a rotation forest variation that creates discrete features from continuous features and then provides random linear combinations of the discrete features to each of the inducers (Ahmad & Brown, 2014).

4.6 | Extremely randomized trees

Extremely randomized trees (Geurts, Ernst, & Wehenkel, 2006) is yet another approach for generating diverse ensembles by injecting randomness into the training process. In addition to selecting the best splitting attribute from a random subset of features, splitting features cut-points are also randomized when training extremely randomized trees. Another distinction of extremely randomized Trees is that in contrast to random forest, this method does not use the bootstrap instances procedure.

The same training set is used to induce all of the decision trees. Geurts et al. showed that although extremely randomized trees tend to have high bias and variance components, fusing a sufficiently large forest of trees may eliminate those issues. Several studies from the past few years present extremely randomized Trees effectiveness in the domain of brain tumor segmentation (Gotz et al., 2014; Pinto et al., 2015; Soltaninejad et al., 2017).

4.7 | Ensemble methods and deep neural networks

Deep neural networks (DNN) have become an important force in the machine learning community. In recent years, the use of DNN has dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection, and many other domains (LeCun, Bengio, & Hinton, 2015). DNNs are composed of multiple layers of nonlinear operations. They are able to discover complex structure and learn high-level concepts in large datasets. In recent years there have been a few attempts to combine the ensemble and DNN approaches. Most of them center around developing an ensemble of DNNs. Deep neural decision forests (Kontschieder, Fiterau, Criminisi, & Rota Bulo, 2015) is a learning approach that unifies convolutional neural networks (CNNs) and decision forests techniques. It introduces a stochastic backpropagation version of decision trees that are then combined into decision forests. Another example of an ensemble composed of CNNs has recently been developed for facial expression recognition tasks (Wen et al., 2017). Qiu, Zhang, Ren, Suganthan, and Amaratunga (2014) presented an ensemble of DNNs for regression and time series forecasting. In this ensemble, the outputs of the DNNs construct the input of a support vector regression (SVR) model that produces the final output. Deng and Platt (2014) stacked convolutional, recurrent, and fully connected neural networks to address speech recognition challenges. Experimental results demonstrate a significant increase in phone recognition accuracy. A novel approach that also combines ensemble methods with DNNs, gcForest (Zhou & Feng, 2017) replaces DNN neurons with random forest models where the output vector of each random forest is fed as the input of the next layers, in contrast to the above mentioned methods. In addition, it supports representational learning by applying multigrained scanning when the inputs are of high dimensionality. It is also worthwhile mentioning that the notable dropout technique (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), in which some of the neurons are dropped from the DNN in each iteration, can also be viewed as an ensemble method composed of different neural networks, each with different dropped neurons.

4.8 | Comparison of ensemble methods

During the short history of ensemble learning, many studies been conducted to evaluate the performance of various ensemble methods under different settings. An early attempt (Dietterich, 2000) compared randomizing, bagging and boosting ensembles, consisting of C4.5 decision trees. The experiment showed that boosting provides the best results when there is little noise in the data. Another early study evaluated bagging and boosting performance in different scenarios (Bauer & Kohavi, 1999) and found that bagging reduces the variance of unstable methods, while boosting methods reduced both the bias and variance of unstable methods but increased the variance for stable methods. Other research compared bagging and boosting and found that boosting is much more sensitive to the characteristics of the dataset and may overfit noisy datasets (Opitz & Maclin, 1999). Banfield, Hall, Bowyer, and Kegelmeyer (2007) conducted an experiment with 57 publicly available datasets, comparing bagging with seven other randomization-based techniques. They found that the best method was statistically more accurate than bagging in only eight out of the 57 datasets. However, they also found that random forest and boosting with 1,000 trees had the best average rank among the various algorithms. Fernández-Delgado et al. (2014) compared 179 from 17 different families over 121 datasets. They concluded that random forest methods, and in particular the parallel random forest implemented in R, tend to outperform other learning methods. In addition to predictive performance, there are other factors that may be considered when selecting the best ensemble method for a given problem:

- *Suitability to a given setting*: Methods differ in terms of their capabilities depending on the setting (e.g., class imbalance, high dimensionality, multilabel classification, and noisy data). Therefore, characterizing the learning task and choosing a method accordingly is recommended.
- *Computational cost*: The complexity cost for training the ensemble and the required prediction time for new instances may be considered as important criteria in assessing ensemble models, especially in real-time systems (Bifet et al., 2009).
- *Software availability*: Different platforms provide their own implementations for machine learning models. The ability to use the same algorithm in several applications may be an important attribute for implementation teams.
- *Usability*: Users prefer to understand how to tune the models they use and hence, they prefer models that provide a clear set of controlling parameters.

5 | CONSENSUS CLUSTERING

A direct application of ensemble learning to clustering problems is not trivial since it is not clear how to associate clusters of different clustering systems and due to the unavailability of class information (Weingessel, Dimitriadou, & Hornik, 2003). Therefore, when combining multiple clustering models, the goal is to create a consensus solution that is superior to other clustering systems (Monti, Tamayo, Mesirov, & Golub, 2003; Strehl & Ghosh, 2002; Topchy, Law, Jain, & Fred, 2004). This approach has been shown to improve the stability and the robustness of clustering models (Fred & Jain, 2002)—that is, the resulted clustering systems are less sensitive to outliers and noise, and they usually outperform single clustering models. Clustering ensembles consist of two main stages, similar to those of supervised ensemble models: *generation* and *integration* (Vega-Pons & Ruiz-Shulcloper, 2011). At the generation stage, different clustering models are trained using the given dataset while at the integration stage, the different clusters are mapped into a single consensus clustering. The consensus clustering is a function that maps an ensemble of clustering models into a combined clustering result (Topchy, Law, et al., 2004). In the generation stage, clustering models can be generated using different mechanisms in accordance with the mechanisms for training inducers of supervised ensemble model. For example, we can use different instance or feature subspaces, different clustering algorithms, different hyperparameters, projecting data onto different subspaces, and so on (Ghaemi, Sulaiman, Ibrahim, & Mustapha, 2009). The integration of multiple clustering models into a consensus clustering is considered the hardest part in creating a clustering ensemble (Topchy, Jain, & Punch, 2004). There are several types of consensus clustering approaches. The voting approach uses voting to assign a final cluster for a given instance after solving the label correspondence problem of finding the relation among sets of labels, given by different clustering models (Ayad & Kamel, 2010). Hypergraph based methods transform the integration problem into a hypergraph partitioning problem in which a graph is built from the different clustering models. The vertices in the graph correspond to data instances while hyperedges represent sets of instances that belong to same clusters. Following the creation of the hypergraph, consensus clustering is obtained by solving an optimization problem using different methods (Ben-Hur, Elisseeff, & Guyon, 2001; Karypis, Aggarwal, Kumar, & Shekhar, 1999; Strehl & Ghosh, 2002). In the evidence accumulation framework, cluster assignments are converted into a co-association matrix that describes the frequency of a pair of points that are assigned to the same cluster. Taking this approach, consensus clustering can be gained without encountering the label correspondence problem (Fred & Jain, 2005). A thorough review of other traditional clustering model integration approaches can be found in Ghaemi et al. (2009) and Vega-Pons and Ruiz-Shulcloper (2011) surveys. Bayesian consensus clustering (BCC) is a recently developed approach aims at providing a scalable and robust consensus clustering for multiple data sources by extending Dirichlet mixture models (Lock & Dunson, 2013). Weighted-object ensemble clustering (WOEC) is another recently developed method for generating consensus clustering. This method integrates instance weights when applying graph partitioning. Experiments found that this method results in a clustering that is more stable and robust compared to state-of-the-art methods (Ren, Domeniconi, Zhang, & Yu, 2017). Consensus clustering has been shown to be very effective in discovering biological meaningful clusters in gene expression data (Kiselev et al., 2017; Monti et al., 2003; Verhaak et al., 2010), video shot segmentation (Chang, Lee, Hong, & Archibald, 2008; Zheng, Zhang, & Li, 2012), online event detection (Wu et al., 2017), and cyber security (Liu & Lam, 2012).

6 | MAKING ENSEMBLES SIMPLER AND SMALLER

Despite their high-predictive performance, other models may be preferred over ensemble models for two main reasons. First, predictions usually take a long time for large ensembles as many inducers are applied in order to aggregate a single prediction. This issue appears to be significant in real-time predictive systems (Partalas, Tsoumakas, & Vlahavas, 2008; Tsoumakas, Partalas, & Vlahavas, 2008). Second, it is almost impossible to interpret ensemble outputs as they consist of the outputs of many inducers. This property usually prevents the use of ensemble models in domains that require a clear and rational explanation for individual decisions (e.g., medicine, insurance, etc.; Freitas, 2014). A review of approaches developed in order to mitigate these issues follows.

6.1 | Ensemble pruning methods

Ensemble pruning methods, also known as ensemble selection methods, aim at reducing the complexity of ensemble models. These methods search for a subset of ensemble members that performs at least as well as the original ensemble (Zhang, Burer, & Street, 2006). Once a subset is completed, only the selected trees are kept for the purpose of aggregating inducers' predictions. Finding a small sized ensemble that achieves a performance equivalent to that of a boosted ensemble has been proven to be an NP-hard problem (Tamon & Xiang, 2000). Furthermore, pruning may sacrifice the ensemble generalization

for unseen instances. When Zhou, Wu, and Tang (2002) proved the "many-could-be-better-than-all" theorem, it became clear that it is possible to get a small, yet accurate, ensemble. In addition, pruning can be performed dynamically, depending on a given instances. Taking this approach, only a subset of inducers can be used during the prediction. Hernández-Lobato, Martínez-Muñoz, and Suárez (2009) and Park and Fürnkranz (2012) presented methods that dynamically query just a subset of the inducers, and the output generated is the same as that of the original ensemble. The main idea behind these methods is that the winning class in majority voting can be determined even without considering all of the votes. Martínez-Muñoz, Hernández-Lobato, and Suárez (2009) showed that predictive performance can be reached to maximum even when using only a subset of the ensemble, if chosen properly. Many ensemble pruning methods have been developed over the years. The two most popular approaches for selecting an ensemble subset are ranking and search-based methods.

- *Ranking methods*: Ranking methods rank the individual members based on a predetermined criterion and select the top ranked base models according to a threshold. Caruana, Niculescu-Mizil, Crew, and Ksikes (2004) suggested a forward stepwise selection procedure in order to select the most relevant inducer (in terms of predictive performance) out of numerous inducers. FS-PP-EROS (Hu, Yu, Xie, & Li, 2007) generates a selective ensemble of rough subspaces. This method conducts an accuracy-guided forward search for finding the most relevant members. An empirical evaluation showed that this method outperforms bagging accuracy while providing a smaller ensemble. AB (Bryll et al., 2003) evaluates the performance of randomly selected m-attribute subsets by using the wrapper approach and only the most highly ranked inducers participate in the ensemble. As diversity among the inducers is an important factor for a high-quality ensemble, there are some methods that use diversity for ranking the inducers. Margineantu and Dietterich (1997) proposed a method that compares Kappa-statistics among the inducers. Pairs of models are then selected according to their level of agreement until a desired ensemble size is reached. The percentage correct diversity measure (PCDM; Banfield, Hall, Bowyer, & Kegelmeyer, 2005) focuses on instances that result in ambiguous predictions by the base models when pruning the ensemble. Partalas, Tsoumakas, and Vlahavas (2010) presented a similar approach that uses the uncertainty weighted accuracy (UWA) measure for assessing prediction uncertainty. The collective-agreement-based pruning (CAP; Rokach, 2009) method ranks subsets by considering the individual predictive ability of each decision tree inducer along with the degree of redundancy among the trees. Zhang and Wang (2009) proposed a specific method for random forest pruning that uses three measures that take accuracy and diversity into account. Dai (2013) introduced four evaluation metrics for rank-based pruning that are suitable for the time-series domain.
- *Search-based methods*: Search-based methods conduct a heuristic search in the classifiers' space and evaluate the collective performance of different classifiers' subsets. The GASEN algorithm for example, selects the most suitable inducers in a given ensemble (Zhou et al., 2002). First, GASEN assigns a random weight to each model. Weights are then refined by using genetic algorithms so they would indicate the fitness of the model within the ensemble and remove base models with low weights. GASEN-b (Zhou & Tang, 2003) further develops GASEN by assigning a bit to each classifier, indicating whether it will participate in the final ensemble. Prodromidis and Stolfo (2001) introduced a backward correlation based type of pruning that is trained on base models' outputs and removes members that are less correlated to a meta-classifier. Rokach, Maimon, and Arbel (2006) suggested ranking the models by their ROC performance and evaluating the performance of the ensemble subset using the top ranked members. The size of the subset is gradually increased by adding the top ranked members until there is no improvement in the predictive performance (Rokach et al., 2006). Zhang et al. (2006) formulated the ensemble pruning problem as a quadratic integer programming problem that seeks to find a subset that has the optimal accuracy-diversity trade-off, using a semi-definite programming (SDP) technique. The Pareto ensemble pruning (PEP) approach (Qian, Yu, & Zhou, 2015) uses novel methods of evolutionary optimization for finding a subset that simultaneously increases accuracy and reduces ensemble size. Experiments conducted showed the strength of PEP over state of the art pruning algorithms, and PEP has recently been shown to be effective for detecting malware in Android systems (Fan, Xue, Chen, Xu, & Zhu, 2016). Forest pruning (FP; Jiang, Wu, & Guo, 2017) is a recently developed pruning method that focuses on an ensemble of decision trees in which trees' branches are pruned based on a novel metric called branch importance that indicates the importance of individual branches and nodes of the ensemble.

Search-based methods usually perform better than ranking methods in terms of accuracy, but they are also very computational expensive as they search in a large space. Therefore, one should select a feasible search strategy.

6.2 | Ensemble derived models

Ensemble pruning methods have been shown to significantly improve ensemble predictive performance and computational costs. However, they do not solve the comprehensibility problem and to acquire knowledge from it. Knowledge acquisition

from an ensemble is a subject that has been addressed in some research. A simplified tree ensemble learner (STEL; Deng, 2014) provides the ability to extract rules from an ensemble of trees and search for frequent variable interactions. STEL rules may subsequently be combined into a rule-based classifier that iteratively searches for the matching rule given a new observation. The REEMTIC method combines ensemble base classifiers into a comprehensible set of rules by applying pedagogical and decomposition methods. These rules are then combined and logically minimized by the ESPRESSO method (Iqbal, 2012). The idea of building a single model that preserves the predictive performance of a given ensemble has also been presented in several studies. One approach for tackling this challenge is to use real or generated unlabeled data to train a simple model based on the ensemble predictions. An example is a method that uses an ensemble model to label a vast amount of unlabeled data and then uses the new labeled data to fit a neural network that is smaller and faster than the ensemble model while maintaining its accuracy (Bucilu , Caruana, & Niculescu-Mizil, 2006). If unlabeled data is unavailable, the MUNGE algorithm can be used for generating synthetic unlabeled data that resembles the original dataset. In "born again trees," a new decision tree is built from the ensemble. After training an ensemble with the available data, new unlabeled data is generated by the kernel estimate and then labeled by the ensemble. The final decision tree is trained using the manufactured data labels (Breiman & Shang, 1996). In CMM, a new model is trained by learning the data partitioning implicitly from the ensemble using the generated data (Domingos, 1998). There are also methods that do not require using unlabeled data for training a simpler model. Instead, those approaches conduct a postprocessing procedure that uses the inner structure of the ensemble. One method that works in such manner is ISM (Van Assche & Blockeel, 2007), an algorithm that constructs a single decision tree that iteratively chooses the most informative node for splitting the new tree based on the ensemble structure. This results in a tree that achieves the same performance as the original ensemble for some datasets. GENESIM (Vandewiele, Janssens, Ongenae, De Turck, & Van Hoecke, 2016) is another example of such an algorithm. It applies genetic algorithms for transforming ensemble inducers into a single decision tree. Experiments conducted showing that GENESIM outperforms an optimized single decision tree in terms of generalized accuracy.

7 | CONCLUSIONS AND OPEN RESEARCH QUESTIONS

Ensemble methods imitate human nature by seeking several opinions before making an important decision. The main idea of such methods is to weigh several individual models, and combine them in order to improve predictive performance. This paper has reviewed the major approaches and techniques in the field while discussing the core principles of training ensembles and combining their inducers' outputs. There are several trends and future research directions of ensemble learning that have been discussed in this paper. One trend is to refine popular algorithms to be more efficient and suitable for "Big Data." These efforts usually involved in enabling the distribution of algorithms across multiple machines and the improvement of computational resource utilization. Another promising research direction is the transformation of ensemble models into models that are simpler and more comprehensive while preserving the predictive accuracy of the ensemble they were sourced from. Finally, we presented recent studies that aims at integrating the ensemble paradigm with deep neural networks, in accordance with the increasing pace of deep neural network research.

CONFLICT OF INTEREST

The authors have declared no conflicts of interest for this article.

RELATED WIREs ARTICLES

Classification and regression trees

REFERENCES

Ahmad, A., & Brown, G. (2014). Random projection random discretization ensembles—Ensembles of linear multivariate decision trees. *IEEE Transactions on Knowledge and Data Engineering*, *26*(5), 1225–1239.

Ali, K. M., & Pazzani, M. J. (1995). *On the link between error correlation and error reduction in decision tree ensembles*. Irvine, CA: Information and Computer Science, University of California.

Amit, Y., Geman, D. (1994). *Randomized inquiries about shape: An application to handwritten digit recognition* [DTIC document].

Avidan, S. (2006). Spatialboost: Adding spatial reasoning to adaboost. *Computer Vision—ECCV*, *2006*, 386–396.

Ayad, H. G., & Kamel, M. S. (2010). On voting-based consensus of cluster ensembles. *Pattern Recognition*, *43*(5), 1943–1953.

Banfield, R. E., Hall, L. O., Bowyer, K. W., & Kegelmeyer, W. P. (2005). Ensemble diversity measures and their application to thinning. *Information Fusion*, *6*(1), 49–62.

Banfield, R. E., Hall, L. O., Bowyer, K. W., & Kegelmeyer, W. P. (2007). A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(1), 6–10.

Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, *36*(1), 105–139.

Beemer, J., Spoon, K., He, L., Fan, J., & Levine, R. A. (2017). Ensemble learning for estimating individualized treatment effects in student success studies. *International Journal of Artificial Intelligence in Education*, 1–21. https://doi.org/10.1007/s40593-017-0148-x

Ben-Hur, A., Elisseeff, A., & Guyon, I. (2001). A stability based method for discovering structure in clustered data. In *Proceedings of the Pacific symposium*, Kauai, Hawaii, USA, 3–7 January 2002 (Vol. 7, pp. 6–17).

Bernard, S., Adam, S., & Heutte, L. (2012). Dynamic random forests. *Pattern Recognition Letters*, *33*(12), 1580–1586.

Bi, Y. (2012). The impact of diversity on the accuracy of evidential classifier ensembles. *International Journal of Approximate Reasoning*, *53*(4), 584–607.

Bifet, A., Frank, E., Holmes, G., & Pfahringer, B. (2010). Accurate ensembles for data streams: Combining restricted Hoeffding trees using stacking. In *Proceedings of 2nd Asian conference on machine learning (ACML2010)*, Tokyo, Japan, Nov. 8–10, 2010 (Vol. 13, pp. 225–240).

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 139–148). New York, NY: ACM.

Blaser, R., & Fryzlewicz, P. (2016). Random rotation ensembles. *Journal of Machine Learning Research*, *17*(4), 1–26.

Błaszczyński, J., Stefanowski, J., & Słowiński, R. (2016). Consistency driven feature subspace aggregating for ordinal classification. In *International joint conference on rough sets* (pp. 580–589). Heidelberg: Springer.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., Shang, N. (1996). *Born again trees.* Retrieved from ftp://ftp stat berkeleyedu/pub/users/breiman/BAtreesps

Brown, G., Wyatt, J. L., & Tiňo, P. (2005). Managing diversity in regression ensembles. *Journal of Machine Learning Research*, *6*(9), 1621–1650.

Bryll, R., Gutierrez-Osuna, R., & Quek, F. (2003). Attribute bagging: Improving accuracy of classifier ensembles by using random feature subsets. *Pattern Recognition*, *36*(6), 1291–1302.

Bucilu , C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining* (p. 535). New York, NY, 541: ACM.

Caruana, R., Niculescu-Mizil, A., Crew, G., & Ksikes, A. (2004). Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on machine learning* (p. 18). New York, NY: ACM.

Chan, P. K., & Stolfo, S. J. (1995a). A comparative evaluation of voting and meta-learning on partitioned data. In *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, California, July 9–12, 1995 (pp. 90–98).

Chan, P. K., & Stolfo, S. J. (1995b). Learning arbiter and combiner trees from partitioned data for scaling machine learning. In *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, California, July 9–12, 1995 (Vol. 95, pp. 39–44).

Chang, Y., Lee, D. J., Hong, Y., & Archibald, J. (2008). Unsupervised video shot detection using clustering ensemble with a color global scale-invariant feature transform descriptor. *Journal on Image and Video Processing*, *2008*, 9.

Chawla, N. V., Hall, L. O., Bowyer, K. W., & Kegelmeyer, W. P. (2004). Learning ensembles from bites: A scalable and accurate approach. *Journal of Machine Learning Research*, *5*(4), 421–451.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 785–794). New York, NY: ACM.

Cortes, C., Mohri, M., & Syed, U. (2014). Deep boosting. In *Proceedings of the 31st international conference on machine learning (ICML-14)*, Beijing, China (pp. 1179–1187).

Dai, Q. (2013). A competitive ensemble pruning approach based on cross-validation technique. *Knowledge-Based Systems*, *37*, 394–414.

Dasarathy, B. V., & Sheela, B. V. (1979). A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE*, *67*(5), 708–713.

Deng, H. (2014). *Interpreting tree ensembles with intrees.* arXiv preprint arXiv:14085456.

Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, *239*, 142–153.

Deng, L., & Platt, J. (2014). Ensemble deep learning for speech recognition. In *Proceedings of the 15th Annual Conference of the International Speech Communication Association*, Singapore, September 14–18, 2014.

Derbeko, P., El-Yaniv, R., & Meir, R. (2002). Variance optimized bagging. In *European conference on machine learning* (pp. 60–72). Heidelberg: Springer.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, *40*(2), 139–157.

Dietterich, T. G. (2002). Ensemble learning. In *The handbook of brain theory and neural networks* (Vol. 2, pp. 110–125). Cambridge, MA: MIT Press.

Dietterich, T. G., & Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, *2*, 263–286.

Domingos, P. (1998). Knowledge discovery via multiple models. *Intelligent Data Analysis*, *2*(1–4), 187–202.

Duan, Q., Ajami, N. K., Gao, X., & Sorooshian, S. (2007). Multi-model ensemble hydrologic prediction using Bayesian model averaging. *Advances in Water Resources*, *30*(5), 1371–1386.

Fan, L., Xue, M., Chen, S., Xu, L., & Zhu, H. (2016). Poster: Accuracy vs. time cost: Detecting android malware through Pareto ensemble pruning. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 1748–1750). New York, NY: ACM.

Fernández-Delgado, M., Cernadas, E., Barro, S., & Amorim, D. (2014). Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research*, *15*(1), 3133–3181.

Fred, A. L., & Jain, A. K. (2002). Data clustering using evidence accumulation. In *2002 Proceedings. 16th international conference on pattern recognition* (Vol. 4, pp. 276–280). New York, NY: IEEE.

Fred, A. L., & Jain, A. K. (2005). Combining multiple clusterings using evidence accumulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(6), 835–850.

Freitas, A. A. (2014). Comprehensible classification models: A position paper. *ACM SIGKDD Explorations Newsletter*, *15*(1), 1–10.

Freund, Y., & Schapire, R. E. (1995). A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23–37). Heidelberg: Springer.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, *29*, 1189–1232.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378.

Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics. Part C: Applications and Reviews*, *42*(4), 463–484.

Galar, M., Fernández, A., Barrenechea, E., & Herrera, F. (2013). EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, *46*(12), 3460–3471.

GarcÃa-Pedrajas, N., GarcÃa-Osorio, C., & Fyfe, C. (2007). Nonlinear boosting projections for ensemble construction. *Journal of Machine Learning Research*, *8*(1), 1–33.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42.

Ghaemi, R., Sulaiman, M. N., Ibrahim, H., & Mustapha, N. (2009). A survey: Clustering ensembles techniques. *World Academy of Science, Engineering and Technology*, *50*, 636–645.

Ghimire, D., & Lee, J. (2014). Extreme learning machine ensemble using bagging for facial expression recognition. *JIPS*, *10*(3), 443–458.

Gomes, H. M., Barddal, J. P., Enembreck, F., & Bifet, A. (2017). A survey on ensemble learning for data stream classification. *ACM Computing Surveys (CSUR)*, *50*(2), 23.

Gotz, M., Weber, C., Blocher, J., Stieltjes, B., Meinzer, H. P., & Maier-Hein, K. (2014). Extremely randomized trees based brain tumor segmentation. In *Proceedings of BRATS challenge-MICCAI*, Boston.

Han, T., Jiang, D., Zhao, Q., Wang, L., & Yin, K. (2018). Comparison of random forest, artificial neural networks and support vector machine for intelligent diagnosis of rotating machinery. *Transactions of the Institute of Measurement and Control*. https://doi.org/10.1177/0142331217708242

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(10), 993–1001.

Haque, M. N., Noman, M. N., Berretta, R., & Moscato, P. (2016). Optimising weights for heterogeneous ensemble of classifiers with differential evolution. In *2016 I. E. Congress on Evolutionary Computation (CEC)* (pp. 233–240). Washington, DC: IEEE.

Hernández-Lobato, D., Martínez-Muñoz, G., & Suárez, A. (2009). Statistical instance-based pruning in ensembles of independent classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(2), 364–369.

Ho, T. K. (1995). Random decision forests. In *1995 Proceedings of the third international conference on document analysis and recognition* (Vol. 1, pp. 278–282). New York, NY: IEEE.

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(8), 832–844.

Hothorn, T., & Lausen, B. (2003). Double-bagging: Combining classifiers by bootstrap aggregation. *Pattern Recognition*, *36*(6), 1303–1309.

Hu, Q., Yu, D., Xie, Z., & Li, X. (2007). EROS: Ensemble rough subspaces. *Pattern Recognition*, *40*(12), 3728–3739.

Huang, J., Fang, H., & Fan, X. (2010). Decision forest for classification of gene expression data. *Computers in Biology and Medicine*, *40*(8), 698–704.

Idrees, F., Rajarajan, M., Conti, M., Chen, T. M., & Rahulamathavan, Y. (2017). PIndroid: A novel Android malware detection system using ensemble learning methods. *Computers & Security*, *68*, 36–46.

Iqbal, M. (2012). Rule extraction from ensemble methods using aggregated decision trees. In *Neural information processing* (pp. 599–607). Heidelberg: Springer.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, *3*(1), 79–87.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, *6*(5), 429–449.

Jiang, T., Li, J., Zheng, Y., & Sun, C. (2011). Improved bagging algorithm for pattern recognition in UHF signals of partial discharges. *Energies*, *4*(7), 1087–1101.

Jiang, X., Wu, C., & Guo, H. (2017). Forest pruning based on branch importance. *Computational Intelligence and Neuroscience*, *2017*, 1–11.

Kamath, C., & Cantu-Paz, E. (2001). *Creating ensembles of decision trees through sampling*. Livermore, CA: Lawrence Livermore National Laboratory.

Karypis, G., Aggarwal, R., Kumar, V., & Shekhar, S. (1999). Multilevel hypergraph partitioning: Applications in VLSI domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *7*(1), 69–79.

Kiselev, V. Y., Kirschner, K., Schaub, M. T., Andrews, T., Yiu, A., Chandra, T., … Hemberg, M. (2017). SC3: Consensus clustering of single-cell RNA-seq data. *Nature Methods*, *14*, 483–486.

Kolter, J. Z., & Maloof, M. A. (2003). Dynamic weighted majority: A new ensemble method for tracking concept drift. In *ICDM 2003: 2003 Third IEEE International Conference on Data Mining* (pp. 123–130). Washington, DC: IEEE.

Kontschieder, P., Fiterau, M., Criminisi, A., & Rota Bulo, S. (2015). Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, Santiago, Chile (pp. 1467–1475).

Kulkarni, V. Y., & Sinha, P. K. (2013). Random forest classifiers: A survey and future research directions. *International Journal of Advanced Computer Technology*, *36*(1), 1144–1153.

Kumar, V., & Minz, S. (2016). Multi-view ensemble learning: An optimal feature set partitioning for high-dimensional data classification. *Knowledge and Information Systems*, *49*(1), 1–59.

Kuncheva, L. I. (2004). *Combining pattern classifiers: Methods and algorithms*. New York, NY: John Wiley & Sons.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Lin, S. W., & Chen, S. C. (2012). Parameter determination and feature selection for C4.5 algorithm using scatter search approach. *Soft Computing*, *16*(1), 63–75.

Liu, A. Y., & Lam, D. N. (2012). Using consensus clustering for multi-view anomaly detection. In *2012 I.E. symposium on security and privacy workshops (SPW)* (pp. 117–124). San Francisco, CA: IEEE.

Lock, E. F., & Dunson, D. B. (2013). Bayesian consensus clustering. *Bioinformatics*, *29*(20), 2610–2616.

Mao, S., Jiao, L., Xiong, L., Gou, S., Chen, B., & Yeung, S. K. (2015). Weighted classifier ensemble based on quadratic form. *Pattern Recognition*, *48*(5), 1688–1706.

Margineantu, D. D., & Dietterich, T. G. (1997). Pruning adaptive boosting. In *ICML*, Nashville, TN, USA (Vol. 97, pp. 211–218).

Martínez-Muñoz, G., Hernández-Lobato, D., & Suárez, A. (2009). An analysis of ensemble pruning techniques based on ordered aggregation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(2), 245–259.

Masoudnia, S., & Ebrahimpour, R. (2014). Mixture of experts: A literature survey. *Artificial Intelligence Review*, *42*, 1–19.

Maudes, J., Rodríguez, J. J., & García-Osorio, C. (2009). Disturbing neighbors diversity for decision forests. In *Applications of supervised and unsupervised ensemble methods* (pp. 113–133). Heidelberg: Springer.

Menahem, E., Rokach, L., & Elovici, Y. (2009). Troika—An improved stacking schema for classification tasks. *Information Sciences*, *179*(24), 4097–4122.

Mendes-Moreira, J., Soares, C., Jorge, A. M., & Sousa, J. F. D. (2012). Ensemble approaches for regression: A survey. *ACM Computing Surveys (CSUR)*, *45*(1), 10.

Minku, L. L., White, A. P., & Yao, X. (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *22*(5), 730–742.

Minku, L. L., & Yao, X. (2012). DDD: A new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *24*(4), 619–633.

Monti, S., Tamayo, P., Mesirov, J., & Golub, T. (2003). Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, *52*(1), 91–118.

Natekin, A., & Knoll, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *7*, 21.

Nikulin, V., GJ, M. L., & Ng, S. K. (2009). Ensemble approach for the classification of imbalanced data. In *AI 2009: Advances in artificial intelligence* (pp. 1–20). Heidelberg: Springer-Verlag.

Omari, A., & Figueiras-Vidal, A. R. (2015). Post-aggregation of classifier ensembles. *Information Fusion*, *26*, 96–102.

Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, *11*, 169–198.

Opitz, D. W., & Shavlik, J. W. (1996). Actively searching for an effective neural network ensemble. *Connection Science*, *8*(3–4), 337–354.

Oza, N. C. (2005). Online bagging and boosting. In *2005 I.E. international conference on systems, man and cybernetics* (Vol. 3, pp. 2340–2345). Waikoloa, HI: IEEE.

Oza, N. C., & Tumer, K. (2008). Classifier ensembles: Select real-world applications. *Information Fusion*, *9*(1), 4–20.

Paisitkriangkrai, S., Shen, C., & van den Hengel, A. (2016). Pedestrian detection with spatially pooled features and structured ensemble learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(6), 1243–1257.

Palit, I., & Reddy, C. K. (2012). Scalable and parallel boosting with mapreduce. *IEEE Transactions on Knowledge and Data Engineering.*, *24*(10), 1904–1916.

Park, S. H., & Fürnkranz, J. (2012). Efficient prediction algorithms for binary decomposition techniques. *Data Mining and Knowledge Discovery*, *24*(1), 40–77.

Partalas, I., Tsoumakas, G., & Vlahavas, I. (2010). An ensemble uncertainty aware measure for directed hill climbing ensemble pruning. *Machine Learning*, *81*(3), 257–282.

Partalas, I., Tsoumakas, G., & Vlahavas, I. P. (2008). Focused ensemble selection: A diversity-based method for greedy ensemble selection. In *Proceedings of 18th European Conference on Artificial Intelligence*, Amsterdam, The Netherlands (pp. 117–121).

Pinto, A., Pereira, S., Correia, H., Oliveira, J., Rasteiro, D. M., & Silva, C. A. (2015). Brain tumour segmentation based on extremely randomized forest with high-level features. In *2015 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC)* (pp. 3037–3040). Milan, Italy: IEEE.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, *6*(3), 21–45.

Prodromidis, A. L., & Stolfo, S. J. (2001). Cost complexity-based pruning of ensemble classifiers. *Knowledge and Information Systems*, *3*(4), 449–469.

Qian, C., Yu, Y., & Zhou, Z. H. (2015). Pareto ensemble pruning. In *AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, Austin, Texas (pp. 2935–2941).

Qiu, X., Zhang, L., Ren, Y., Suganthan, P. N., & Amaratunga, G. (2014). Ensemble deep learning for regression and time series forecasting. In *2014 I.E. symposium on computational intelligence in ensemble learning (CIEL)* (pp. 1–6). Orlando, FL: IEEE.

Rätsch, G., Onoda, T., & Müller, K. R. (2001). Soft margins for AdaBoost. *Machine Learning*, *42*(3), 287–320.

Ren, Y., Domeniconi, C., Zhang, G., & Yu, G. (2017). Weighted-object ensemble clustering: Methods and analysis. *Knowledge and Information Systems*, *51*(2), 661–689.

Ren, Y., Zhang, L., & Suganthan, P. N. (2016). Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational Intelligence Magazine*, *11*(1), 41–53.

Rodriguez, J. J., Kuncheva, L. I., & Alonso, C. J. (2006). Rotation forest: A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1619–1630.

Rokach, L. (2008). Genetic algorithm-based feature set partitioning for classification problems. *Pattern Recognition*, *41*(5), 1676–1700.

Rokach, L. (2009). Collective-agreement-based pruning of ensembles. *Computational Statistics & Data Analysis*, *53*(4), 1015–1026.

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review*, *33*(1), 1–39.

Rokach, L. (2016). Decision forest: Twenty years of research. *Information Fusion*, *27*, 111–125.

Rokach, L., Maimon, O., & Arbel, R. (2006). Selective voting—Getting more for less in sensor fusion. *International Journal of Pattern Recognition and Artificial Intelligence*, *20*(3), 329–350.

Saffari, A., Leistner, C., Santner, J., Godec, M., & Bischof, H. (2009). On-line random forests. In *2009 I.E. 12th international conference on computer vision workshops (ICCV workshops)* (pp. 1393–1400). New York, NY: IEEE.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227.

Schapire, R. E., & Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the 11th annual conference on computational learning theory* (pp. 80–91). New York, NY: ACM.

Schclar, A., & Rokach, L. (2009). Random projection ensemble classifiers. *Enterprise Information Systems*, *24*, 309–316.

Seewald, A. K., & Fürnkranz, J. (2001). An evaluation of grading classifiers. In *International symposium on intelligent data analysis* (pp. 115–124). Heidelberg: Springer.

Shieh, A. D., & Kamm, D. F. (2009). Ensembles of one class support vector machines. In *International workshop on multiple classifier systems* (pp. 181–190). Heidelberg: Springer.

Soltaninejad, M., Yang, G., Lambrou, T., Allinson, N., Jones, T. L., Barrick, T. R., … Ye, X. (2017). Automated brain tumour detection and segmentation using superpixel-based extremely randomized trees in FLAIR MRI. *International journal of computer assisted radiology and surgery.*, *12*(2), 183–203.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Strehl, A., & Ghosh, J. (2002). Cluster ensembles—A knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research*, *3*(12), 583–617.

Tamon, C., & Xiang, J. (2000). On the boosting pruning problem. In *European conference on machine learning* (pp. 404–412). Heidelberg: Springer.

Ting, K. M., Wells, J. R., Tan, S. C., Teng, S. W., & Webb, G. I. (2011). Feature-subspace aggregating: Ensembles for stable and unstable learners. *Machine Learning*, *82*(3), 375–397.

Topchy, A., Jain, A. K., & Punch, W. (2004). A mixture model for clustering ensembles. In *Proceedings of the 2004 SIAM international conference on data mining* (pp. 379–390). Philadelphia, PA: SIAM.

Topchy, A. P., Law, M. H., Jain, A. K., & Fred, A. L. (2004). Analysis of consensus partition in cluster ensemble. In *2004. ICDM'04. Fourth IEEE international conference on data mining* (pp. 225–232). New York, NY: IEEE.

Tsoumakas, G., Partalas, I., & Vlahavas, I. (2008). A taxonomy and short review of ensemble selection. In *ECAI 2008, workshop on supervised and unsupervised ensemble methods and their applications*, Patras, Greece, 2008 (pp. 41–46).

Tukey, J. W. (1977). *Exploratory data analysis*. Reading, MA: Addison-Wesley.

Van Assche, A., & Blockeel, H. (2007). Seeing the forest through the trees: Learning a comprehensible model from an ensemble. In *ECML* (Vol. 7, pp. 418–429). Heidelberg: Springer.

Vandewiele, G., Janssens, O., Ongenae, F., De Turck, F., Van Hoecke, S. (2016). *GENESIM: Genetic extraction of a single, interpretable model.* arXiv preprint arXiv:161105722.2016.

Vega-Pons, S., & Ruiz-Shulcloper, J. (2011). A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, *25*(03), 337–372.

Verhaak, R. G., Hoadley, K. A., Purdom, E., Wang, V., Qi, Y., Wilkerson, M. D., … Cancer Genome Atlas Research Network. (2010). Integrated genomic analysis identifies clinically relevant subtypes of glioblastoma characterized by abnormalities in PDGFRA, IDH1, EGFR, and NF1. *Cancer Cell*, *17*(1), 98–110.

Vezhnevets, A., & Vezhnevets, V. (2005). Modest AdaBoost-teaching AdaBoost to generalize better. In Computer Graphics and Applications (GraphiCon'2005) Fifteenth International Conference, Novosibirsk Akademgorodok, Russia, June 20–24, 2005 (Vol. 12, pp. 987–997).

Weingessel, A., Dimitriadou, E., & Hornik, K. (2003). An ensemble method for clustering. In *Proceedings of the 3rd international workshop on distributed statistical computing*, Vienna.

Wen, G., Hou, Z., Li, H., Li, D., Jiang, L., & Xun, E. (2017). Ensemble of deep neural networks with probability-based fusion for facial expression recognition. *Cognitive Computation*, *9*, 1–14.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.

Woźniak, M., Graña, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, *16*, 3–17.

Wu, J., Wu, Z., Cao, J., Liu, H., Chen, G., & Zhang, Y. (2017). Fuzzy consensus clustering with applications on Big data. *IEEE Transactions on Fuzzy Systems*, *25*, 1430–1445.

Zhang, C. X., & Zhang, J. S. (2008). RotBoost: A technique for combining rotation forest and AdaBoost. *Pattern Recognition Letters*, *29*(10), 1524–1536.

Zhang, H., & Wang, M. (2009). Search for the smallest random forest. *Statistics and Its Interface*, *2*(3), 381.

Zhang, Y., Burer, S., & Street, W. N. (2006). Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, *7*(7), 1315–1338.

Zheng, Y. T., Zhang, Y. D., Li J. T. (2012). *Ensemble learning with LDA topic models for visual concept detection.* Multimedia-A Multidisciplinary Approach to Complex Issues. InTech, 2012.

Zhong, G., & Cheriet, M. (2013). Adaptive error-correcting output codes. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, Beijing, China (pp. 1932–1938).

Zhou, J. T., Tsang, I. W., Ho, S. S., Muller, K. R. (2016). *N-ary error correcting coding scheme.* arXiv preprint arXiv:160305850.

Zhou, Z. H., Feng, J. (2017). *Deep forest: Towards an alternative to deep neural networks.* arXiv preprint arXiv:170208835.

Zhou, Z. H., & Tang, W. (2003). Selective ensemble of decision trees. In *Rough sets, fuzzy sets, data mining, and granular computing*, Chongqing, China (p. 589).

Zhou, Z. H., Wu, J., & Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, *137*(1–2), 239–263.

Zhu, J., Shan, Y., Mao, J., Yu, D., Rahmanian, H., Zhang, Y. (2017). *Deep embedding forest: Forest-based serving with deep embedding features.* arXiv preprint arXiv:170305291x.

How to cite this article: Sagi O, Rokach L. Ensemble learning: A survey. *WIREs Data Mining Knowl Discov*. 2018; 8:e1249. https://doi.org/10.1002/widm.1249