Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Some features speak loud, but together they all speak louder: A study on the correlation between classification error and feature usage in decision-tree classification ensembles

Bárbara Cervantes[a], Raúl Monroy[a,*], Miguel Angel Medina-Pérez[a], Miguel Gonzalez-Mendoza[a], Jose Ramirez-Marquez[b]

[a] *Tecnologico de Monterrey, Campus Estado de México, Carretera al Lago de Guadalupe Km. 3.5, Atizapán de Zaragoza, Estado de México, 52926, Mexico*
[b] *Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ, 07030, United States*

A R T I C L E   I N F O

Keywords:
Decision tree ensemble

A B S T R A C T

While diversity has been argued to be the rationale for the success of an ensemble of classifiers, little has been said on how uniform use of the feature space influences classification error. Following an observation from a recent result, published elsewhere, among several ensembles of decision trees, those with a more uniform feature-use frequency also have a smaller classification error. This paper provides further support to such hypothesis. We have conducted experiments over 60 classification datasets, using 42 different types of decision tree ensembles, to test our hypothesis. Our results validate the hypothesis, prompting the design of ensemble construction methods that make a more uniform use of features, for classification problems of low and medium dimensionality.

1. Introduction

Ensemble classifiers combine the output of several classifiers; the errors in one classifier are outweighed by the success in others leading to a better decision. They rely on diversity in order to complement errors and increase performance. So, a desirable property of an ensemble is diversity, which is introduced by one or more of the following: (i) changing the number and type of the base classifiers; (ii) using different abstractions of the training dataset; (iii) using different approaches in the training of the base classifiers; and (iv) using each classifier in the ensemble to solve a different classification task.

While diversity has been largely studied to account for improvement on classifier performance, little has been said about how uniform use of the feature space influences classification error. In this paper, we have explored the relationship between feature usage and classification error in ensembles based on decision trees, applied to low and medium dimensionality problems (i.e. classification problems with less than 100 features). Our work stems from a recent result, (Camiña et al., 2016), where the authors noticed that, among four decision tree ensembles, there was an apparent correlation between uniformity of feature-use frequency and classification error. The ensembles were used to solve the problem of masquerade detection in the WUIL dataset repository (Camiña et al., 2016). As part of these experiments, feature-use frequency was computed with the expectation it would shed light on the most relevant features for classification. Instead, authors noticed that those ensembles with a more uniform use of the feature space had a smaller classification error. However, this remained merely as an observation, as it was out of the scope of Camiña et al. (2016).

In this paper, we have turned this observation into our working hypothesis and successfully validated it using an experimental approach. To show that the correlation observed in Camiña et al. (2016) is not limited to a few types of ensembles, we included more ensemble variants in our experiments. We present a taxonomy, which is based on the sources of diversity in ensembles; by changing some of the parameters that describe the ensemble between a predefined a set of values, we obtain 42 different variants. To show that the hypothesis is not only valid in one domain, we extended the experiment to include 60 datasets from different domains and with a variety of characteristics. These datasets solve binary and multi-class problems; they are of different sizes and have a dimensionality of less than one hundred features.

Our statistical analysis confirms that, in 56% of the cases, a more uniform use of the feature space implies a smaller classification error, when using a decision-tree based classification ensemble. We discuss how the validity of the hypothesis is beneficial for ensemble creation

[*] Corresponding author.
E-mail address: raulm@itesm.mx (R. Monroy).

https://doi.org/10.1016/j.engappai.2017.10.007
Received 3 February 2017; Received in revised form 5 August 2017; Accepted 4 October 2017
Available online 5 November 2017
0952-1976/© 2017 Elsevier Ltd. All rights reserved.

methods. Our results indicate that one should strive towards creating an ensemble of decision tree classifiers that drives its construction towards making a more uniform use of features, in a low or medium dimensionality classification problem. There is not a single or obvious approach to achieve this, many paths are open to be explored; we point out some of them. However, the design of an algorithm for the creation of ensembles which takes advantage of the hypothesis is beyond the scope of this paper.

The main contributions of this paper is the successful validation of our hypothesis. Although the hypothesis might sound simple, the existence of the link between uniformity in the feature use frequency and classification error opens a new avenue for researchers to explore and exploit this relationship with the goal of creating more accurate ensemble classifiers. Usually researchers are encouraged to reduce the dimensionality of datasets; we have found that for decision tree ensembles that solve classification problems of less than one hundred features, in fact, keeping all the features, and using them uniformly, favours the odds of having a more accurate performance. A complementary contribution, is the presentation of a taxonomy to describe with detail the sources of diversity in an ensemble.

The rest of the paper is organized as follows. Section 2 presents an overview of the decision tree ensemble methods, the idea of diversity and its relation to a classifier's success. Section 3 provides an overview of the construction of decision tree ensembles; a reader familiarized with decision tree classifiers and ensembles may skip this section. Sections 4–6 are the core of the paper. Section 4 formally describes a key concept, feature-use frequency variance, which is used to measure how uniform is the use of the feature space by a given decision tree based classification ensemble. Section 5 enumerates the classification ensembles and the datasets used throughout our experiments. It also presents an outline of our experimentation protocol. Then, Section 6 presents the statistical analysis we performed to validate experimentally our working hypothesis. Finally, in Section 7 we add our final remarks.

2. Related work

The domains on which ensemble classifiers have been used are broad, the work by Woźniak et al. (2014) reviews recent applications in remote sensing, finance, computer security, recommender systems, and medicine. They have proven to be useful for problems with imbalanced classes (Krawczyk et al., 2014a; Loyola-González et al., 2017). We focus exclusively on ensembles based on decision tree classifiers. These classifiers are considered unstable because a small variation in a split may change the whole subtree below that node. The instability of decision trees, which is a disadvantage for a single classifier, becomes an advantage and it is exploited by ensemble methods to generate diversity. In addition, their low computational cost makes them suitable for an ensemble approach as many decision trees need to be trained to construct an ensemble. Some instances of problems solved with decision tree ensemble classifiers are classification of tree genera and species (Ko et al., 2016), landslide susceptibility mapping (Bui et al., 2014), predicting tryptic cleavage (Fannes et al., 2013), and monitoring cardiac complications of diabetes (Kelarev et al., 2013).

Bagging (Breiman, 1996) and boosting (Freund and Schapire, 1995) were the first popular ensemble methods based on decision trees; the most representative boosting method being AdaBoost (Freund and Schapire, 1996). Both methods generate different classifiers by changing the training datasets for each tree that is generated. In bagging, trees are trained with a different sample of the training set each time. To classify a new object, tree results are aggregated using a plurality voting system. Boosting is similar as bagging but it assigns weights to objects in order to make each new classifier focus on the objects that are difficult to classify. To classify a new object tree results are aggregated according to each tree associated weight, which is computed after building each tree. Both bagging and boosting train with a dataset that contains the whole feature set, however, this does not mean that all features will be used in the trees that are part of the ensemble. Even more, how features are used in the trees has not been a subject of attention. Breiman points out that bagging can decrease classification error (compared to a single decision tree) because of the instability of the classifiers with respect to the training dataset (Breiman, 1996). This means that the trees created will be different and the aggregation of them is what brings value to the ensemble. Boosting attributes its success over bagging on its ability to force the trees to concentrate on the objects that have been classified incorrectly by previous trees.

Later, methods like Random Subspace (Ho, 1998) and Random Forests (Breiman, 2001) started to include feature selection as part of their building process. The Random Subspace method introduced by Ho (Ho, 1998) constructs each tree based on a random subset of features, while Random Forest combines bagging and random feature selection (Breiman, 2001). Like the Random Subspace method, feature selection in Random Forest is done by taking a random subset of features, except here, the selection is performed at a node level. Feature selection in the Random Subspace method is done under the assumption that if the subspace changes within the trees there will be more different trees. Ho finds that her algorithm is more accurate than bagging and boosting for datasets with many features and objects. She attributed the classifier's success to the fact that its trees are less similar. In this case, the similarity is measured as tree agreement, a measure of when the classifiers predict the same class for an object, see (Ho, 1998) for details. Random Forest results show that its performance as good as AdaBoost and sometimes better; and that the classification error decreases in a forest when the correlation between trees is minimized while the strength of the trees is maintained. In this case, the strength and correlation are measured in terms of a margin function, which determines how confident is each tree prediction, see (Breiman, 2001) for details.

Experiments in both Random Subspace (Ho, 1998) and Random Forests (Breiman, 2001) look at the effect of the size of the feature subset but, again, no much attention is given to how the features are used in the tree. Random Subspace experiments have the purpose of gaining knowledge about how many features should be included in the subset, but there is no information about whether or not all features were used across the ensemble, and how frequently. Breiman performs a similar experiment for Random Forests; he concludes that bagging seems to reduce the classification error when feature selection is part of the algorithm. While building a decision tree, some features tend to be used more because they are positively evaluated by the split function. Intuition tells us that since the subset of random features might not contain these features, methods which include feature selection allow more features, that would be otherwise overlooked, to be part of the ensemble. This might be the reason bagging reduces the classification error, it may be the case that positively evaluated features in one training dataset are poorly evaluated in another, giving chance to other features to be selected. However, this was not measured in Breiman (2001), so our idea remains an intuition and needs to be confirmed. Additionally, Breiman performs another experiment, on the diabetes dataset, to determine feature importance in Random Forests (Breiman, 2001). Feature importance is seen as the contribution of the feature to reduce the classification error. As part of this experiment, after determining feature importance, an ensemble is built using only the most important feature. Then another using the two most important features, and so on. We can observe in the results that for this dataset adding more features causes a decrease in the classification error. If we look at the error of the ensemble that uses all features we see that, in fact, for the diabetes dataset, together all features speak louder.

As we can observe, the methods for creating ensembles aim to build different trees to increase diversity, assuming as a consequence the error of the ensemble will decrease. There have been several attempts to link diversity with classification error, however, the details of this relationship are still unclear. One of the problems faced is that there is no consensus of what counts as diversity. There are many diversity measures. We already mentioned two of them: tree agreement and tree correlation. It is not our intention to list all diversity measures but to provide an overview of the insights we have about diversity; interested readers are referred to (Kuncheva and Whitaker, 2003) for a discussion about diversity measures.

A study by Cunningham and Carney compares diversity and quality of classification ensembles (Cunningham and Carney, 2000). Although it is not limited to ensembles based on decision tree classifiers, the authors centre their discussion on the Random Subspace method (Ho, 1998) and a modification of it that uses a genetic algorithm to select features (Guerra-Salcedo and Whitley, 2000). Guided feature selection outperforms random feature selection when the number of features in the dataset is large otherwise diversity is lost, and Random Subspaces are better (Guerra-Salcedo and Whitley, 2000). Additionally, they found that if the data used to drive the feature selection is small, feature subsets start to overfit the data (Cunningham and Carney, 2000). Cunningham and Carney argue that classification ensembles should consider diversity to guide decisions in training and proposed an entropy-based diversity measure which captures the importance of small error spread across all classes (Cunningham and Carney, 2000).

Kuncheva addressed this topic in Kuncheva and Whitaker (2003) considering ten different measures of diversity. She found that even when there is no consensus about how to measure diversity, all the measures correlate, indicating an agreement in the general idea of diversity. She also found that there is a threshold for diversity which guarantees improvement for ensembles (with diversity higher than the threshold) over the best individual classifier. However, there is no evidence of a connection between the level of diversity and the error decrease. In a later work (Kuncheva, 2014), Kuncheva has brought to attention that diversity and independence are not synonyms and that independent classifiers are not the best scenario. An ensemble of independent classifiers achieves a lower error than an individual classifier but it is the dependency among the classifiers in the ensemble what may greatly decrease or increase the error of the ensemble. Kuncheva defines two types of diversity: "good diversity helps to achieve correct majority with the minimum number of votes while bad diversity wastes the maximum number of correct votes without reaching majority" (Kuncheva, 2014). Kuncheva concludes that diversity is important but it may or may not decrease ensemble error and that methods that implicitly increase diversity have performed well while explicitly considering it in the building process has had limited success.

Our belief is that there is another ensemble property which could guide the building process of a decision tree ensemble. This property is how fair the algorithm is at using all the features in the dataset. This does not mean that diversity should be disregarded, on the contrary, ensembles should aim to be diverse. Uniformity of feature usage is proposed as complement measure to look for in ensembles. In this paper, we have measured how uniform is the use of the features in different decision tree ensemble strategies and how it correlates with their error. We found that, in 56% of the cases, out of two algorithms the one that uses features more uniformly is the one with lower error. As with diversity, algorithms that implicitly increase uniformity on feature usage, tend to perform better. It remains to be seen if tailoring algorithms to explicitly favour this measure can create a more accurate classifiers.

3. Decision tree ensembles, an overview

In this section, we detail the approach taken in the construction of the ensembles. It is divided in three subsections. In Section 3.1, we introduce a taxonomy that describes the construction of ensembles. We use this taxonomy to specify the variations in each of the ensembles used in our experiments. Section 3.2 covers the source of variation coming from the decision tree classifiers themselves. Decision tree classifiers can be altered in many ways. Here, we limit the section to cover only the parameters that were changed in the experiments: split measure and pruning. However, our taxonomy allows the specification of a broader range of variations. Finally, in Section 3.3, we use the taxonomy to describe the six algorithms that were used in our experiments.

3.1. A taxonomy for the construction of ensembles

The methods to construct ensembles aim to incorporate diversity among the ensemble's classifiers with the intention of getting a broader perspective of the problem that leads to a more accurate classification. In Section 1 we presented four routes that have been used among the research community to introduce diversity. Here we describe them with more detail:

1. Changing the number and type of the base classifiers (Kuncheva, 2014): every ensemble is composed of a fixed number of classifiers. We can choose to combine different types of classifiers, in different proportions, or to have them all of the same type. For example, we could create an ensemble where all of the classifiers are multi-layer perceptrons or we could create one where half of the classifiers are decision trees and the other half are support vector machines, the possibilities are vast, as many as the number of combinations possible.
2. Using different abstractions of the training dataset: this includes resampling the dataset (Kuncheva, 2014), partitioning it (Kuncheva, 2014), or using new features derived from de original ones. Partitioning can be either vertical or horizontal, or both: a vertical partition takes all the objects in the training set but only some of the features; conversely, a horizontal partition takes all of the features but only some of the objects in the training set. A different abstraction of the training set comes from derived features when instead of using the raw features, we perform some processing before the classification task; for example, to indicate a position information, instead of working with two coordinates we could compute the distance to a point of reference and use this distance as a feature for the classifier.
3. Using different approaches in the training of the base classifiers (Kuncheva, 2014): this includes changing something in the algorithm that trains the classifier. For example, changing k, or the distance function, in a KNN classifier.
4. Using each classifier in the ensemble to solve a different classification task (Kuncheva, 2014): in this case, each classifier in the ensemble may discriminate between different classes. For example, in a multiclass problem, one classifier in the ensemble may solve a dichotomy, while other solves a different dichotomy, etcetera.

From these, we developed a taxonomy that defines the construction strategy of an ensemble with a tuple of four elements:

1. Number and type of classifiers.
2. Training dataset.
3. Parameters or techniques used in training.
4. Problem decomposition details.

The first element is a list of key–value pairs that state the number and type of the classifiers in the ensemble. For example, [KNN:3, SVM:3, C45:4] describes an ensemble of ten classifiers of which three are K Nearest Neighbours, three are Support Vector Machines and four are C4.5 Trees.

The second element is a set of statements that describe how the original training set must be processed in order to get the training set for each of the classifiers. Here we state if any sampling is performed (specifying size and sampling distribution), if any partitioning is performed (specifying size and type), and if there is any processing to be performed in the features (specifying map function from original features to training features). For simplicity, we do not include the information when the original form is used. For example, let I be the original training dataset; if the second element of the tuple is: {partition(I,50,horizontal)}, it means that each classifier of the ensemble will be trained on a different horizontal partition of I, of size 50.

The third element content depends on the type of classifiers in the ensemble. It includes all the specifications of the parameters or

Table 1
Symbols and description.

Symbol	Description
C	Set of class labels, such that $C = \{C_1, \ldots, C_c\}$ for a c-class problem.
$\mathrm{o}(node)$	List of objects associated to the node $node$.
$\mathrm{o}_{class}(node)$	List of objects of class $class$ in node $node$.
$\mathrm{length}(l)$	Number of elements in the list l.
$\mathrm{child}^{\mathrm{R}}(node, f)$	List of objects that form the right child of $node$ when it is split by feature f.
$\mathrm{child}^{\mathrm{L}}(node, f)$	List of objects that form the left child of $node$ when it is split by feature f.

techniques that are inherent to the classifier. For example, for a KNN classifier, we could have: $\{k=3,\ distance\_metric=\text{euclidean}\}$, which indicates the use of euclidean distance and of three neighbours for the classification decision.

The fourth element of the tuple, states if the problem is decomposed and how. An empty element describes an ensemble without decomposition, which is the most common scenario. For example, {} indicates that all the classifiers in the ensemble solve the same problem; they all discriminate among all the classes in the classification problem.

We will use this taxonomy to describe the ensembles used in our experiments. The following sections will be incrementally adding information to the tuples that describe the methods followed in the construction of the ensembles.

3.2. Decision tree classifiers

All of the ensembles in the experiments have the same number and type of classifiers: 100 C4.5 tree classifiers. A C4.5 tree (Quinlan, 2014) is a decision tree classifier. It is based on the idea of divide and conquer. The training objects are divided at each node depending on the value of a feature. The idea is to refine the training dataset into subsets of objects that are of the same class. The feature selected to divide the objects at a certain node must test the objects in a way that the outcomes are mutually exclusive. The training dataset is divided by the root node according to one of the features. One branch is created for each possible outcome. Each node in the branches now contains the subset of the training dataset that complies with the corresponding outcome of the test in the root node. This process is repeated recursively in each of the nodes until all the subsets consist of objects belonging to the same class or until no test offers any improvement.

In our experiments we work with binary trees, so each test has two mutually exclusive outcomes. For numeric data, we test if the value of the feature is greater than a certain value. For nominal data, we test if the value of the feature falls into a set of values. The decision of on which feature to test must be taken carefully to build a tree that has predictive value. There are several ways to evaluate the split that a feature generates, we will explain the ones we used in the following section.

3.2.1. Split evaluation measures

We used three different split evaluation measures: Quinlan Gain, Gini Impurity and Hellinger Distance. These measures are based on two main ideas: impurity of a node and feature affinity between classes. We did not include any measures based in cost in our experiments because of the difficulty on setting the cost matrix. As mentioned before in Loyola-González et al. (2016); Zhou et al. (2016); Krawczyk et al. (2014b); Freitas (2011), for most datasets this information is unknown and it is hard for experts to establish it.

We will use the symbols in Table 1 to define them.

Quinlan gain (Kuncheva, 2014). This is an entropy-based measure of impurity. The impurity of a node, for a multi-class problem, is defined as:

$$\mathrm{i}(node) = -\sum_{j\in C} \Pr(\mathrm{o}^{j}_{node})\log_2 \Pr(\mathrm{o}^{j}_{node})$$

where $\Pr(\mathrm{o}^{j}_{node})$ is the probability of class j at the node $node$. We can estimate this probability as the proportion of the examples within $node$ that belong to class j:

$$\Pr(\mathrm{o}^{j}_{node}) = \mathrm{length}(\mathrm{o}_j(node))/\mathrm{length}(\mathrm{o}(node))$$

To evaluate the split using feature f, we look at the drop of impurity on average. Since we work with binary trees, the gain of the split is:

$$\Delta\mathrm{i}(node, f) = \mathrm{i}(node) - \Pr(\mathrm{l}^{f}_{node}) \times \mathrm{i}(\mathrm{child}^{\mathrm{L}}(node, f)) - \Pr(\mathrm{r}^{f}_{node}) \times \mathrm{i}(\mathrm{child}^{\mathrm{R}}(node, f))$$

where $\Pr(\mathrm{l}^{f}_{node})$ and $\Pr(\mathrm{r}^{f}_{node})$ denote the probability of each branch of the tree:

$$\Pr(\mathrm{l}^{f}_{node}) = \mathrm{length}(\mathrm{child}^{\mathrm{L}}(node, f))/\mathrm{length}(\mathrm{o}(node))$$

$$\Pr(\mathrm{r}^{f}_{node}) = \mathrm{length}(\mathrm{child}^{\mathrm{R}}(node, f))/\mathrm{length}(\mathrm{o}(node))$$

Gini impurity (Kuncheva, 2014). This measure is computed as the drop of impurity on average, like Quinlan Gain, except this time the impurity of each node is computed in the following way:

$$\mathrm{i}(node) = 1 - \sum_{j\in C} \Pr{}^2(\mathrm{o}^{j}_{node})$$

Hellinger distance. This is a measure of distributional divergence, defined for a binary classification problem (Cieslak and Chawla, 2008). Instead of looking at node impurity, Hellinger distance looks at feature affinity between classes. Let C_+ be the class + and C_- be the class −; the distance between C_+ and C_- for splitting node n using feature f is defined as:

$$\mathrm{d_H}(C_+, C_-, n, f) = \sqrt{\sum_{k=1}^{p}\left(\sqrt{|C_{+k}|/|C_+|} - \sqrt{|C_{-k}|/|C_-|}\right)^2}$$

where k denotes a certain value of the feature for which the distance is being computed. The feature is discretized into p partitions. Our trees use a binary split at each node so p is always equal to two. We will simplify the previous expression using L and R to denote the values of the feature that fall on the left or right partition:

$$\mathrm{d_H}(C_+, C_-, n, f) =$$

$$\sqrt{\left(\sqrt{|C_{+L}|/|C_+|} - \sqrt{|C_{-L}|/|C_-|}\right)^2 + \left(\sqrt{|C_{+R}|/|C_+|} - \sqrt{|C_{-R}|/|C_-|}\right)^2}$$

where:

$$|C+L| = \mathrm{length}(c+(child\hat{L}(n,f))) \quad |C-L| = \mathrm{length}(c-(child\hat{L}(n,f)))$$
$$|C+R| = \mathrm{length}(c+(child\hat{R}(n,f))) \quad |C-R| = \mathrm{length}(c-(child\hat{R}(n,f)))$$
$$|C+| = \mathrm{length}(c+(n)) \quad |C-| = \mathrm{length}(c-(n))$$

Since some of the datasets we tested have more than two classes, we take the multi-class approach proposed by Hoens et al. (2012). In this approach, N distance values are computed, for an N-class problem. For each class, a one-vs-all approach is followed, taking each time one class as the positive class and the rest as the negative class. The highest value is kept as the distance. Fig. 1 outlines the algorithm. This version of the measure selects the same split as traditional Hellinger distance for binary classification problems and it can be applied to multi-class classification problems.

```
$Hellinger = -1$
for each pair of subsets:
    $cur\_value = \left(\sqrt{|C_{+L}|/|C_+|} - \sqrt{|C_{-L}|/|C_-|}\right)^2 + \left(\sqrt{|C_{+R}|/|C_+|} - \sqrt{|C_{-R}|/|C_-|}\right)^2$
    if $cur\_value > Hellinger$
        $Hellinger = cur\_value$
return $\sqrt{Hellinger}$
```

Fig. 1. Algorithm for Hellinger multiclass, simplified representation of Hoens et al. (2012) for a binary split.

Given that this variation in the ensemble is given through the specification of a function which is inherent to the tree classifier, we will include it in the third element of the tuple as the function $split\_eval$ which can take any of the three measures presented. At this point, the tuples that describe our ensembles look like this:

$$\{[\text{C45} : 100], \{\}, \{split\_eval = [\text{Hellinger}|\text{GiniImpurity}|\text{QuinlanGain}]\}, \{\}\}$$

3.2.2. Pruning(P)

The purpose of pruning is to simplify a decision tree without compromising its accuracy by removing some nodes in the tree. While pruning may cause to misclassify more of the training cases, it aims to reduce overfitting the data. There are several pruning methods (Mingers, 1989): cost-complexity pruning, critical value pruning, minimum error pruning, reduce error pruning and pessimistic error pruning.

Some of the ensembles in our experiments perform pruning during the construction of the decision trees. Every time we performed pruning in our experiments, we used pessimistic error pruning (Quinlan, 1987). This pruning method is fast and does not require a separate test set. Since pruning is also exclusive to decision tree classifiers, we specify this source of diversity in our experiments as part the third element of the tuple:

$$\{\ split\_eval = [\text{Hellinger}|\text{GiniImpurity}|\text{QuinlanGain}],$$
$$[pruning = \text{PessimisticError}]\ \}$$

3.3. Ensemble strategies

To train the ensemble, and build the 100 decision tree classifiers that are part of it, we used seven different algorithms. Our initial observation of an ensemble performing better when the use of the features was more uniform, appeared in experiments with bagging ensembles (Camiña et al., 2016). We selected the following variants to avoid bias for one strategy and to test if the hypothesis persisted when other decision tree ensembles were considered. We selected two bagging algorithms, two boosting algorithms and three randomized algorithms. They mostly affect the training dataset of each of the trees in the ensemble by sampling the original training set. Here we use the function $\text{sample}(I, N, d)$ to denote the process of sampling N objects from the dataset defined by I using a probability distribution d. Throughout this section, I is the original training set and $N = |I|$.

Bagging (B). This is the simplest of the ensembles. Each classifier in the ensemble trains on a bootstrap replicate of the training set (Kuncheva, 2014); this is, we sample N times with replacement to create a different training set for each classifier. Each object in the dataset has an equal probability of being chosen; we will use $\text{uniform}(I)$ to define a probability distribution over I where each element has a probability of $1/N$ with $N = |I|$.

$$\{[\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\}, \dots\}$$

Stratified bagging (SB). Like the previous algorithm, we take a bootstrap replicate of the training set, except this time we keep class proportion. Let I_{Cc} denote the objects of class Cc in the dataset with $N_{Cc} = |I_{Cc}|$. For each class c in the dataset, we sample with replacement N_{Cc} times from the objects belonging to that class. For example, for a k-class problem:

$$\{\ [\text{C45} : 100], \{\ \text{sample}(I_{C1}, N_{C1}, \text{uniform}(I_{C1})) + \cdots$$
$$+ \text{sample}(I_{Ck}, N_{Ck}, \text{uniform}(I_{Ck}))\ \}, \dots\ \}$$

Random forest (RF) (Breiman, 2001). This method samples with replacement N objects to create the training dataset for each classifier. Random Forest modifies the C4.5 algorithm in the following way: at each node, it works with a vertical partition of the sampled training set. This means that instead of looking for the best split among all features of the dataset (F) it will consider only a subset of the features:

$$\{\ [\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\},$$
$$\{node\_objs = \text{partition}(S, \log_2|F| + 1, \text{vertical}), \dots\}, \{\}\ \}$$

where S is used to denote the training set obtained from the specification defined by the second element of the tuple.

Random features(RFE). Modification of Random Forest, the difference being that this method does not include sampling the training set:

$$\{[\text{C45} : 100], \{\}, \{node\_objs = \text{partition}(I, \log_2|F| + 1, \text{vertical}), \dots\}, \{\}\}$$

Random subspace (RS) (Breiman, 2001). This method builds each classifier using a vertical partition of the dataset so every tree in the forest will look at a different feature subset:

$$\{[\text{C45} : 100], \{\text{partition}(I, \max(2, |F|/2), \text{vertical})\}, \dots\}$$

AdaBoost (ADA). This ensemble trains classifiers incrementally, one at a time. The training set for the first classifier is sampled from a uniform distribution. After that, the sample distribution is updated based on the pseudo-loss of the previous classifier and the training set for the next classifier is sampled from this updated distribution. We use the function $\text{boost}(\cdot)$ to describe this behaviour:

$$\{[\text{C45} : 100], \{\text{sample}(I, N, \text{boost}(I))\}, \dots\}$$

AdaBoost associates a weight with each object in the dataset. This weight becomes a probability in the sample distribution, the sum of all weights is equal to one. The function boost(I) returns the sample distribution based on these weights. The weights of all objects start by being 1/N, so the first time we call $\text{boost}(I)$ is equivalent to calling $\text{uniform}(I)$.

After that, these weights are updated at every step, they depend on the degree to which each object is misclassified. We used the weight update proposed in section 5.2 of Freund and Schapire (1995). This version of AdaBoost is known as AdaBoost.M2.

RUSBoost (RUS) (Seiffert et al., 2010). This is the only strategy which does not use a training dataset of size N. The size of the training dataset depends on the number of objects in the minority class N_{min}. Random under-sampling is performed in all the other classes, so a balanced dataset, with N_{min} objects from each class, is used in each iteration. The balanced dataset is then sampled using a distribution based on the

Table 2
Feature-use frequency.

Feature	Times used	Feature-use frequency
f1	5	5/12
f2	4	4/12
f3	3	3/12

pseudo-loss (similar to AdaBoost). For example, for a k-class problem we would have:

$$\{[\text{C45} : 100], \{\text{sample}(B, N_{min} \times k, \text{boost}(B)),$$

$$B = \text{sample}(I_{C1}, N_{min}, \text{uniform}(I_{C1})) + \cdots$$

$$+ \text{sample}(I_{Ck}, N_{min}, \text{uniform}(I_{Ck}))\}, \dots\}$$

RUSBoost is designed to deal with classification problems with imbalanced data. We decided to include it in the analysis because imbalanced datasets are frequent in real-world scenarios.

After the ensemble is built, we use two different approaches to classify a new object. For the first five algorithms, new objects are labelled using a plurality voting system: each of the trees in the ensemble casts a vote and the class with more votes is the predicted class. ADABoost and RUSBoost use a different voting system: each of its base classifiers has an associated weight, instead of casting a vote, they aggregate its weighted result into a global result, and the new observation is labelled as the class with the maximum value in the global result.

4. Variance on the feature-use frequency

The central idea behind our experiments is to establish a relation between the uniformity on the feature-use frequencies and the classification error of decision tree ensembles. Our working hypothesis is that when an ensemble uses features more uniformly among the trees it will have a smaller error. To determine when the ensemble uses the features more uniformly, we rely on the concept of feature-use frequency which was informally described in Camiña et al. (2016). In this paper, we provide a formal definition. We compute the feature-use frequency of feature f in an ensemble as:

$$\text{ff}(f) = \sum_{j=1}^{S} \text{nodes}_f(T_j) / \sum_{j=1}^{S} \text{nodes}(T_j)$$

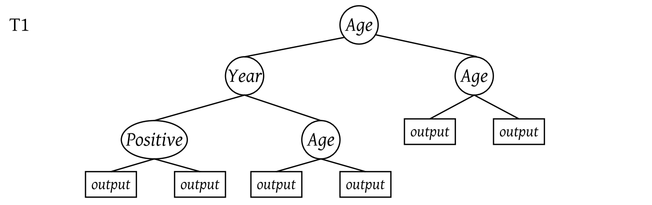
where S is the size of the ensemble and T_j represents the jth tree in the ensemble. The function $\text{nodes}_f(T)$ returns the number of nodes in the tree T with splitting feature f. The function $\text{nodes}(T)$ returns the number of nodes in the tree T.

The feature-use frequency is basically the proportion of times a feature was used to split a node across all the trees in the ensemble. A feature might be used more than once or not at all in a tree. Once we have the feature-use frequencies of all the features we compute the variance among them, as a measure of uniformity. If one feature was not used, or if one feature was used consistently more than the others, the variance will be greater than if all features were used uniformly.

As an illustrative example, we will use the ensemble that contains the trees shown in Fig. 2. This ensemble has only two trees, denoted as T1 and T2. It solves a classification problem with three features, denoted as *f1*, *f2* and *f3*. The first step is to compute the frequency of each feature throughout the ensemble. In this example, *f1* is the splitting feature in three nodes of the first tree and in two nodes of the second tree, so $\sum_{j=1}^{2} \text{count}_{f1}(T_j)$ is equal to 5. We do this computation for all features in the ensemble and divide by the total of nodes in the ensemble to get what we call the feature-use frequency, as shown in Table 2.

Once we have the frequencies of all features, we get the variance. In our example, we have $\mu = 1/3$, so the variance is

$$\sigma^2 = ((5/12 - 1/3)^2 + (4/12 - 1/3)^2 + (3/12 - 1/3)^2)/(3 - 1) = 0.0069$$



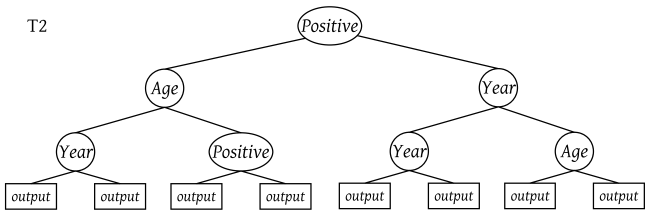


Fig. 2. Two examples (T1 and T2) of decision trees.

This variance is the measure we use to compare ensembles and to look for a relationship with classification error. In the next sections, it will be referred as VarianceFF, for short. The measure does not represent diversity between the trees in the ensemble. Rather, it represents how fair the ensemble is in terms of feature usage. A variance closer to zero denotes a fairer feature usage. This explains this paper's title: in terms of classification, some features speak loud, but together they all speak louder.

5. Material and methods

To test the hypothesis we solved 60 different classification problems using 42 different ensembles strategies and then compared their error with their varianceFF. These strategies are the result of taking the dot product of 7 different algorithms, 3 split measures and 2 pruning modes. Table 3 shows the 42 decision tree ensembles we used. The first column specifies the acronym that identifies each ensemble. The acronyms are formed by the juxtaposition of the algorithm and distance initials, followed by a 'P' if pruning is part of the strategy. The next six columns indicate the algorithm that was used to build the ensemble. These algorithms are described in Section 3.3. Each row will have an $\times$ in one of these six columns to mark the algorithm used by that ensemble. The next three columns indicate the split evaluation function used to build each tree in the ensemble. These functions are described in Section 3.2. Each row will have an $\times$ in one of these three columns to mark the split evaluation function used by that ensemble. The last column indicates if pruning was performed. When row a row has an $\times$ in this column it means pessimistic error pruning was used by that ensemble.

For example, the BQ ensemble, uses the Bagging (B) algorithm to build the ensemble with each tree using Quinlan Gain (Q) to evaluate its splits and no pruning. We can express it as: $\{[\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\}, \{split\_eval = \text{QuinlanGain}\}, \{\}\}$, according to our taxonomy defined in Section 3.1. Appendix describes the ensembles using this taxonomy.

The classification problems were taken from the KEEL data set repository (Alcalá-Fdez et al., 2011). The datasets in KEEL are widely used in the literature (Loyola-González et al., 2016; Krawczyk et al., 2014b; Pendharkar, 2016; Krawczyk et al., 2015; Zhang et al., 2017; Gu and Jin, 2017; Visentini et al., 2016) and include a wide range of problems: binary and multi-class (up to 26 classes), which are different in size, number of features, difficulty level and domain. The KEEL data set repository provides 10-DOBSCV versions of the standard classification data sets, this means that each dataset has the same format and is partitioned in 10 folds using distribution optimally balanced

Table 3
Decision Tree Ensembles used in the experiments.

	Bagging (B)	Stratified Bagging(SB)	Random Forest (RF)	Random Features (RFE)	Random Subspace (RS)	AdaBoost.M2 (ADA)	RUSBoost (RUS)	**Quinlan Gain (Q)**	**Gini Impurity (G)**	**Hellinger Distance (H)**	***pruning (P)***
BQ	×							×			
BG	×								×		
BH	×									×	
BQP	×							×			×
BGP	×								×		×
BHP	×									×	×
SBQ		×						×			
SBG		×							×		
SBH		×								×	
SBQP		×						×			×
SBGP		×							×		×
SBHP		×								×	×
RFQ			×					×			
RFG			×						×		
RFH			×							×	
RFQP			×					×			×
RFGP			×						×		×
RFHP			×							×	×
RFEQ				×				×			
RFEG				×					×		
RFEH				×						×	
RFEQP				×				×			×
RFEGP				×					×		×
RFEHP				×						×	×
RSQ					×			×			
RSG					×				×		
RSH					×					×	
RSQP					×			×			×
RSGP					×				×		×
RSHP					×					×	×
ADAQ						×		×			
ADAG						×			×		
ADAH						×				×	
ADAQP						×		×			×
ADAGP						×			×		×
ADAHP						×				×	×
RUSQ							×	×			
RUSG							×		×		
RUSH							×			×	
RUSQP							×	×			×
RUSGP							×		×		×
RUSHP							×			×	×

stratified cross-validation (DOB-SCV) (Moreno-Torres et al., 2012). The datasets do not have missing values.

We run the experiment for each of the predefined training and testing folds. We left out of the analysis the datasets that achieved perfect classification in all the variants. So, we have 42 different ensembles for each of the 60 classification problems (see Table 4 for a list of datasets used), a total of 2520 ensembles. We average the results of the 10-folds to obtain the error and VarianceFF of each ensemble variant.

6. Results and discussion

6.1. Error-VarianceFF correlation

If given a set of decision tree ensembles, when the error of the ensemble goes up so does the varianceFF, there will be a monotonic relationship between these variables in our results. To validate this hypothesis we applied a Spearman's rank correlation test between the varianceFF and the error of the ensemble. To see the general effect, we take the average of these two variables across all datasets as the paired variables. We have a total 42 pairs, one per type of ensemble, on which we perform Spearman's test. The result is a 0.4420225 correlation (with a p-value of 0.003672), which means that, on average, the error increases when the variance of the feature-use frequencies increases, supporting our hypothesis of a more uniform use of the features leading to a better classification performance.

Now, instead of the average, we look at each dataset individually so we perform Spearman's test 60 times. The paired variables are, again, the variance of feature-use frequencies and the error of the ensemble. We have 42 pairs per dataset. Out of 60 datasets, 42 have a positive correlation while only 18 have a negative correlation. However, only in 29 datasets the relation is significant (p-value < 0.05), as seen in Table 5. If we only consider significant correlations we see that 89.66% of these cases have a positive correlation, while 10.34% of them have a negative correlation. Fig. 3 shows how the correlation between the varianceFF and the error in the majority of the databases is greater than zero.

We can see that in some classification problems the correlation is not as expected. We grouped the problems by feature dimensionality, in order to see how strong is the hypothesis for problems with low and medium dimensional feature space. We take as a reference the division presented in the UCI Machine Learning Repository (Lichman, 2013) and assign the datasets with less than 10 features to the low dimensionality group, and datasets with 10 to 100 are assigned to the medium group; there are no high dimensionality problems (with more than 100 features) among the KEEL benchmark datasets, so this group is not included in this analysis and left as future work.

We perform Spearman's test again, this time we take the average of the error and varianceFF per group. For the low dimensionality group we find a 0.03492424 correlation (with a p-value of 0.8258) which indicates that there really is no monotonic relationship between the error and the varianceFF in these datasets. For the medium dimensionality group we find a 0.5527105 correlation (with a p-value of 0.0001883) which is a

Table 4
Datasets used for the experiments.

Dataset	Classes	Attributes	Objects	Dataset	Classes	Attributes	Objects
movement-libras	15	90	360	heart	2	13	270
coil2000	2	85	9,822	marketing	9	13	6,876
optdigits	10	64	5,620	vowel	11	13	990
sonar	2	60	208	wine	3	13	178
splice	3	60	3,190	flare	6	11	1,066
spambase	2	57	4,597	magic	2	10	19,020
spectfheart	2	44	267	page-blocks	5	10	5,472
texture	11	40	5,500	breast	2	9	277
chess	2	36	3,196	contraceptive	3	9	1,473
dermatology	6	34	358	saheart	2	9	462
ionosphere	2	33	351	shuttle	7	9	58,000
wdbc	2	30	569	tic-tac-toe	2	9	958
automobile	6	25	150	wisconsin	2	9	683
thyroid	3	21	7,200	pima	2	8	758
german	2	20	1,000	yeast	10	8	1,484
ring	2	20	7,400	appendicitis	2	7	106
twonorm	2	20	7,400	ecoli	8	7	336
bands	2	19	365	led7digit	10	7	500
hepatitis	2	19	80	bupa	2	6	345
segment	7	19	2,310	car	4	6	1,728
lymphography	4	18	148	mammographic	2	5	830
vehicle	4	18	846	newthyroid	3	5	215
housevotes	2	16	232	phoneme	2	5	5,404
kr-vs-k	18	6	28,056	tae	3	5	151
letter	26	16	20,000	balance	3	4	625
penbased	10	16	10,992	hayes-roth	3	4	160
zoo	7	16	101	iris	3	4	150
crx	2	15	653	haberman	2	3	306
australian	2	14	690	titanic	2	3	2,201
cleveland	5	13	297	banana	2	2	5,300

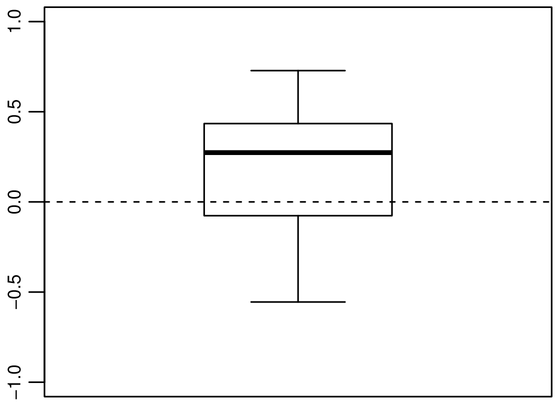


Fig. 3. Spearman's rank correlation coefficient.

stronger correlation than for all datasets. The original observation comes from a classification problem with 16 features, (Camiña et al., 2016). This points to the hypothesis being true for this type of classification problem.

6.2. *An insight into varianceFF behaviour*

After observing the use of the features in the ensembles, we noticed that the choice of performing feature selection, bagging, or boosting, has more effect in the varianceFF than the split evaluation measure or pruning. Fig. 4 shows the average variance of the ensembles, in all datasets, grouped by algorithm. We can see how the six points for each algorithm stay concentrated in a small range, independently of the split measure or pruning. Algorithms with feature selection are fairer with the use of features, followed by Adaboost, and finally by RUSboost and algorithms with just bagging. It is interesting to see how AdaBoost which does not manipulate the features is closer in variance to randomized algorithms.

Table 5
Summary of Spearman tests results in 60 datasets. Count column states the number of datasets that fall in the category dictated by the Correlation column. The ρ range column states the minimum and maximum ρ obtained among the datasets that fall in the category dictated by the Correlation column.

Correlation	Count	ρ range
Positive significant correlation	26	0.32 to 0.72
Not significant correlation	31	−0.28 to 0.29
Negative significant correlation	3	−0.67 to −0.32

Fig. 5 shows a similar graph for binary and multiclass problems. Here we see different markers, depending on the split measure and pruning. For binary classification problems, pruning the ensemble results in a greater varianceFF for bagging algorithms (SB and B) and in a lower varianceFF for the rest of the algorithms(RFE, RF, RS, ADA and RUS), compared to the non-pruned counterparts. For multiclass classification problems, there is no obvious relation between pruning and the varianceFF of the ensemble. The most evident change between binary and multiclass problems is the behaviour of the varianceFF in RUSBoost, being much greater in the multiclass version. This is probably due to the minority class being much smaller than the other classes. In binary classification problems, RUSboost's and AdaBoost's variance has a similar behaviour. We can notice that there is no single measure that achieves the lowest variance per algorithm. Since there is no clear indicator as to how the split measure affect the uniformity of feature usage; it is difficult to consider it as part of an heuristic to obtain a classifier with a lower varianceFF.

6.3. *What does the majority say?*

Given any two ensembles, we expect that the one with the lowest error is also the one with a more uniform feature-use frequency. To test this, we performed a one-proportion z-test in our classification results. This test will evaluate a population proportion p_0, in this case, p_0 is the proportion of the ensembles that have a lower error when they have a more uniform feature-use frequency. To test if the majority of times this

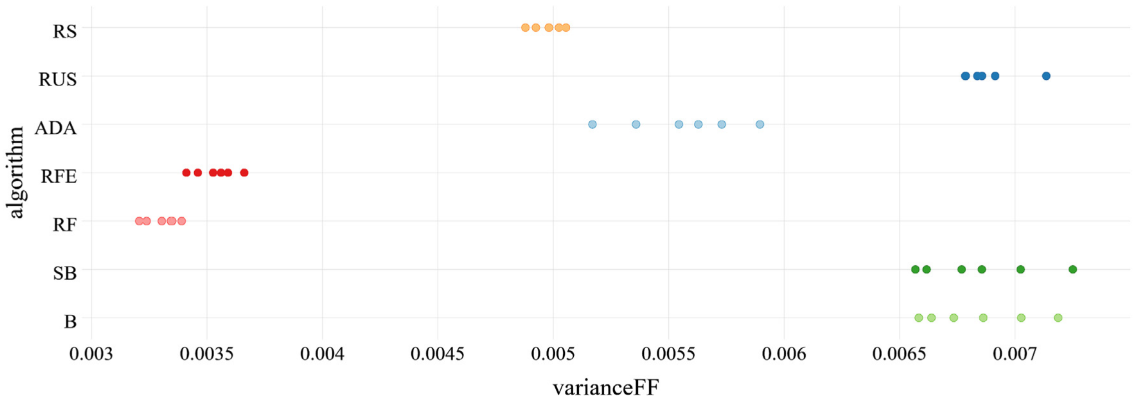
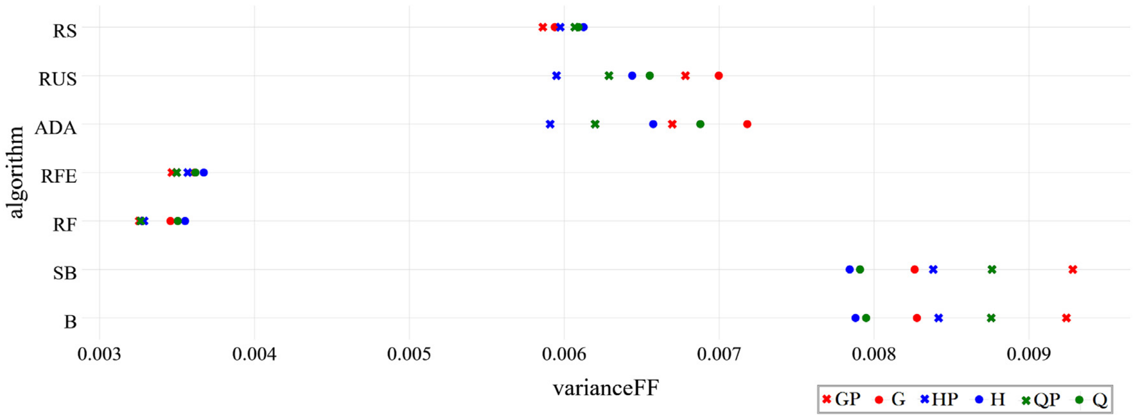
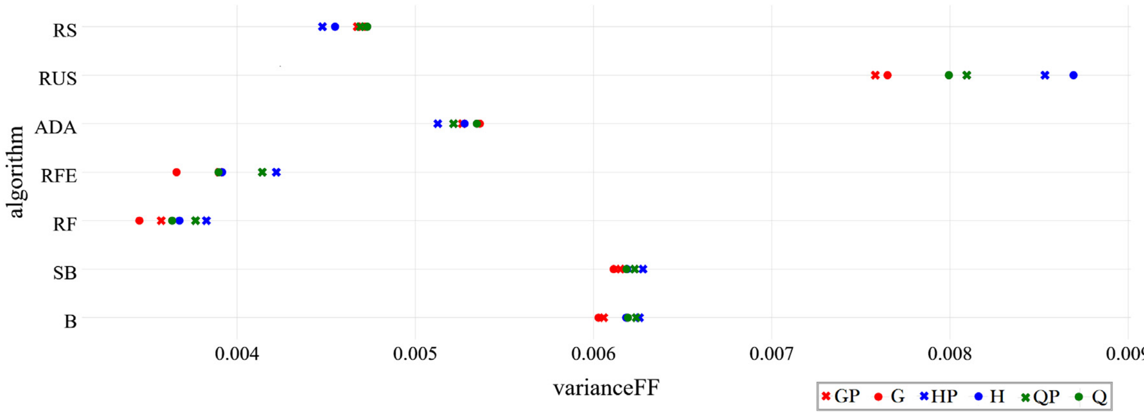


Fig. 4. VarianceFF per algorithm. Algorithms: Random Subspace(RS), RUSBoost (RUS), AdaBoost.M2 (ADA), Random Features(RFE), Random Forest (RF), Stratified Bagging(SB), and Bagging (B).



(a) Variance in binary problems.



(b) Variance in multiclass problems.

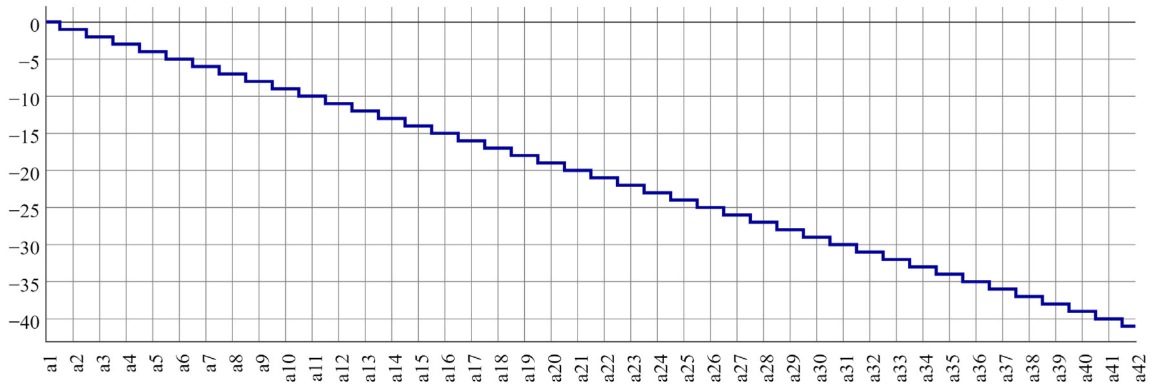
Fig. 5. VarianceFF per algorithm, divided in binary and multi-class classification problems. Algorithms: Random Subspace(RS), RUSBoost (RUS), AdaBoost.M2 (ADA), Random Features(RFE), Random Forest (RF), Stratified Bagging(SB), and Bagging (B). Measures and Pruning: Gini pruned (GP), Gini not pruned (G), Hellinger pruned (HP), Hellinger not pruned (H), Quinlan pruned (QP), Quinlan not pruned (Q)..

is true, we perform a right tailed test and set the null and alternative hypotheses to be:

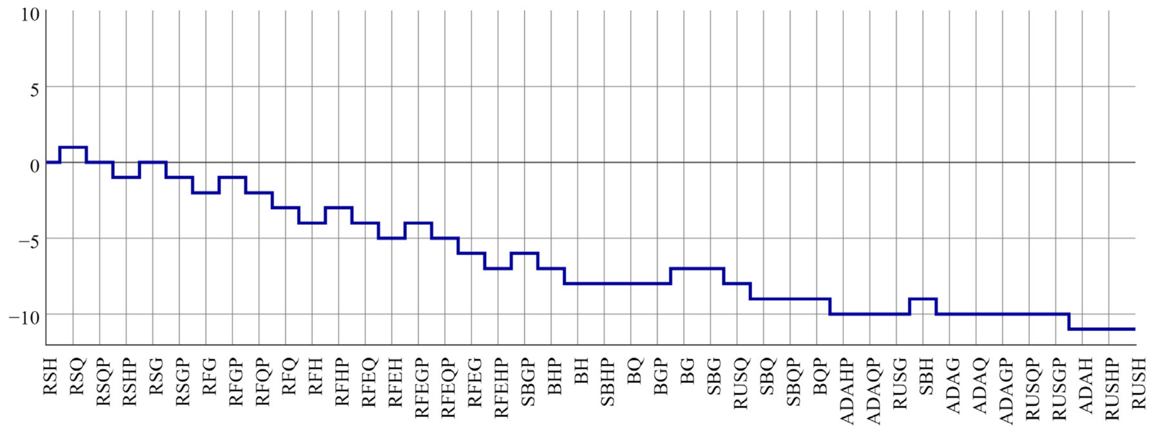
$H_0 : p_0 = 0.5$

$H_a : p_0 > 0.5$

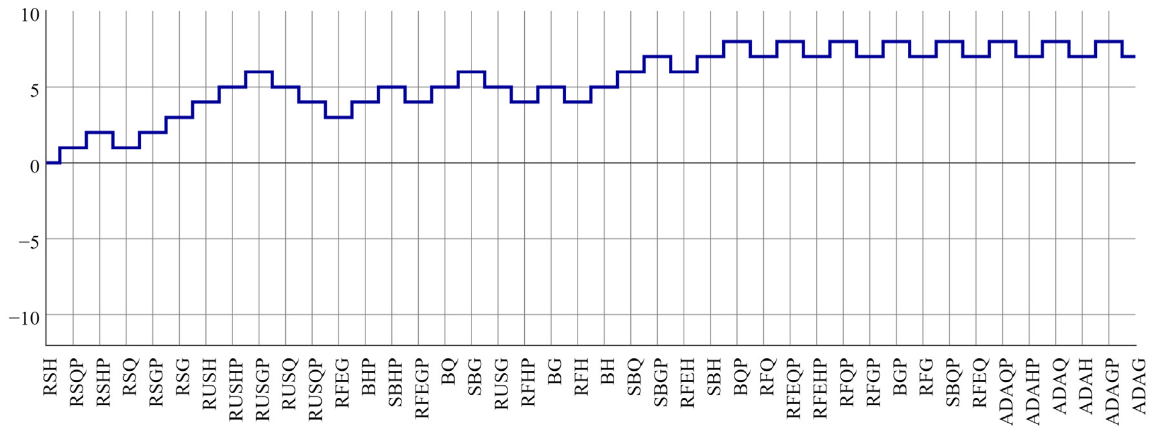
For each of the classification problems, we compare every pair of ensembles (861 pairs per dataset). If the ensemble with the lowest error also has a more uniform feature-use frequency then we take the pair as positive. On the contrary, if the ensemble with the highest performance is the one with a more uniform feature-use frequency then we take the pair as negative. And finally, if both ensembles have the same error, we discard the pair. We have 50471 pairs in total, as 1189 were discarded due to ties, 28916 of this pairs are positive so we have $\hat{p} = 0.572923$. With this information, we reject H_0, from which we can conclude with high certainty (p-value $= 0$) that given any two decision tree ensembles, in the majority of the occasions, the ensemble with a more uniform use of the feature space will also be the one with a lower error. Furthermore, we can go up to $p_0 = 0.56$ and still accept the hypothesis with high certainty in a right-tailed test.



(a) Perfect relation.



(b) Dataset chess.



(c) Dataset magic.

Fig. 6. Step graphs. X-axis marks the algorithm being compared (against its predecessor). The algorithms are sorted in descending order by error. Y-axis marks the step level. The top graph is the representation of 42 ensembles (a1–a42) that comply with the relationship of lower variance leading to lower error. The next two graphs show the relation for the specified datasets.

6.4. Visualizing the results

We created a visual representation of the results in which we observe the increase or decrease in varianceFF as the error gets smaller. For each dataset, the results of the 42 ensembles are sorted by error in descending order. Starting from a zero level we draw a step up whenever the varianceFF increases, compared to the one at the previous level, and a step down whenever it decreases. The steps are of the same size independently of the increase (or decrease) quantity, they only show the direction of the change. When we have two algorithms whose error is the same, the steps remain at the same level. After a tie, we compare the next ensemble's varianceFF with the mean varianceFF of the ensembles at the current level. Our expectation is that as the error decreases the varianceFF must decrease, and so the steps will be going down. We consider a good graph when the steps finish below zero because it is an indication that the relation was true more times than it was not. If the relationship was perfect, the graph for a given problem would only have decreasing steps, as shown in Fig. 6(a).

After plotting the 60 datasets[1] , we see that 37 fulfill our expectation. Fig. 6(b) shows the behaviour of the dataset that ended lowest in the

[1] These plots are available online at https://sites.google.com/site/miguelmedinaperez/supplementarymaterials/EAAI2017.pdf.

scale. Six of the datasets ended exactly at zero, while 17 datasets finish at a level higher than the original. Fig. 6(c) shows the dataset with the worst performance according to this metric. This visualization is not directly linked to any statistical test but it allows us to see if there is a general decreasing tendency. It is also easily appreciated when there are ties between the classifiers, as an example we have the dataset chess, in Fig. 6(b), where long horizontal lines show ties in the error among the classifiers.

7. Conclusions and future work

Our results confirm that, in the majority of the cases, a more uniform use of the features inside a decision tree ensemble leads to a smaller error. Even if the correlation is not always true, significant negative effects happen in less than 8% of the classification problems tested. The correlation between this two variables, motivates us to create an algorithm that drives the construction of an ensemble based on decision trees towards a more uniform use of the features, for datasets with low and medium dimensionality.

Giving the same importance to all features in the dataset might sound contra-intuitive, as feature selection methods are a hot topic in the machine learning community. Feature Selection methods (like Sequential Forward Selection or Sequential Backward Selection) assume that, in general, a subset of the features will yield a lower classification error. However, in this analysis, we have found that for decision tree ensembles solving classification problems of less than one hundred features, this phenomenon is not always true. On the contrary, decision tree classifiers that use all features uniformly have a lower error most of the times. Of course, features with zero entropy and duplicated features are not part of the analysis, as they clearly have no extra value for a classifier. We believe that it is worth to further experiment along this line to create and test techniques that aim to maximize uniformity of feature-use frequency in decision tree ensembles.

Future research will be directed towards analysing if the variance on the feature-use frequency can serve as guiding measure in the construction of a decision tree ensemble. It could be an alternative to using diversity, which is known to be relevant for ensemble performance but has not had good results when explicitly used to guide the construction of the ensemble (Kuncheva, 2014). However, methods that implicitly add diversity tend to have a good performance. It remains an open question if explicitly using this new measure will benefit the ensemble since it might be the case that the varianceFF has the same effect in the construction of ensembles as diversity.

We envision two possible paths to tailor the construction of an ensemble into integrating knowledge about the current feature usage: ensemble pruning and algorithm tuning. A forest can reduce its size by keeping the desired number of trees to form a smaller forest taking into consideration its varianceFF. One option is to discard classifiers if by doing so the varianceFF is not affected (is smaller than before or within a set threshold). Another option is to incrementally add trees that minimize the varianceFF. These ensemble pruning techniques could be combined with other measures to create a more advanced ensemble pruning technique with multiple criteria analysis, as suggested by Krawczyk et al. (2017). The algorithms that include feature selection, such as Random Forest, Random Features, and Random Subspace, can be tuned to include information about the trees that have already been built. The idea is that instead of having a completely random feature selection, we have a weighted feature selection, where it is more probable for features that have not been used to be selected. Of course, this requires a snapshot or history of the forest; so a percentage of the trees are built with random selection and then the algorithm switches to weighted selection. Another option is to incrementally build the forest, and stop adding trees once the varianceFF has stabilized, all of this to build a smaller forest. There could be many more variations of algorithms to be explored, we hope this paper will motivate researches to pursue further research the use of features in decision tree ensembles.

Acknowledgements

We thank the members of the Machine Learning Models group at Tecnologico de Monterrey, Estado de Mexico, for providing invaluable, useful suggestions and advice on this paper. We also thank our reviewers from providing the invaluable suggestions and observations that led to significant improvement of our text. The first author was supported by CONACYT student scholarship 372236.

Appendix. Ensembles used in the experiments

Here are the decision tree ensembles used in the experiments expressed by the taxonomy presented in the paper. Note that the first element is always the same because all ensembles contain exactly one hundred trees. The second element takes the value defined by each of the algorithms (see Section 3.3). In the third element, the function $split\_eval$ takes one of the three split measures considered in the experiments, and the function $pruning$ is present on ensembles that performed pruning. When specified by the algorithms, the third element of the tuple adds the extra information needed to build the classifier. The fourth element is always empty, as there is no problem decomposition.

$$\begin{aligned}
BQ = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I, N, \mathrm{uniform}(I))\},\\
& \{split\_eval = \mathrm{QuinlanGain}\}, \{\} \}\\
BG = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I, N, \mathrm{uniform}(I))\},\\
& \{split\_eval = \mathrm{GiniImpurity}\}, \{\} \}\\
BH = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I, N, \mathrm{uniform}(I))\},\\
& \{split\_eval = \mathrm{Hellinger}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
BQP = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I, N, \mathrm{uniform}(I))\},\\
& \{split\_eval = \mathrm{QuinlanGain},\\
& pruning = \mathrm{PessimisticError}\}, \{\} \}\\
BGP = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I, N, \mathrm{uniform}(I))\},\\
& \{split\_eval = \mathrm{GiniImpurity},\\
& pruning = \mathrm{PessimisticError}\}, \{\} \}\\
BHP = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I, N, \mathrm{uniform}(I))\},\\
& \{split\_eval = \mathrm{Hellinger},\\
& pruning = \mathrm{PessimisticError}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
SBQ = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I_{C1}, N_{C1}, \mathrm{uniform}(I_{C1})) + \cdots\\
& + \mathrm{sample}(I_{Ck}, N_{Ck}, \mathrm{uniform}(I_{Ck}))\},\\
& \{split\_eval = \mathrm{QuinlanGain}\}, \{\} \}\\
SBG = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I_{C1}, N_{C1}, \mathrm{uniform}(I_{C1})) + \cdots\\
& + \mathrm{sample}(I_{Ck}, N_{Ck}, \mathrm{uniform}(I_{Ck}))\},\\
& \{split\_eval = \mathrm{GiniImpurity}\}, \{\} \}\\
SBH = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I_{C1}, N_{C1}, \mathrm{uniform}(I_{C1})) + \cdots\\
& + \mathrm{sample}(I_{Ck}, N_{Ck}, \mathrm{uniform}(I_{Ck}))\},\\
& \{split\_eval = \mathrm{Hellinger}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
SBQP = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I_{C1}, N_{C1}, \mathrm{uniform}(I_{C1})) + \cdots\\
& + \mathrm{sample}(I_{Ck}, N_{Ck}, \mathrm{uniform}(I_{Ck}))\},\\
& \{split\_eval = \mathrm{QuinlanGain}, pruning = \mathrm{PessimisticError}\}, \{\} \}\\
SBGP = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I_{C1}, N_{C1}, \mathrm{uniform}(I_{C1})) + \cdots\\
& + \mathrm{sample}(I_{Ck}, N_{Ck}, \mathrm{uniform}(I_{Ck}))\},\\
& \{split\_eval = \mathrm{GiniImpurity}, pruning = \mathrm{PessimisticError}\}, \{\} \}\\
SBHP = \{ & [\mathrm{C45} : 100], \{\mathrm{sample}(I_{C1}, N_{C1}, \mathrm{uniform}(I_{C1})) + \cdots\\
& + \mathrm{sample}(I_{Ck}, N_{Ck}, \mathrm{uniform}(I_{Ck}))\},\\
& \{split\_eval = \mathrm{Hellinger}, pruning = \mathrm{PessimisticError}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{RFQ} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\},\\
& \{node\_objs = \text{partition}(S, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{QuinlanGain}\}, \{\} \}\\
\text{RFG} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\},\\
& \{node\_objs = \text{partition}(S, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{GiniImpurity}\}, \{\} \}\\
\text{RFH} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\},\\
& \{node\_objs = \text{partition}(S, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{Hellinger}\}, \{\} \}\\
\text{RFQP} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\},\\
& \{node\_objs = \text{partition}(S, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{QuinlanGain},\\
& pruning = \text{PessimisticError}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{RFGP} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\},\\
& \{node\_objs = \text{partition}(S, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{GiniImpurity},\\
& pruning = \text{PessimisticError}\}, \{\} \}\\
\text{RFHP} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{uniform}(I))\},\\
& \{node\_objs = \text{partition}(S, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{Hellinger},\\
& pruning = \text{PessimisticError}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{RFEQ} = \{ & [\text{C45} : 100], \{\},\\
& \{node\_objs = \text{partition}(I, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{QuinlanGain}\}, \{\} \}\\
\text{RFEG} = \{ & [\text{C45} : 100], \{\},\\
& \{node\_objs = \text{partition}(I, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{GiniImpurity}\}, \{\} \}\\
\text{RFEH} = \{ & [\text{C45} : 100], \{\},\\
& \{node\_objs = \text{partition}(I, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{Hellinger}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{RFEQP} = \{ & [\text{C45} : 100], \{\},\\
& \{node\_objs = \text{partition}(I, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{QuinlanGain}, pruning = \text{PessimisticError}\}, \{\} \}\\
\text{RFEGP} = \{ & [\text{C45} : 100], \{\},\\
& \{node\_objs = \text{partition}(I, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{GiniImpurity}, pruning = \text{PessimisticError}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{RFEHP} = \{ & [\text{C45} : 100], \{\},\\
& \{node\_objs = \text{partition}(I, \log_2|F| + 1, \text{vertical}),\\
& split\_eval = \text{Hellinger}, pruning = \text{PessimisticError}\}, \{\} \}\\
\text{RSQ} = \{ & [\text{C45} : 100], \{\text{partition}(I, \log_2|F| + 1, \text{vertical})\},\\
& \{split\_eval = \text{QuinlanGain}\}, \{\} \}\\
\text{RSG} = \{ & [\text{C45} : 100], \{\text{partition}(I, \log_2|F| + 1, \text{vertical})\},\\
& \{split\_eval = \text{GiniImpurity}\}, \{\} \}\\
\text{RSH} = \{ & [\text{C45} : 100], \{\text{partition}(I, \log_2|F| + 1, \text{vertical})\},\\
& \{split\_eval = \text{Hellinger}\}, \{\} \}\\
\text{RSQP} = \{ & [\text{C45} : 100], \{\text{partition}(I, \log_2|F| + 1, \text{vertical})\},\\
& \{split\_eval = \text{QuinlanGain},\\
& pruning = \text{PessimisticError}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{RSGP} = \{ & [\text{C45} : 100], \{\text{partition}(I, \log_2|F| + 1, \text{vertical})\},\\
& \{split\_eval = \text{GiniImpurity},\\
& pruning = \text{PessimisticError}\}, \{\} \}\\
\text{RSHP} = \{ & [\text{C45} : 100], \{\text{partition}(I, \log_2|F| + 1, \text{vertical})\},\\
& \{split\_eval = \text{Hellinger},\\
& pruning = \text{PessimisticError}\}, \{\} \}\\
\text{ADAQ} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{boost}(I))\},\\
& \{split\_eval = \text{QuinlanGain}\}, \{\} \}\\
\text{ADAG} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{boost}(I))\},\\
& \{split\_eval = \text{GiniImpurity}\}, \{\} \}\\
\text{ADAH} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{boost}(I))\},\\
& \{split\_eval = \text{Hellinger}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{ADAQP} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{boost}(I))\},\\
& \{split\_eval = \text{QuinlanGain},\\
& pruning = \text{PessimisticError}\}, \{\} \}\\
\text{ADAGP} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{boost}(I))\},\\
& \{split\_eval = \text{GiniImpurity},\\
& pruning = \text{PessimisticError}\}, \{\} \}\\
\text{ADAHP} = \{ & [\text{C45} : 100], \{\text{sample}(I, N, \text{boost}(I))\},\\
& \{split\_eval = \text{Hellinger},\\
& pruning = \text{PessimisticError}\}, \{\} \}\\
\text{RUSQ} = \{ & [\text{C45} : 100], \{\text{sample}(B, N_{min} \times k, \text{boost}(B)),\\
& B = \text{sample}(I_{C1}, N_{min}, \text{uniform}(I_{C1})) + \cdots\\
& + \text{sample}(I_{Ck}, N_{min}, \text{uniform}(I_{Ck}))\},\\
& \{split\_eval = \text{QuinlanGain}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{RUSG} = \{ & [\text{C45} : 100], \{\text{sample}(B, N_{min} \times k, \text{boost}(B)),\\
& B = \text{sample}(I_{C1}, N_{min}, \text{uniform}(I_{C1})) + \cdots\\
& + \text{sample}(I_{Ck}, N_{min}, \text{uniform}(I_{Ck}))\},\\
& \{split\_eval = \text{GiniImpurity}\}, \{\} \}\\
\text{RUSH} = \{ & [\text{C45} : 100], \{\text{sample}(B, N_{min} \times k, \text{boost}(B)),\\
& B = \text{sample}(I_{C1}, N_{min}, \text{uniform}(I_{C1})) + \cdots\\
& + \text{sample}(I_{Ck}, N_{min}, \text{uniform}(I_{Ck}))\},\\
& \{split\_eval = \text{Hellinger}\}, \{\} \}
\end{aligned}$$

$$\begin{aligned}
\text{RUSQP} = \{ & [\text{C45} : 100], \{\text{sample}(B, N_{min} \times k, \text{boost}(B)),\\
& B = \text{sample}(I_{C1}, N_{min}, \text{uniform}(I_{C1})) + \cdots\\
& + \text{sample}(I_{Ck}, N_{min}, \text{uniform}(I_{Ck}))\},\\
& \{split\_eval = \text{QuinlanGain}\},\\
& pruning = \text{PessimisticError}\ \}, \{\} \}\\
\text{RUSGP} = \{ & [\text{C45} : 100], \{\ \text{sample}(B, N_{min} \times k, \text{boost}(B)),\\
& B = \text{sample}(I_{C1}, N_{min}, \text{uniform}(I_{C1})) + \cdots\\
& + \text{sample}(I_{Ck}, N_{min}, \text{uniform}(I_{Ck}))\ \},\\
& \{split\_eval = \text{GiniImpurity}\},\\
& pruning = \text{PessimisticError}\ \}, \{\} \}\\
\text{RUSHP} = \{ & [\text{C45} : 100], \{\ \text{sample}(B, N_{min} \times k, \text{boost}(B)),\\
& B = \text{sample}(I_{C1}, N_{min}, \text{uniform}(I_{C1})) + \cdots\\
& + \text{sample}(I_{Ck}, N_{min}, \text{uniform}(I_{Ck}))\ \}\ \},\\
& \{split\_eval = \text{Hellinger}\},\\
& pruning = \text{PessimisticError}\ \}, \{\} \}
\end{aligned}$$

References

Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F., 2011. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. J. Multiple-Valued Logic Soft Comput. 17 (2–3), 255–287.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Bui, D.T., Ho, T.C., Revhaug, I., Pradhan, B., Nguyen, D.B., 2014. Landslide susceptibility mapping along the national road 32 of vietnam using gis-based j48 decision tree classifier and its ensembles. In: Cartography from Pole to Pole. Springer Berlin Heidelberg, pp. 303–317.

Camiña, J.B., Monroy, R., Trejo, L.A., Medina-Perez, M.A., 2016. Temporal and spatial locality, an abstraction for masquerade detection. IEEE Trans. Inf. Forensics Secur. 11 (9), 2036–2051.

Cieslak, D.A., Chawla, N.V., 2008. Learning decision trees for unbalanced data. In: Daelemans, W., Goethals, B., Morik, K. (Eds.), Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 241–256.

Cunningham, P., Carney, J., 2000. Diversity versus quality in classification ensembles based on feature selection, in: Proceedings of the 11th European Conference on Machine Learning. pp. 109–116.

Fannes, T., Vandermarliere, E., Schietgat, L., Degroeve, S., Martens, L., Ramon, J., 2013. Predicting tryptic cleavage from proteomics data using decision tree ensembles. J. Proteome Res. 12 (5), 2253–2259.

Freitas, A., 2011. Building cost-sensitive decision trees for medical applications. AI Commun. 24 (3), 285–287.

Freund, Y., Schapire, R.E., 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory. Springer, pp. 23–37.

Freund, Y., Schapire, R.E., 1996. Experiments with a New Boosting Algorithm. Int. Conf. Mach. Learn. 148–156.

Gu, S., Jin, Y., 2017. Multi-train: a semi-supervised heterogeneous ensemble classifier. Neurocomputing 249, 202–211.

Guerra-Salcedo, C., Whitley, D., 2000. Feature Selection Mechanisms for Ensemble Creation: A Genetic Search Perspective. Dna 180 (39), 1186.

Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20 (8), 832–844.

Hoens, T.R., Qian, Q., Chawla, N.V., Zhou, Z.-h., 2012. Building decision trees for the multi-class imbalance problem. In: Advances in Knowledge Discovery and Data Mining. pp. 122–134.

Kelarev, A.V., Abawajy, J., Stranieri, A., Jelinek, H.F., 2013. Empirical investigation of decision tree ensembles for monitoring cardiac complications of diabetes. Int. J. Data Warehousing Min. 9 (4), 1–18.

Ko, C., Sohn, G., Remmel, T.K., Miller, J.R., 2016. Maximizing the diversity of ensemble random forests for tree genera classification using high density lidar data. Remote Sens. 8 (8), 646.

Krawczyk, B., Minku, L.L., ao Gama, J., Stefanowski, J., Woźniak, M., 2017. Ensemble learning for data stream analysis: A survey. Inf. Fusion 37, 132–156.

Krawczyk, B., Woźniak, M., Herrera, F., 2015. On the usefulness of one-class classifier ensembles for decomposition of multi-class problems. Pattern Recognit. 48 (12), 3969–3982.

Krawczyk, B., Woźniak, M., Schaefer, G., 2014a. Cost-sensitive decision tree ensembles for effective imbalanced classification. Appl. Soft Comput. 14 (Part C), 554–562.

Krawczyk, B., Woźniak, M., Schaefer, G., 2014b. Cost-sensitive decision tree ensembles for effective imbalanced classification. Appl. Soft Comput. 14, 554–562. http://dx.doi.org/10.1016/j.asoc.2013.08.014.

Kuncheva, L.I., 2014. Combining Pattern Classifiers: Methods and Algorithms, Second edition. John Wiley & Sons, Inc.

Kuncheva, L.I., Whitaker, C.J., 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach. Learn. 51 (2), 181–207.

Lichman, M., 2013. UCI Machine Learning Repository . http://archive.ics.uci.edu/ml.

Loyola-González, O., Medina-Pérez, M.A., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A., Monroy, R., García-Borroto, M., 2017. PBC4cip: A new contrast pattern-based classifier for class imbalance problems. Knowl.-Based Syst. 115, 100–109.

Loyola-González, O., Martí nez Trinidad, J.F., Carrasco-Ochoa, J.A., Garca-Borroto, M., 2016. Effect of class imbalance on quality measures for contrast patterns: an experimental study. Inform. Sci. 374, 179–192.

Mingers, J., 1989. An empirical comparison of pruning methods for decision tree induction. Mach. Learn. 4 (2), 227–243.

Moreno-Torres, J.G., Saez, J.A., Herrera, F., 2012. Study on the impact of partition-induced dataset shift on k-fold cross-validation. IEEE Trans. Neural Netw. Learn. Syst. 23 (8), 1304–1312.

Pendharkar, P.C., 2016. Bayesian posterior misclassification error risk distributions for ensemble classifiers. Eng. Appl. Artif. Intell.

Quinlan, J., 1987. Simplifying decision trees. Int. J. Man-Mach. Stud. 27 (3), 221–234.

Quinlan, J., 2014. C45: Programs for machine learning. Morgan Kaufmann series in machine learning. Elsevier Sci..

Seiffert, C., Khoshgoftaar, T.M., Van Hulse, J., Napolitano, A., 2010. RUSBoost: A hybrid approach to alleviating class imbalance. IIEEE Trans. Syst. Man Cybern. A 40 (1), 185–197.

Visentini, I., Snidaro, L., Foresti, G.L., 2016. Diversity-aware classifier ensemble selection via f-score. Inf. Fusion 28, 24–43.

Woźniak, M., Graña, M., Corchado, E., 2014. A survey of multiple classifier systems as hybrid systems. Inf. Fusion 16, 3–17.

Zhang, Z.-L., Luo, X.-G., Garca, S.., Herrera, F., 2017. Cost-sensitive back-propagation neural networks with binarization techniques in addressing multi-class problems and non-competent classifiers. Appl. Soft Comput. 56, 357–367.

Zhou, Q., Zhou, H., Li, T., 2016. Cost-sensitive feature selection using random forest: selecting low-cost subsets of informative features. Knowl.-Based Syst. 95, 1–11.