applied sciences

MDPI

Article

Shapley Values as a Strategy for Ensemble Weights Estimation

Vaidotas Drungilas *, Evaldas Vaičiukynas, Linas Ablonskis and Lina Čeponienė

Department of Information Systems, Faculty of Informatics, Kaunas University of Technology, 44249 Kaunas, Lithuania

* Correspondence: vaidotas.drungilas@ktu.lt

Abstract: This study introduces a novel performance-based weighting scheme for ensemble learning using the Shapley value. The weighting uses the reciprocal of binary cross-entropy as a base learner's performance metric and estimates its Shapley value to measure the overall contribution of a learner to an equally weighted ensemble of various sizes. Two variants of this strategy were empirically compared with a single monolith model and other static weighting strategies using two large banking-related datasets. A variant that discards learners with a negative Shapley value was ranked as first or at least second when constructing homogeneous ensembles, whereas for heterogeneous ensembles this strategy resulted in a better or at least similar detection performance to other weighting strategies tested. The main limitation being the computational complexity of Shapley calculations, the explored weighting strategy could be considered as a generalization of performance-based weighting.

Keywords: machine learning; ensemble methods; Shapley value; performance weighting; privacy-preserving distributed learning

Citation: Drungilas, V.; Vaičiukynas, E.; Ablonskis, L.; Čeponienė, L. Shapley Values as a Strategy for Ensemble Weights Estimation. *Appl. Sci.* **2023**, *13*, 7010. https://doi.org/10.3390/app13127010

Academic Editors: Xiaofeng Ding and Pan Zhou

Received: 11 May 2023
Revised: 6 June 2023
Accepted: 8 June 2023
Published: 10 June 2023

1. Introduction

With increasing interest in machine learning technologies, the number of created and used machine learning models is increasing. The information technology sector also faces an ever-growing demand for digital data storage facilities, and approximately 50 massive new data centers appear each year, with some researchers warning of an imminent information catastrophe [1]. This growth may influence more and more parties to work using similar data and could require the ability to collaborate. Such collaboration could be achieved by federated learning [2] and privacy-preserving approaches, and also simply by combining the models into ensembles in distributed or local environments. The ensemble methods have been successful and brought many popular techniques today, such as bagging and boosting. Although most of these techniques usually use homogeneous model types in the creation of ensembles, ensembles with heterogeneous model types might as well be used to tackle machine learning tasks of similar nature.

Ensemble learning approaches aim at constructing a strong learner from a set of weak learners [3], with the most popular techniques nowadays being bagging and boosting [4]. These techniques often outperform a monolith single model but typically are applied as off-line algorithms, which still need centralized data repositories for ensemble building. Online ensemble-building techniques suitable for training on streaming data are also being developed, i.e., online AdaBoost [5] or online Arcing [6]. Ensemble-based approaches for adapting to concept drift are closely related to online ensembles and seek to give more emphasis to the most recent training data batch. One set of approaches is to maintain a fixed-size committee of classifiers with new base classifiers built on each arriving batch of data and included in the committee by replacing its worst-performing member. This is done after validation of new data, either directly, as in the streaming ensemble algorithm [6] using majority weighting, or indirectly, by using top k performance-weighted classifiers, as in the accuracy-weighted ensemble [7]. Another online ensemble learning approach [8] tries to achieve robustness to concept drift by adapting weights in a pool of online learners

and expanding or pruning the committee using an overall ensemble performance as a benchmark. This is carried out by using a dynamic weighted majority approach [9], where weights are normalized to always sum to one, and the individual weight is decreased if a base learner makes a mistake. Simultaneously, data privacy is a major concern in numerous areas, including in citizen science projects [10], where participants frequently reveal personal details. One of the approaches to overcome the fear of sharing sensitive data and, consequently, the lack of motivation for data sharing overall, could be privacy-focused variants of distributed learning [11–16]. One variant of such approaches is knowledge transfer architecture where an ensemble of models corresponds to the teacher and is trained on disjoint internal subsets of the sensitive data, whereas a student model is trained on public data to replicate ensemble predictions [17]. Such architecture could benefit from an ensemble weighting strategy.

The Shapley value [18], in the context of machine learning, is mostly used for feature selection or assessment of data quality, but it could also be applied as a weight selection approach. The goal of our research is to introduce a novel performance-based weighting scheme for ensembles based on the Shapley value. Even though both performance-based weighting and Shapley values are commonly used in the machine learning field, the combination of these two, to the best of our knowledge, has not been experimentally tested and presented as an ensemble weighting approach. To test the new ensemble weighting approach, we compare it to other weighting strategies: random weights, equal weights, simple performance-based and Shapley voting-based [19] weighting by using two large datasets for machine learning classification tasks from the banking domain. The proposed weighting strategy could be used for general ensembling purposes but especially in the knowledge transfer architecture domain to help in constructing a teacher model with strong privacy guarantees after the student model is learned as a weighted ensemble. We envision the medical, financial, and scientific domains as the primary areas of application when participants are few, but their proprietary data is large and sensitive. We are planning to use the proposed weighting strategy as an incentive mechanism in blockchain-powered PPML systems by continuing the work on previously presented research in this field [20,21]. Our proposed approach introduces two novel performance-based weight selection strategies.

This study introduces a novel performance-based weighting scheme for ensemble learning using the Shapley value. The proposed weighting strategy is applied to two large banking datasets using logistic regression and decision tree classifiers and has been benchmarked against other known approaches. The study explores the applicability of the proposed strategy not only with homogeneous but also with heterogeneous ensembles.

The rest of this paper is structured as follows. Section 2 provides an overview of existing weighting strategies and related research on using the Shapley value in the context of machine learning. Section 3 describes the methods used to experimentally test the proposed solution. In Section 4, the settings and parameters used in the experiment are introduced. The experiment results for two tested ensemble creation scenarios are presented in Section 5 as well as discussion on the advantages and drawbacks of the proposed strategy. Section 6 concludes the investigation.

2. Related Work

Machine learning models that are trained as a single classifier and then combined by merging their outputs into final predictions are called ensembles [22]. These ensembles are classified into two categories. Homogeneous ensembles consist of a single model type and heterogeneous ones are composed of more than one model type. The success of machine learning model ensembles could be attributed to prognostic diversity. This diversity comes from the possibility that each model could interpret the same training data in a different way and could result in models that complement one another, thus increasing the overall performance of an ensemble. Ensemble diversity may be obtained through input data sampling approaches such as bagging [4], variations in learner design, or adding a penalty to the outputs to encourage diversity. The bagging approach uses a random subset of data

to train classifiers that are later combined, using voting to obtain a final prediction. Decision tree classifiers are commonly used for the bagging approach. The ensemble weighting approach explored in our solution is closest to bagging [4] and divide-and-conquer [23] strategies due to the horizontal partitioning of the input space. It also allows variations in learner design through a multi-inducers strategy [23], where either the same type of base model can be used with different hyperparameters or a different type of model can be supported.

Ensemble weights can be equal, as in the majority voting [24] case, or based on individual models' accuracy, as in performance weighting [22,23,25] or optimized to maximize the accuracy of the whole ensemble on the validation set [22], as in search-based ensemble selection approaches. Our proposal could be viewed as an extension of the performance-based weighting strategy. The performance-based weighting process consists of two primary steps: (a) benchmarking of the model performance and (b) adjustment of the ensemble weights based on the benchmarking results. The fusion of model outputs into a final prediction can also be derived through stacking, as in meta-learning [21]. Model fusion through stacking employs a combination of weak learners with training data to train the meta-learning model that provides final predictions. Recently, a strategy for an optimal ensemble was proposed in [26] by combining the tuning of hyperparameters and weights for regression tasks. Effective weights could also help sort classifiers for ranking-based ensemble selection and filter out non-useful ones as a technique of ensemble reduction. Search-based approaches tend to outperform ranking-based ones with respect to accuracy [25], but due to the optimization involved in weight tuning, they are more computationally expensive.

Shapley value [18] was first introduced as a measure to fairly distribute coalition worth among the participants by measuring their contributions. In the context of Shapley values are used mostly to facilitate model-agnostic selection of the most significant features by measuring the importance of each feature to the final prediction [27–29]. There are attempts to measure the importance of input data through Shapley values [30,31]. Our approach applies Shapley values in measuring the contribution of each model (in a fixed set of models) to all possible variants of ensembles.

The closest study to our proposed approach would be [19] where a technique for quantification of the model contribution to the ensemble of classifiers was proposed. It calculates approximate Shapley values by considering classifier predictions on each individual dataset instance. The study has used three datasets for evaluation, demonstrating that their own Shapley value approximation method is close in performance to other Shapley value approximation methods. The authors of the study have also tested the performance of an ensemble selected by their algorithm. It outperformed other approaches in two datasets out of the three tested. Even though the authors apply similar strategies, our approach differs from [19] in that we use a less localized assessment of detection performance. In our approach, binary cross-entropy values of models are used for Shapley value calculations as opposed to using predictions and ground-truth values directly. The goal of the authors in [19] is to improve the ensemble performance at the level of individual prediction points, while our approach evaluates the contribution of ensemble members by using their performance measures as a basis for Shapley value calculations. The method proposed in [19] is also utilizing Shapley approximation methods, thus differing from our proposed approach of calculating Shapley values exactly. The authors of this study did not consider the possibility to use Shapley values produced by their method as an ensemble weight selection strategy. To test if this could be a viable solution, we compared our proposition with the strategy proposed by [19].

Shapley values [18] were also applied in the ranking-based ensemble selection approach [32] that was based on the inducted subgraph game from combinatorics. This approach evaluated the contribution of every available classifier by incorporating ensemble characteristics such as individual accuracies and group variety into a Shapley value. The researchers validated the effectiveness of the approach by comparing it to five other techniques using 26 UCI benchmark datasets using AdaBoost [4] with a decision stump as

their base learner. The results of this new ranking approach showed improvement over the original ensemble and outperformed other tested approaches in more than half cases. This approach evaluated the contribution of every available classifier by incorporating ensemble characteristics such as individual accuracies and group variety into a Shapley value. The difference in approach in [32] from ours is that we choose not to measure ensemble diversity explicitly.

Ensemble heterogeneity and the possibility to apply Shapley values to evaluate classifiers have also been explored by [33]. The authors proposed a heterogeneous ensemble model based on the generalized Shapley value and the Choquet integral. The proposed method employs a fuzzy measure that uses model accuracy and diversity to construct a machine learning classifier. The predictions for such a classifier are aggregated using the Shapley–Choquet integral. The authors prove that the proposed classifier can outperform four existing classifiers, as well as five homogeneous model ensembles and three heterogeneous model ensembles using four banking-related datasets. Similarly, as in our approach, authors applied their model in the banking-related domain for credit risk modeling and compared ranks of classifiers using the Friedman test. This method differs from our strategy due to the application of fuzzy measures and advocating the use of heterogeneous ensembles.

Recently, Shapley model ranking techniques have been applied in the field of federated learning [31]. The authors propose a method to approximate federated Shapley values with increased efficiency. The presented research also suggests that Shapley values could be used in a wide range of data evaluation tasks. The comparison of related research and their application areas are presented in Table 1.

Table 1. Comparison of methods employing Shapley value for data or model performance evaluation.

Article	Metrics Used	Shapley Calculation Approach	Application Area
Benedek Rozemberczki [19]	Prediction voting	Expected marginal contributions approximation	Method to quantify the model importance in an ensemble
Hadjer Ykhlef [32]	Classifier accuracy	Exact calculations	Ensemble selection method
Tianhao Wang [31]	Classifier accuracy	Permutation sampling-based approximation, group testing-based approximation	Method to evaluate data importance in federated learning approaches
Xiaohong Chen [33]	Generalized Shapley Choquet integral	Exact calculations	A heterogeneous ensemble classifier for credit risk management
Our proposal	Binary cross-entropy	Exact calculations	Ensemble weight selection strategy

Our proposal differs from existing approaches by two main aspects: (a) weighting strategies utilize binary cross-entropy in combination with Shapely value calculations, as opposed to relying on individual predictions on a case-based level; and (b) Shapley value is calculated by utilizing empty coalition as a benchmark value corresponding to expected binary cross-entropy of random guessing with respect to class imbalance.

3. Methods

This section presents two proposed weighting strategies and their evaluation methods. It also describes datasets and data preparation measures utilized in the experiments. The last subsection presents a description of performance comparison methods that were applied to the experiment results. The general process representing the workflow of our proposed approach is presented in Figure 1. The overview of the proposed weight selection strategy is presented in Figure 2, demonstrating the data division into batches, model training and weighting processes. The weight selection strategy produces a weighted ensemble that produces final predictions.

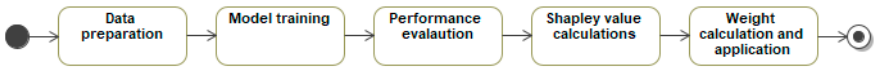

Figure 1. Workflow of the proposed ensemble weight calculation process.

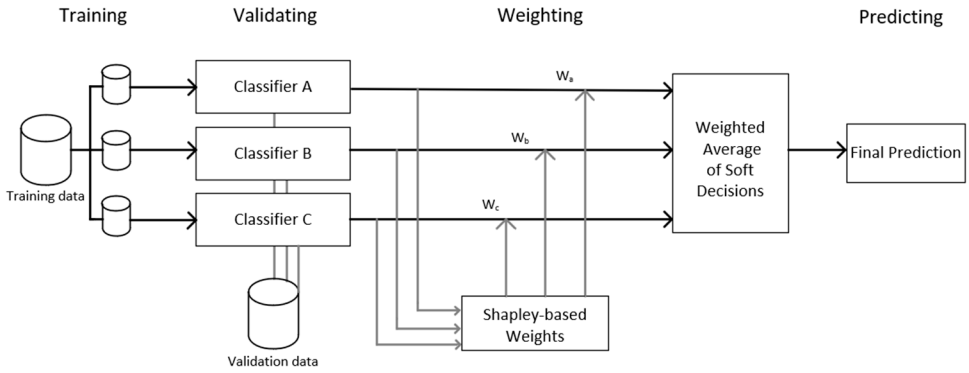

Figure 2. Overview of the proposed ensemble weight calculation strategy.

3.1. Base Learners

We have used two types of classifiers as base learners in the experiments: CART (classification and regression tree) and binary logistic regression. These two classifier types were chosen because they are generally widely known and are commonly applied for machine learning classification tasks, such as, for example, logistic regression in the banking industry [34]. Other classifiers [35,36] would be also suitable, as long as they can predict class probabilities. In our research, these learners are used to train either a single model on a full training dataset meaning that the training dataset was not divided into smaller batches (such model will be called a monolith) or multiple models that are trained on a part of training data and are further combined into ensemble.

The CART classifier represents a traditional decision tree [37]. The leaf nodes contain predicted values of the class of the input vector and the intermediate nodes contain binary decisions, each rejecting a subset of potential sequential points and classification choices. The tree is built by applying a greedy split algorithm over the learning data by minimizing the Gini impurity (1); more specifically, the best split gives a minimal weighted average of class impurity in child nodes. The Gini impurity is calculated as:

$$\mathrm{Gini}(k) = \sum_{i \subseteq C} \rho_{i,k}(1 - \rho_{i,k}) = 1 - \sum_{i \subseteq C} \rho_{i,k}^2 \tag{1}$$

where C is a list of all classes, k corresponds to a specific category (child node) after splitting and $\rho_{i,k}$ is a probability of category k having class i. The main advantage of decision trees over other classifiers is that the "reasoning" behind the classification output can always be described in human-friendly terms. An illustrative example of the Bank Marketing Dataset used in our work is provided in Figure 3. Our approach differs from the random forest ensemble learning method by not using bootstrap aggregation and combining classifiers trained on datasets with varying sizes respecting data size distribution patterns defined in Section 3.3.

Binary logistic regression is also one of the traditional approaches [38] to the problem of binary classification. It models the linear expectation of vector $X = \{x_1, x_2, \ldots, x_n\}$ belonging to class A as $E(A|X)$, where:

$$E(A|X) = \frac{e^y}{1 + e^y} \tag{2}$$

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \cdots + \beta_n * x_n \tag{3}$$

The logistic regression model fitting weighting coefficients $\{\beta_0, \beta_1, \beta_2, \cdots, \beta_n\}$ in a manner that best satisfies the logistic loss (log-loss) function:

$$cost(P(A|X), X) = \begin{cases} -log(P(A|X)), if\ X\ is\ in\ class\ A \\ -log(1 - P(A|X)), if\ X\ is\ not\ in\ class\ A \end{cases} \quad (4)$$

The feature selection step and regularization were omitted from the logistic regression model, as the goal of this research was to compare existing weighting strategies to two novel approaches.

The performance of models for detection tasks in our experiments was evaluated using binary cross-entropy metric [39]. Cross-entropy is one of the more popular loss functions used for classification problems. It quantifies the difference between two probability distributions—the predicted output distribution and the real output distribution. Binary cross-entropy (*BCE*) can be seen as a generalization of log-loss for multi-class cases and is defined as:

$$BCE\left(y, \hat{y}\right) = -(y \cdot log(\hat{y}) + (1 - y) \cdot log(1 - \hat{y})) \quad (5)$$

The y represents the ground-truth class label and $\hat{y}$ corresponds to the model's prediction of class probability.

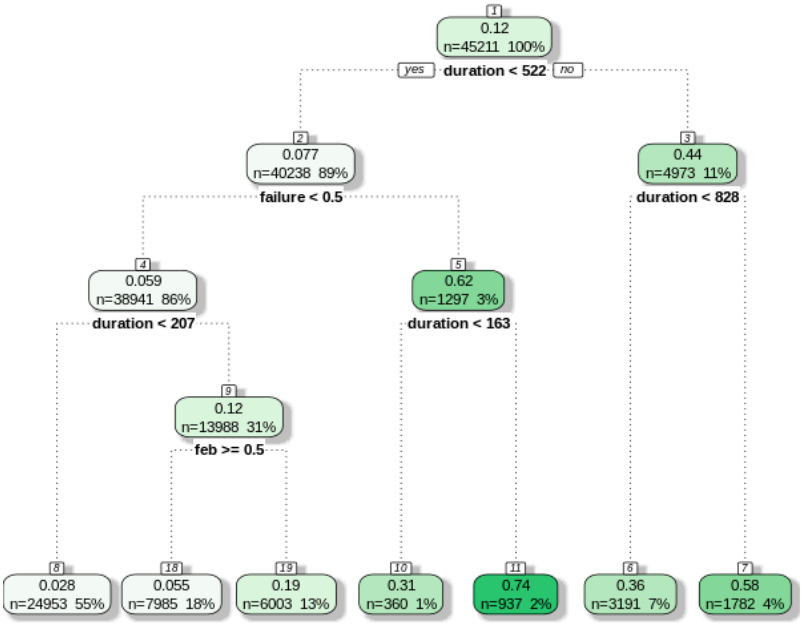

Figure 3. Decision tree visualization for Bank Marketing dataset, where the darker the color of the node the more target class cases (clients subscribed to the service) of the training data exist there.

3.2. Weighting Strategies

We have used five ensemble weight estimation strategies: Equal, Random, Perf, posShap, and maxShap. The purpose of the Equal weight strategy was to determine how well the model performs in an ensemble without any effort to estimate the weights. All ensemble members receive the same values under the Equal weighting approach. In the Random weighting strategy, each model gets its value randomly generated in a range from 0 to 1, which constitutes a baseline to compare against other weighting heuristics. Performance weighting (Perf) was calculated using Function (6), where *BCE* measure corresponds to Formula (5) and was calculated on validation data (Figure 4). This corresponds to the reciprocal of *BCE*:

$$w_n^{Perf} = \frac{1}{BCE} \quad (6)$$

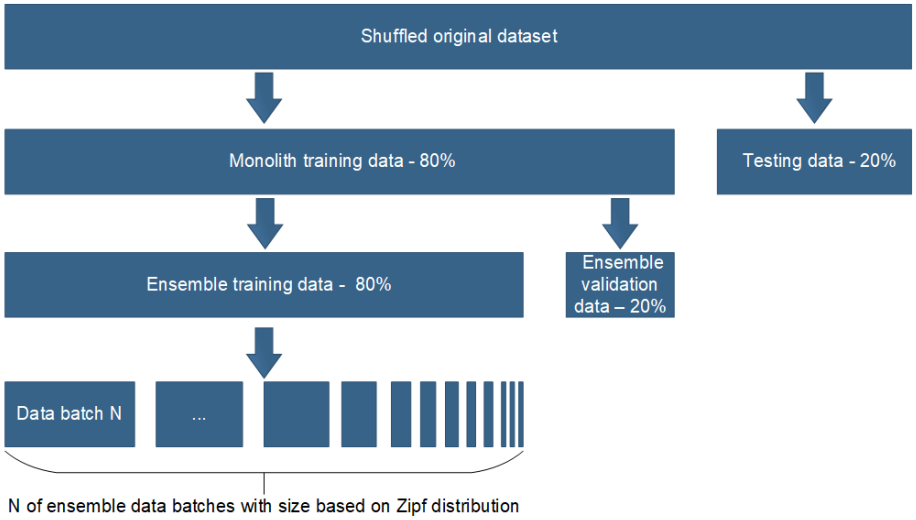

Figure 4. Double hold-out validation scheme with Zipf-derived data splitting for training ensemble members.

Shapley value was first introduced as a way to evaluate coalition participants' contribution to an overall coalition in a game theory paper [18]. Since then, Shapley values have been adopted in many research fields such as economics [40] and machine learning [19,29,30]. In our proposed weighting strategy, the Shapley transformation of performance-based weight for model n was calculated as follows:

$$w_n^{Shap}(v) = \sum_{S \subseteq N \setminus \{n\}} \frac{|S|!(N - |S| - 1)!}{N!} (v(S \cup \{n\}) - v(S)) - \varnothing \quad (7)$$

with reciprocal *BCE* corresponding to coalition members' contribution v. N is the set of classification models and sub-ensemble S is a subset $S \subseteq N$. $(S \cup \{n\}) - v(S)$ represents a marginal contribution of a single coalition member. The empty coalition $\varnothing$ value was calculated using Formula (7) with all predictions fixed at 0.5 and tested against ground-truth values of validation data, which in effect provides a performance baseline derived with respect to existing class imbalance. It is well known that a baseline threshold of *BCE* in the case of balanced classes is exactly 0.693 and *BCE* exceeding this value indicates that a corresponding detector is performing worse than chance [41]. Moreover, this threshold value decreases in case of a stronger class imbalance. Therefore, a need to account for that when using Shapley estimation with a possibility to assign value to an empty coalition arises.

The posShap weighting strategy calculated weights based on Formula (7) and models that resulted in negative Shapley values were not included in the ensemble by setting their weight to 0:

$$f(w) = \begin{cases} w, & w \geq 0 \\ 0, & w < 0 \end{cases} \quad (8)$$

where w is obtained using Formula (8) and posShap transformation of w is identical to rectified linear unit activation function ReLU [42].

The maxShap weights were calculated in the case of negative Shapley values, which meant the net loss with respect to the overall coalition. For such models, the assumption was made that the classes were switched when training a model or that some labeling noise was present in the training data. To allow such models to be used in a model coalition, we have inverted the model predictions as follows:

$$f(p) = \begin{cases} p, & p \geq 0 \\ 1 - p, & p < 0 \end{cases} \quad (9)$$

where p is the output from the detection model. This was supposed to enable a model that was trained on mislabeled data to still be useful in the overall coalition to the extent of its contribution if it was due to label noise.

The voting-based Shapley weights Roz were calculated using the method described in [19]. This method uses marginal contribution approximation to speed up the calculations and calculates Shapley values at individual data point levels. To transform the individual data point level Shapley values to ensemble weights we average these values into their weight representation using Formula (10):

$$w_{\mathrm{n}}^{\mathrm{Roz}} = \frac{1}{N}\sum_{i=1}^{N}\varphi_i \tag{10}$$

where N is the number of model predictions and n is the model number in an ensemble and φ represents Shapley value for individual data point levels.

3.3. Data Preparation

The five ensemble weighting strategies were tested using two datasets: Bank Marketing and BNG-Credit_a. Data characteristics and dataset sources are described in Table 2. The Bank Marketing dataset was also used for experiments in [32]. Preparation of the Bank Marketing dataset consisted of transforming targeted value from "Yes" and "No" to 1 and 0, respectively, and one hot encoding of categorical data. One hot encoding transformed categories into an expanded feature set. The BNG-Credit_a dataset features A6, which represented bank customer occupation, and A7, which represented the last known month the customer was contacted including many unique categories. To reduce the number of categories, categories that had less than 25,000 entries for feature A6 and 89,044 entries for feature A7 were aggregated into the new "other" category. After reducing the number of categories in A6 and A7, all categorical data was one hot encoded resulting in new features. The training data was shuffled randomly based on the seed provided for each separate iteration.

Table 2. Experiment dataset characteristics (number of features was calculated excluding the target value). Total features were obtained by using all numerical features and converting categorical features into dummy types by one hot encoding.

Data Characteristic	Bank Marketing	BNG-Credit_a
Initial features	16	15
Categorical features	10	10
Total features	51	33
Total instances	45,211	1,000,000
Classes	2	2
Target class proportion	0.12	0.544
Dataset sources	Bank Marketing [43]	BNG_credit-a [44,45]

3.4. Evaluation Scheme

The hold-out validation consisted of two steps. First, shuffle data and split it into 80% and 20% parts, where 80% would be used for training a monolith or providing it for ensembling, and 20% would be held out for testing purposes to enable a fair comparison between comparison monolith and ensembling strategies. After the initial data split, the second step was done on non-testing data, again, taking 80% for training ensemble models and leaving 20% as validation data for weight estimation in performance weighting and Shapley-based proposal. During the last step of the hold-out strategy, the data dedicated to training ensemble models were split into smaller model-wise batches based on Zipf's law distribution with its exponent value set to 0.2. Each model was trained on its individual data batch and tested on validation data to obtain a *BCE* estimate of its performance.

Zipf's distribution was used to partition the training dataset into chunks of varying sizes, and thus introduce non-IID nature into the data splits used for training base models. The non-IID nature was desirable because it is observed that, in general, datasets used in real-world federated learning are non-IID (non-independent and identically distributed) [46]. To simulate non-IID conditions in artificial environments, training datasets are usually partitioned into chunks where either the chunk size or class distribution inside the chunks follows some variant of power law $Y = kX^a$. Here, k is a scaling factor, X is a chunk index, a is the exponent factor and Y is either the chunk size or the number of classes reflected in a chunk or some other property of the chunk relevant to the training. Some authors choose to select power law coefficients in an arbitrary way, i.e., [47–49], while some choose to use one of the specific power law variations, with Zipf's distribution being one example [50–52].

3.5. Performance Comparison

The classifier performance comparison was performed using the cd-diagram library [53] and using similar techniques to the ones presented in [54]. Friedman's test [55] was used to detect whether there is a statistical difference between the performance of the classifiers compared. If the Friedman test showed that classifiers are statistically different, pairwise analysis was performed as recommended in [56] by replacing the average rank comparison with the Wilcoxon signed-rank test [57] corrected with Holm's alpha correction [58]. In critical difference diagrams (see diagrams in Section 4.3, e.g., Figure 6), the bold line between the classifiers displays a lack of statistical difference between the results presented by these two classifiers. Similar ranking approaches also appear as the choice of result representation in [32].

4. Results and Analysis

4.1. Experimental Setup

The main goal of the experiments was to compare our proposed weight estimation techniques with existing ensemble weighting techniques and a monolith model. To reach this goal, we have conducted two experiments that tested different types of ensembles: homogeneous ensembles consisting of either logistic regression or decision tree classifiers and heterogeneous ensembles that used a mixture of both classifiers. Each experiment consisted of three main parts: model training, ensemble weight selection and ensemble performance evaluation. During the model training step, individual models were trained. The weight estimation step consisted of individual model performance evaluation on validation data and calculation of weights using a selected scheme. Finally, the models were joined into ensembles by using a weighted average of their predictions. All ensembles were evaluated on reserved testing data for proper comparison of detection performance and results were analyzed using the Friedman test. The monolith model differed from the ensembles that it consisted of a single model created using a training dataset that was not divided into smaller batches, as shown in Figure 4.

The experiments were conducted on a virtual machine with an Intel Xeon Silver 4114 CPU running at 2.20 GHz that contained 10 CPU cores and featured 32 gigabytes of RAM with SSD storage. The experiment environment was developed using Ubuntu 18.04 operating system. Two execution environments, R 4.1.3 and Python 3.6.9, were used for implementation. We chose Python and R programming languages based on their popularity in the ML field and to demonstrate that the proposed method could be applicable in systems that use components with heterogeneous environments. By testing this approach on two popular machine learning languages, we demonstrate that the method is not hard to replicate in different settings and provide additional insight into the suitability of weighting strategies, irrespective of the implementation platform. The R implementation used the machine learning library MLR3 version 0.13.3. The Python implementation used the machine learning library PySpark version 3.1.2. The parameters used for each mode

type are listed in Table 3, showing unified parameter values where possible, as well as a few implementation-specific settings.

Table 3. Hyperparameter values for different model implementations.

Model Type	Common Parameters	Implementation-Specific Parameters
MLR3 LR [1]	E = e × 10^8 iterations = 25	singular.ok = True; trace = False
PySpark LR [1]		regParam = 0.0; aggregationDepth = 2; threshold = 0.5; elasticNetParam = 0.0; fitIntercept = True
MLR3 DT [2]	maxDepth = 30 minInfoGain = 0.01	minSplit = 20; maxcompete = 4; maxsurrogate = 5; surrogatestyle = 0; usesurrogate = 2; xval = 10

PySpark DT [2]		minInstancesPerNode = 20; Standardization = False;
		minWeightFractionPerNode = 0.0; minInstancesPerNode = 1;
		maxBins = 32; minInfoGain = 0.0; impurity = 'gini'

[1] LR—logistic regression. [2] DT—decision tree.

Both MLR3 and PySpark were used to train models, perform model inference and store inference results in files. The results of trained models were later used in an ensemble creation environment implemented using caret 6.0, kappalab 0.4 and matrixStats 0.61, which summarized inference success by *BCE*, calculated Shapley values, and combined output from model inference into a weighted ensemble. One hundred iterations of the model training process were performed. The implementation provided in [59] was used to test the method described in [19]. We used marginal contribution approximation as recommended by Ref.'s [19] research with the quota parameter set to 0.35. The Shapley values for this method were calculated for individual predictions and then aggregated by averaging all individual model's Shapley values. In cases when this method was unable to produce Shapley values they were substituted with equal weights. The heterogeneous experiments used the same models that were created for homogeneous ensembles, but mixed different model types into a single ensemble. Monolith (Mono) approach was calculated separately by providing model training libraries with undivided training data and training a single model. To enable a comparison of overall performance, the Mono strategy that presented the best performance in a homogeneous experiment was also added to the experiment results. Experiments were conducted with multiple ensemble sizes: {2, 3, 5, 8, 13} for homogeneous ensembles and {4, 6, 10, 16} for heterogeneous ensembles. Data hold-out was performed using the R script with Caret package version 6.0-90.

4.2. Experimental Results

The results of the experiment will be presented using two diagrams: the line diagram displays how *BCE* error values change with increasing ensemble size and the critical difference diagram ranks models after pooling the *BCE* results of all tested ensemble sizes. Detailed results of the experiments are available online [60]. The experiments compared the monolith approach (Mono) to five ensemble weighting strategies: equal weight strategy (Equal), performance-based (Perf), randomly generated (Rand), voting-based Shapley [19] (Roz), positive Shapley (PosShap) and maximum Shapley (MaxShap). The monolith approach Mono used the undivided training and testing dataset, providing a base value for model capabilities without the introduction of ensembling. Due to a sufficient amount of data, members of ensembles built on the BNG-credit_a dataset did not produce negative Shapley values; thus, the MaxShap weighting strategy was not applied in relation to this dataset. Critical difference diagrams evaluate the statistical significance overall of the ensemble sizes tested.

4.3. Results of Homogeneous Model Ensembles

The homogeneous model ensembles were created using a single type of model. The two model types chosen were decision tree and logistic regression. The results of logis-

tic regression ensembles for the BNG-credit_a dataset are presented in Figure 5. These results indicate that the monolith approach had the lowest *BCE* of 0.326. All ensemble weighting strategies exhibited similar results across all evaluated ensemble lengths. The critical difference diagram in Figure 6 shows an overview of the aggregated rankings for all tested ensemble configurations and notes that the equivalent model rankings were produced by both Python and R implementations. These model ranks specify that there was no statistical significance difference between the other weighting strategies and the Roz strategy indicating that the gains provided by ensembles using logistic regression model type we minimal. A Mono approach had the highest rank with posShap ranked as the second most successful.

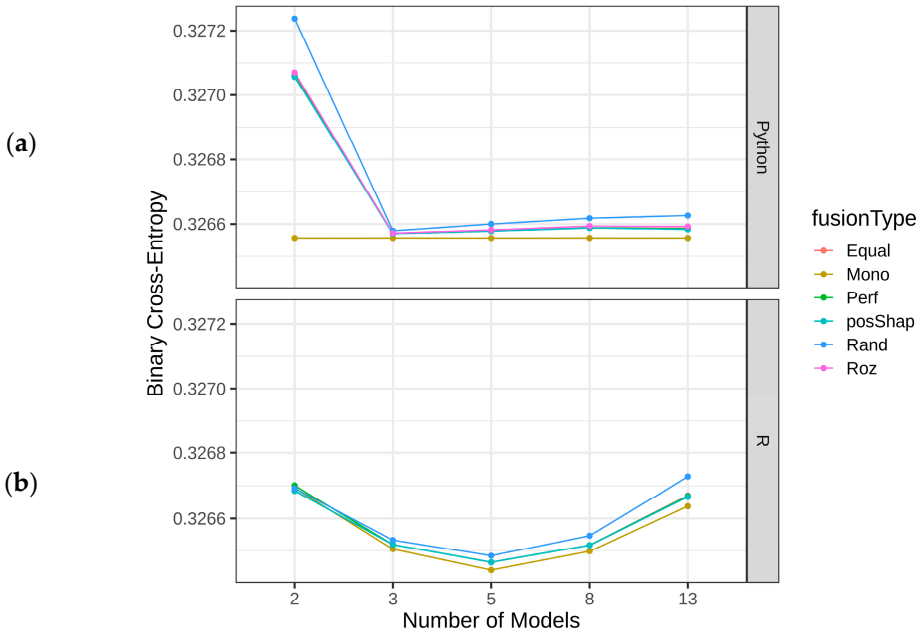

Figure 5. Detection error by *BCE* for various homogenous ensemble sizes: logistic regression in Python (**a**) and R (**b**) using BNG_credit-a dataset.

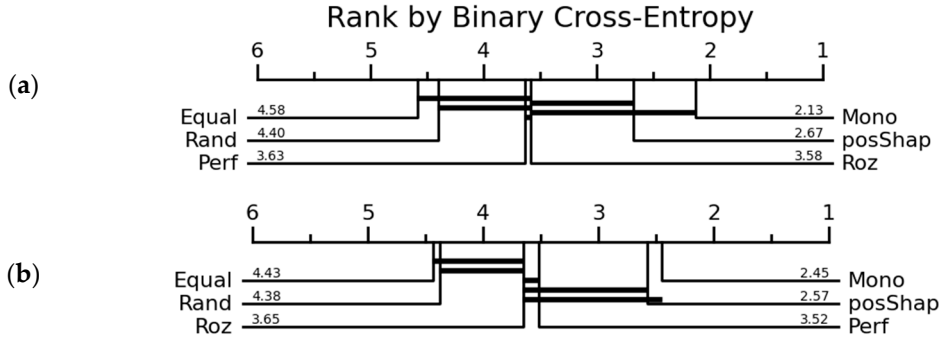

Figure 6. Ranking of homogenous ensembles: logistic regression in Python (**a**) and R (**b**) using BNG_credit-a dataset.

Homogeneous decision tree ensembles in Figure 7 indicate that all ensemble weighting strategies surpassed the performance of the Mono approach. The difference between weighted ensembles and the Mono approach increased on higher ensemble member counts. This difference increased from 0% to 4.8% for the best-performing posShap weighting strategy. If we compare the performance of only weighting strategies, the results reveal that if the number of models in an ensemble is small (2 or 3), all tested strategies perform similarly, when the number of models is greater than 5 posShap outperforms other approaches with the best result of 0.317 *BCE*. The performance difference between other

strategies and posShap also appears in the critical difference diagram in Figure 8 where the posShap strategy obtained the highest rank. The results also reveal that there is no statistical significance between Equal and Roz strategy results, as well as Perf and Roz strategies for the Python implementation. The results for the R language implementation produced the highest performance gains of all experimentally tested solutions. When compared to a single model approach Mono on the largest number of ensemble member (13) setting, ensemble weighting increased the performance with 4.1% and 4.8 % for Perf and posShap weighting strategies, respectively.

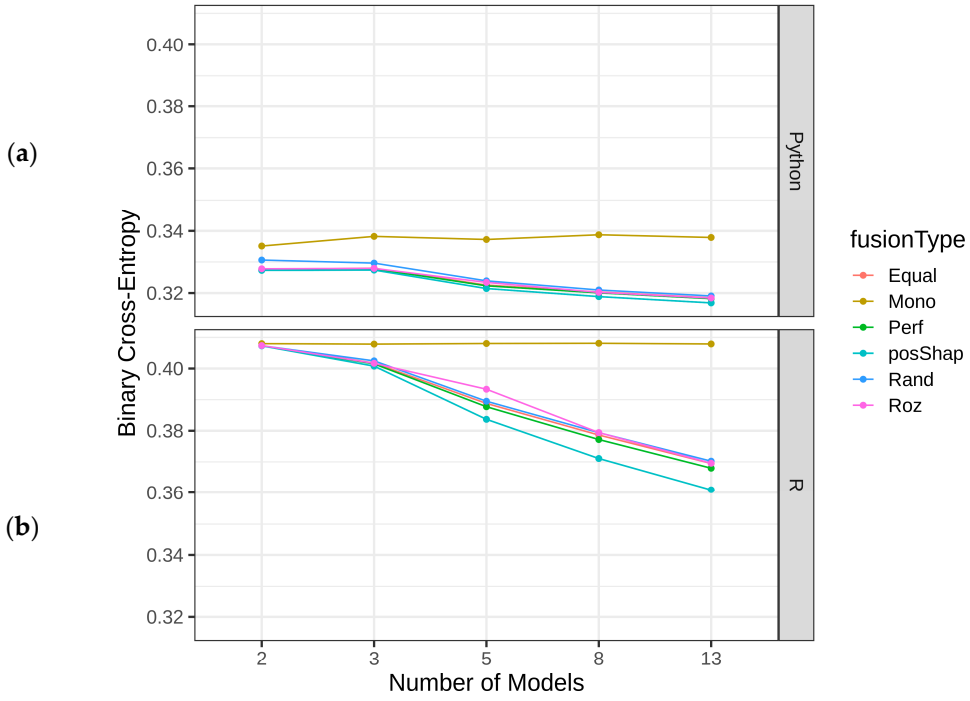

Figure 7. Detection error by *BCE* for various homogenous ensemble sizes: decision tree in Python (**a**) and R (**b**) using BNG_credit-a dataset.

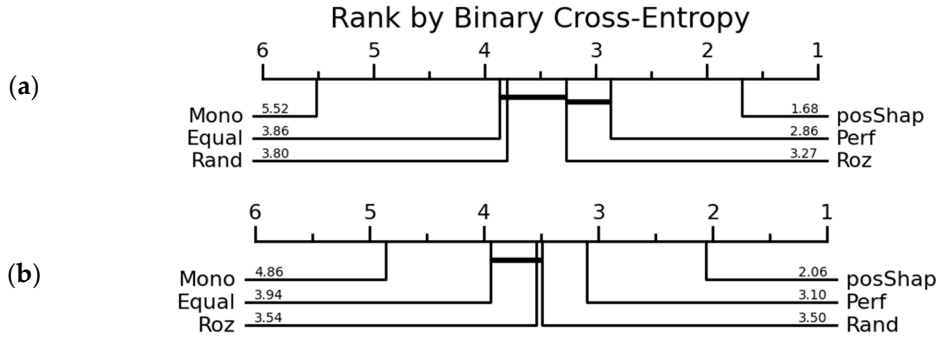

Figure 8. Ranking of homogenous ensembles: decision tree in Python (**a**) and R (**b**) using BNG_credit-a dataset.

Logistic regression ensembles for the Bank Marketing dataset (Figure 9) indicate similar results for ensembles with 2, 3 and 5 member counts over all tested strategies. For ensembles sized 8 and 13, there was a major increase in *BCE* for the maxShap weighting approach. When the ensemble member count reached 8 members, the distinction between Mono and other strategies became apparent, but the difference remained minimal at only less than 0.1%. Of all tested cases, the Perf weighting approach showed the best performance of 0.236 *BCE*. The weighting strategy comparison via the critical difference diagram in Figure 10 reveals that the posShap strategy had the best performance, while the

Perf strategy gave the second-best results. Both posShap and Perf were ranked better than the Mono approach indicating a performance boost of weighting. Although the results indicate a clear performance boost of weighting strategies, the 0.4% gain compared to the monolith approach could be considered trivial this also explains no statistical significance between the most tested weighting strategies.

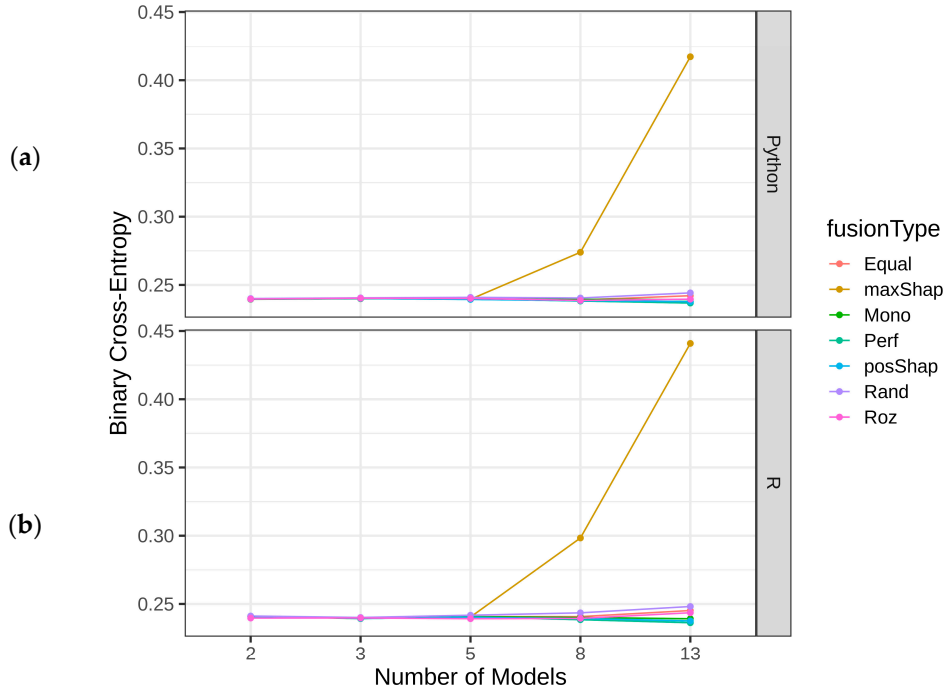

Figure 9. Detection error by *BCE* for various homogenous ensemble sizes: logistic regression in Python (**a**) and R (**b**) using Bank Marketing dataset.

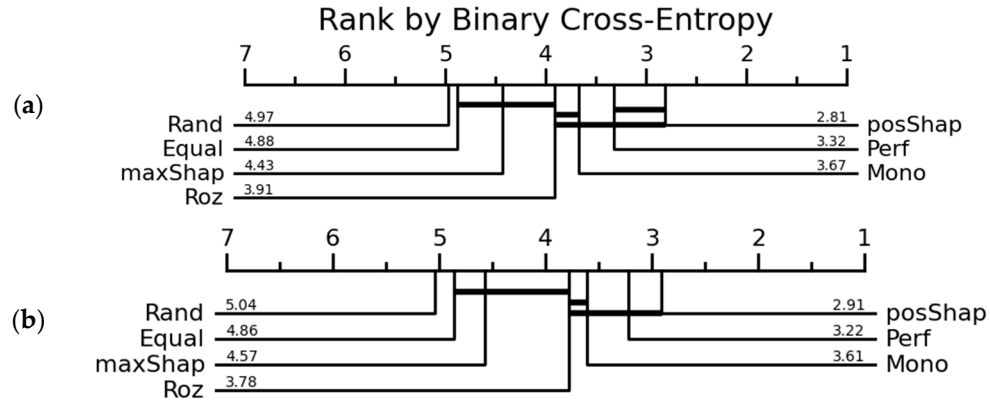

Figure 10. Ranking of homogenous ensembles: logistic regression in Python (**a**) and R (**b**) using Bank Marketing dataset.

Decision tree model ensembles for the Bank Marketing dataset (Figure 11) demonstrate that all weighting strategies surpassed the monolith approach, even in the case of very small ensembles. When the ensemble contained 3 or more members, the *BCE* of maxShap classifiers performance decreased and kept on diminishing with higher member counts. The results of Python implementation revealed that ensembles with member counts of 2 to 8 outperformed the monolith approach, and for larger ensembles of 13 members only, the posShap strategy exhibited better results of 0.264 *BCE* for Python and 0.274 *BCE* for R implementations. As for experiment implementation using R language Perf, Equal and posShap weighting strategies presented a lower *BCE* than the monolith approach for all

evaluated model counts. The results also indicate that the Perf ensemble weighting strategy with member count 8 produced the best result of 0.251 *BCE*. The posShap approach was ranked first for the R implementation and second for the Python implementation, as can be seen in the critical difference diagram (Figure 12). In both implementations, the results were not statistically significant between Random and Roz strategies as well as the PosShap and Equal approaches in Python implementation.

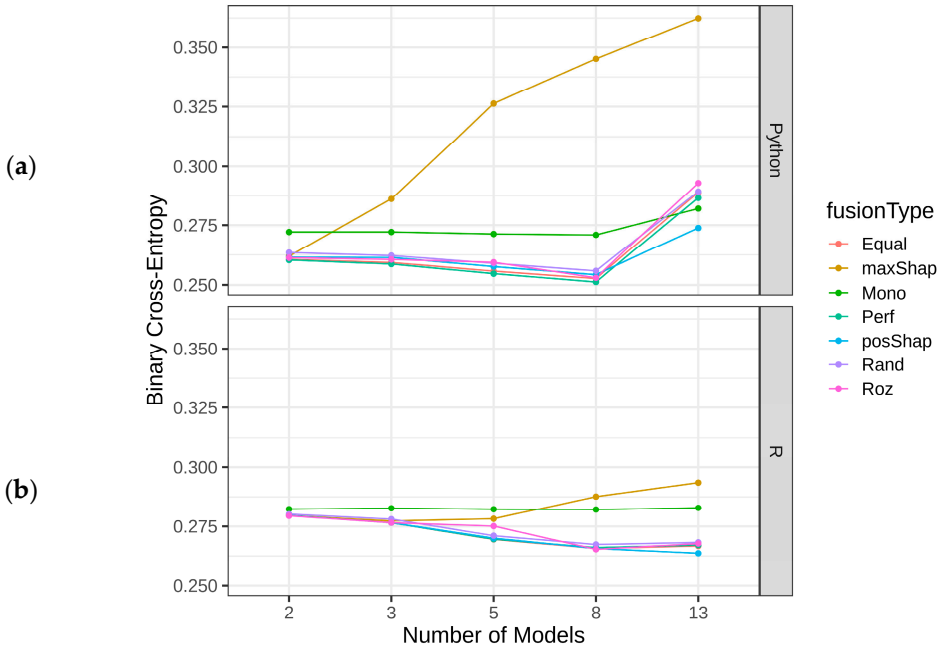

Figure 11. Detection error by *BCE* for various homogenous ensemble sizes: decision tree in Python (**a**) and R (**b**) using Bank Marketing dataset.

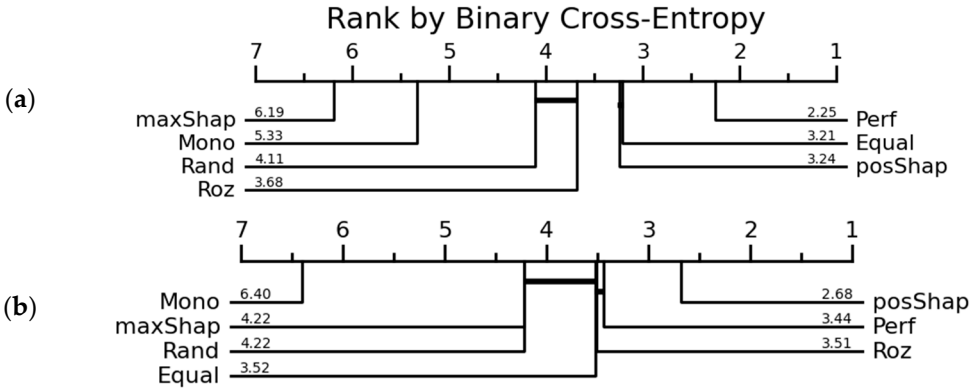

Figure 12. Ranking of homogenous ensembles: decision tree in Python (**a**) and R (**b**) using Bank Marketing dataset.

The results of the homogeneous ensembles experiment results presented in Table 4 indicate that ensembles composed of the decision tree classifier benefited more from weighting than the logistic regression classifier ensembles. Decision tree-based ensembles exhibited a performance increase when compared to the monolith approach by 1.9 % for Bank Marketing and 4.8% for BNG_credit-a datasets, whereas the performance of logistic regression ensembles resulted in only a trivial gain of 0.2 % and 0.002%. Although the benefit of weighting was not as effective in logistic regression ensembles, posShap weighting still performed very similarly to the Perf strategy, which outperformed other weighting strategies. The lowest *BCE* of 0.236 for the Bank Marketing dataset was exhibited by the Perf weighting strategy with the largest ensemble size of 13, where the posShap

strategy presented the second-best result of 0.238 *BCE*. Similar experiment results were exhibited for the BNG_credit-a dataset where posShap and Perf performance was 0.317 and 0.318 *BCE*. The effectiveness of the posShap weighting strategy was more apparent when applied to ensembles with a larger number of members. In contrast to posShap, the maxShap weighting strategy performed the worst of all tested strategies for larger ensembles of sizes 8 and 13. In all tested experiment settings our approach surpassed or at least performed similarly to the weighting strategy Roz [19]. A comparison of implementations between two different programming languages revealed that even though some minor differences in results were evident, general trends and ensemble performance insights were consistent.

Table 4. Rank comparison from Friedman test for homogeneous experiment results with lowest rank value highlighted in bold.

Model Type	**LR** [1]		**DT** [2]		**LR** [1]		**DT** [2]	
Dataset	**BNG** [3]		**BNG** [3]		**Bank** [4]		**Bank** [4]	
Implementation / **Weighting Strategy**	**Python**	**R**	**Python**	**R**	**Python**	**R**	**Python**	**R**
Mono	**2.13**	**2.45**	5.52	4.86	3.67	3.61	5.33	6.40
Rand	4.40	4.38	3.80	3.50	4.97	5.04	4.11	4.22
Equal	4.58	4.43	3.86	3.94	4.88	4.86	3.21	3.52
Perf	3.63	3.52	2.86	3.10	3.32	3.22	**2.25**	3.44
Roz	3.58	3.65	3.27	3.54	3.91	3.78	3.68	3.51
MaxShap	-	-	-	-	4.43	4.57	6.19	4.22
PosShap	2.67	2.57	**1.68**	**2.06**	**2.81**	**2.91**	3.24	**2.68**

[1] LR—logistic regression. [2] DT—decision tree. [3] BNG—BNG_credit-a dataset. [4] Bank—Bank Marketing dataset.

4.4. Heterogenous Ensembles

The homogeneous experiment demonstrated weighting capabilities applied to using identical types of machine learning models. In real-world applications, researchers may prefer using different model types to increase the ensemble's diversity and improve its generalization, but also due to varying capabilities [61] of forming internal representations from features available. In the following experiment, weighting strategies were applied to heterogeneous ensembles. The logistic regression monolith model was chosen here as a baseline due to the best performance in the homogeneous experiment.

The results for heterogeneous ensembles trained on the Bank Marketing dataset using the Python implementation (Figure 13) were similar to the results presented in the R implementation. The equal ensemble weighting strategy demonstrated the lowest *BCE* overall with the best performance in an ensemble with 16 models that exhibited 0.233 *BCE*. The maxShap weighting strategy displayed reduced performance over ensembles with a higher number of models and exhibited the worst performance overall. The opposite could be observed for the results of the posShap weighting strategy, although it performed similarly to the Mono approach in ensembles with 4 and 6 members. Ensembles with 10 and 16 members outperformed the Mono approach. The posShap strategy also outperformed Perf when the number of models in the ensemble was 16. The posShap advantage over the Perf strategy demonstrates that Shapley values are able to measure model contribution in greater detail. This advantage stems from the evaluation of all possible permutations revealing combinations that cannot be considered by the simple performance metric of a single model. The ranking of weighting strategies (Figure 14) also reveals that in this scenario, the Equal strategy outperformed other tested approaches. The Rand approach was ranked as second best, and Perf and posShap were ranked third and fourth, respectively. The Mono and maxShap ensemble weighting strategies were the least performant.

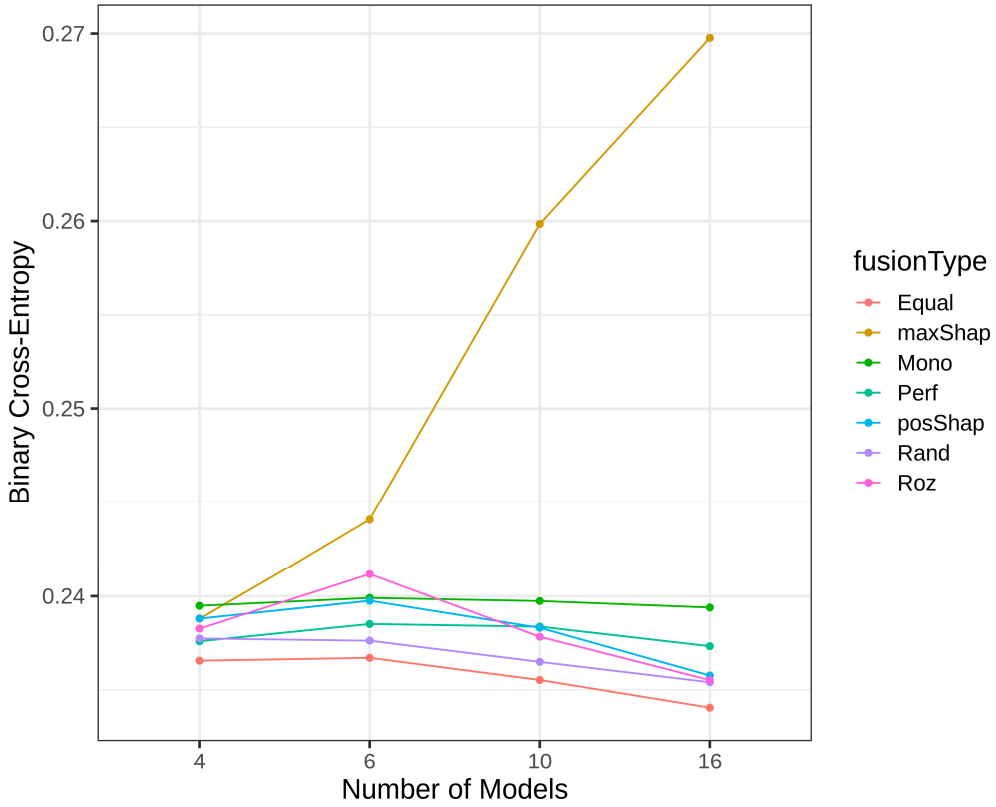

Figure 13. Detection error by *BCE* for various heterogenous ensemble sizes using Python models and Bank Marketing dataset.

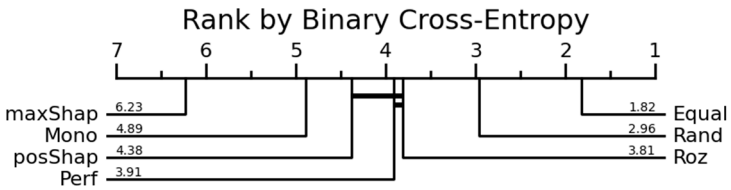

Figure 14. Ranking of heterogenous ensembles: using Bank Marketing dataset and Python implementation.

Using the same Python implementation and training the heterogeneous weighted ensembles on the BNG-credit_a dataset, the results (Figure 15) were similar to homogeneous ensemble implementation (Figure 7) for the same dataset. The introduction of logistic regression models further reduced the *BCE* for ensembles with applied weighting strategies and improved the performance of ensembles. The results of the posShap weighting strategy were the best out of all tested strategies, resulting in a median *BCE* of 0.312. The Perf and Equal strategies resulted in a somewhat similar performance. The critical difference diagram (Figure 16) did not show a statistically significant difference between these two approaches. The model ranking also revealed that the monolith strategy was ranked the lowest, meaning that all the ensembling strategies increased their performance over their homogeneous counterparts. For ensembles of 16 models, the performance gain over Mono ranged from 1.3% for Equal, Perf and Rand strategies to 1.4 % for posShap strategy.

The heterogeneous ensemble results (Figure 17) for the Bank Marketing dataset using R language implementation demonstrated that for ensembles with 10 and 16 members, the posShap weighting strategy surpassed the performance of the monolith approach, but the best-performing weighting strategy was Equal with 0.233 *BCE*. With the ensemble consisting of 10 or more members, posShap outperformed the Perf strategy and produced the same median as the Rand strategy of 0.235 *BCE*. Although the weighting performance

gains were only 0.6%, all weighting strategies, except maxShap, outperformed the Mono approach. The ranking of ensembles (Figure 18) revealed that the Mono strategy was ranked as the best-performing one, with posShap and maxShap methods ranked as the least-performing strategies.

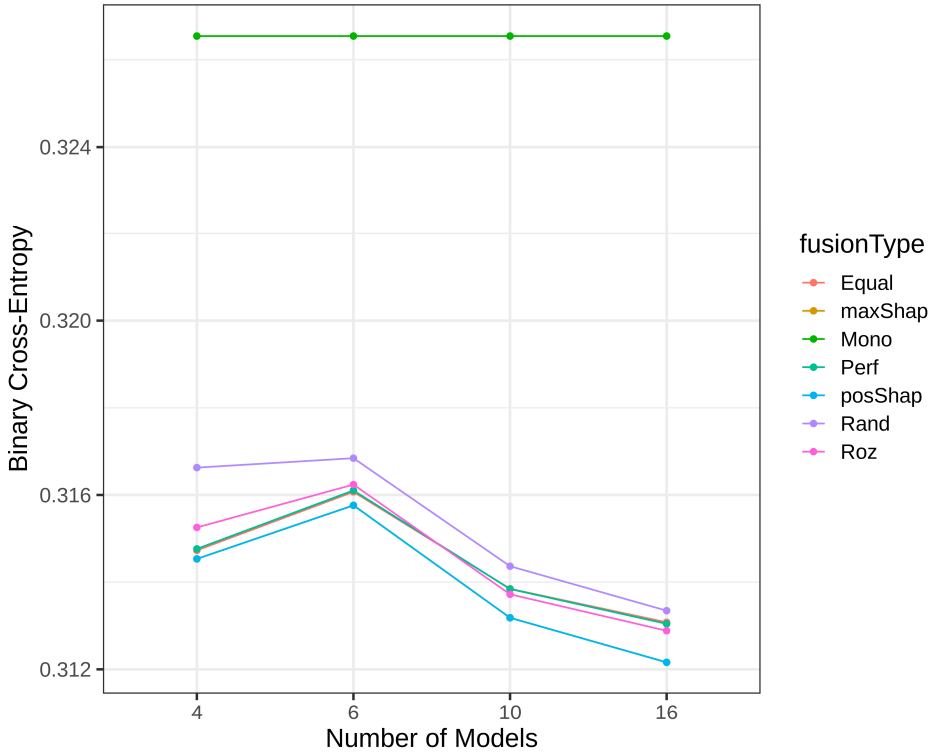

Figure 15. Detection error by *BCE* for various heterogenous ensemble sizes using Python models and BNG-credit_a dataset.

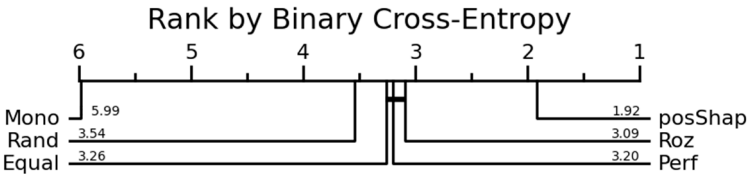

Figure 16. Ranking of heterogenous ensembles: using BNG-credit_a dataset and Python implementation.

The results of the heterogeneous ensembles for the BNG-creadit_a dataset implemented in the R language (Figure 19) reveal that all ensemble weighting strategies were outperformed by the Mono approach. Weighting strategies performed similarly with no statistically significant differences. Surprisingly, the best result of 0.339 *BCE* was achieved by Rand weighting. As for posShap, its performance dropped when the number of models in an ensemble was 16, but when applied to ensembles with 6 and 10 members, it exhibited the best performance. The ensemble ranking (Figure 20) also reveals that the results for all approaches have no statistically significant difference between them, the Mono approach ranked as the best, and other approaches performed without statistically significant difference between their results.

The results of the heterogeneous ensemble experiment are presented in Table 5, which demonstrated that in some cases choosing the right model type is more important than choosing the right weighting strategy. This is especially evident from Figure 19 where the monolithic logistic regression model outperformed the ensembles with performance differences ranging from 1.3% to 2.2%.

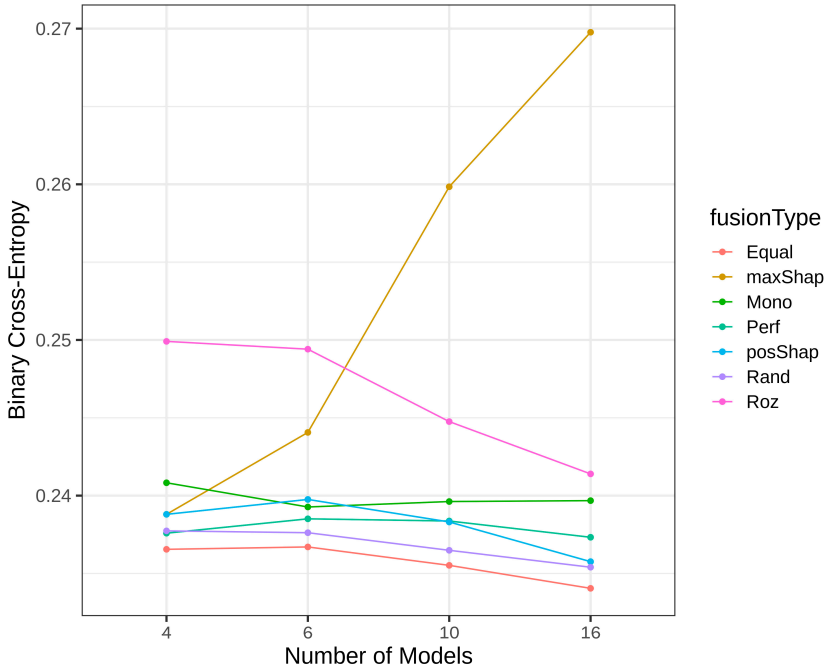

Figure 17. Detection error by *BCE* for various heterogenous ensemble sizes using R models and Bank Marketing dataset.

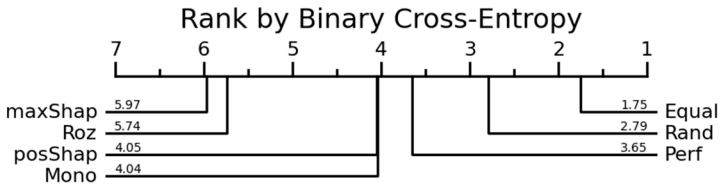

Figure 18. Ranking of heterogenous ensembles: using Bank Marketing dataset and R implementation.

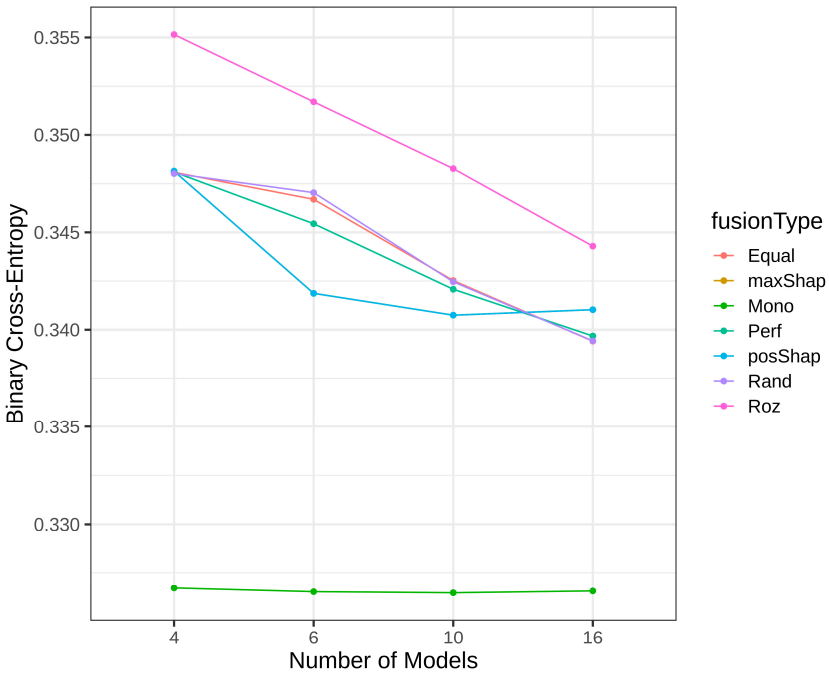

Figure 19. Detection error by *BCE* for various heterogenous ensemble sizes using R models and BNG-creadit_a dataset.

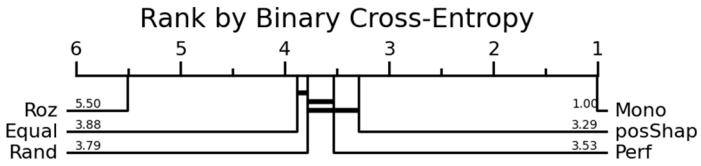

Figure 20. Ranking of heterogenous ensembles: using BNG-creadit_a dataset and R implementation.

Table 5. Rank comparison from Friedman test for heterogenous experiment results with lowest rank highlighted in bold.

Dataset	BNG [1]		Bank [2]	
Implementation / Weighting Strategy	Python	R	Python	R
Mono	5.99	**1**	4.89	4.04
Rand	3.54	3.79	2.96	2.79
Equal	3.26	3.88	**1.82**	**1.75**
Perf	3.20	3.53	3.91	3.65
Roz	3.09	5.50	3.81	5.74
MaxShap	-	-	6.23	5.97
PosShap	**1.92**	3.29	4.38	4.05

[1] BNG—BNG_credit-a dataset. [2] Bank—Bank Marketing dataset.

One could argue that the difference between the monolith model and the weighting strategies was caused by implementation-specific hyperparameter settings since no such advantage of the monolith can be noticed with the Python implementation (Figure 15) when using the BNG-Credit-A dataset.

The differences between implementations were also minimal when comparing Figures 13 and 17. The performances of the two implementations ranged from 0.233 to 0.269 *BCE* and exhibited the same best-performing ensembles achieved with an equal weighting strategy. The introduction of additional model types improved ensemble performance in three out of four test cases. Compared to the best result produced in homogeneous experiments, 0.236 *BCE*, the heterogeneous ensembles increased the performance to 0.233 *BCE*. Similarly, the experiment results reveal that in three out of four cases, any weighting strategy can outperform a monolithic model, signifying the value of ensembling. PosShap strategy outperformed the Roz strategy in all 3 tested heterogeneous experiment settings except for Python implementation using the Bank Marketing dataset, where results were not statistically different between the two strategies. The maxShap weighting strategy presented the worst results overall, indicating that underlying assumptions and efforts to account for it only deteriorated the ensemble. This was the more evident the larger the ensemble.

5. Discussion

Even though the maxShap weighting strategy performance was the lowest when compared to other tested approaches in both experiments, we consider that model variety would play a bigger role in real-life applications for heterogeneous ensembles. We assume that the cause of this phenomenon may be attributed to classifiers trained on smaller data sets that may contain more noisy data and may result in constructing a very weak learner upon it. Such models in real-life scenarios might be discarded as erroneous even before including them in an ensemble. The posShap method benefited from the opposite effect when the exclusion of poor-performing members boosted the performance of the ensemble. The experiment results revealed that even though in some tested cases the weighting strategy produces a positive gain in ensemble performance, it mostly depends on the used model type and datasets characteristics. When comparing performance between implementations in heterogeneous experiment results (Figures 15 and 19) a clear difference between the monolith model and weighting strategies, when the Mono approach outperforms weight-

ing strategies (Figure 19), surprisingly, only in the R implementation case. We speculate that this disparity could be attributed to implementation-specific calculation methods of Mono models and differences in hyperparameters, although we tried to unify those values as much as possible.

We assume that the presented weighting strategy could be used in applications that require high model precision such as medical research because in some other cases, the accuracy gains would not be significant enough to justify the computational costs of Shapley value exact estimation. The computational complexity of exact Shapley computations is O(N!), whereas the expected marginal contributions [19] (EMC) approximation methods complexity is O(N). For illustration, the runtime of Shapley calculations has been evaluated and the results, supporting these theoretical complexities, are presented in Figure 21. The exact Shapley computation runtime was similar for ensembles that contained 2–5 members, but its runtime increased exponentially with higher member counts. The average calculation time for exact Shapley calculations with 13 ensemble members was 13.612 s and for EMC approximation 0.002 s.

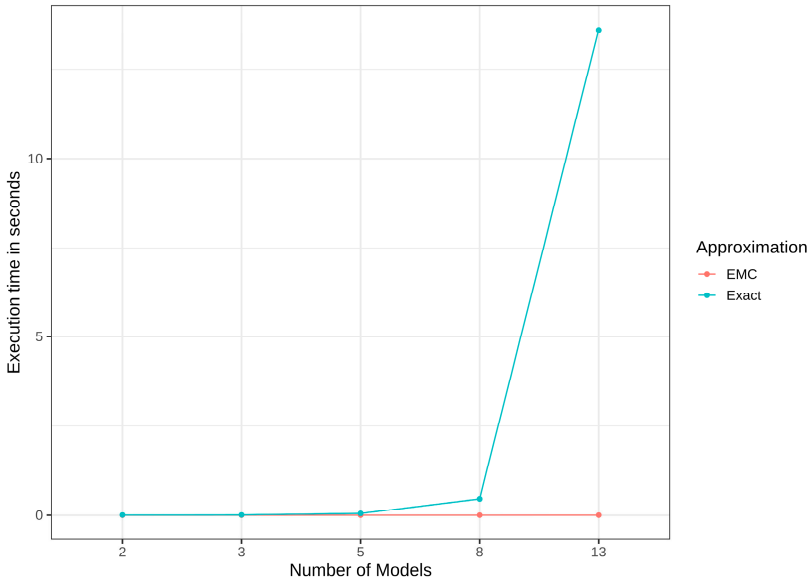

Figure 21. Runtime comparison of Shapley-based weighting strategies used in experiments. Note: exact was used in the maxShap or posShap strategies and EMC was used in Roz strategy [19].

This problem of increase in runtime could be addressed by estimating Shapley value using existing approximation algorithms [62,63], as in, related Shapley vote-based strategy [19] or using approaches to reduce the number of models inside the ensemble without significantly worsening its performance [64]. However, if approximation is applied our method would have a computational advantage over competitors' strategy [19] because their calculations were performed at an individual data point level, whereas we pool all the data into a validation set and estimate prediction performance on it. The proposed strategy could also be applied to achieve privacy preservation when applied in conjunction with knowledge transfer architecture [17]. The strategy could provide optimal weights for an ensemble of individually trained shared models that would result in a strong teacher model.

Another use for this study could be developing the architecture described in [20,21] to provide motivation and an incentive mechanism for the blockchain network participants. Incentive mechanisms, based on Shapley value, could be used to measure a participant's contribution, and provide rewards for model and data sharing. The incentive mechanism would evaluate all machine learning models uploaded by calculating the *BCE* metric using all existing data donated to the network. This model performance results presented in *BCE* should then be used as an input into the Shapley function that evaluates all possible combinations.

6. Conclusions

The proposed weighting strategy for ensemble learning used the reciprocal of binary cross-entropy as a model performance metric and the Shapley value estimation to enhance performance weighting with a more global ensembling aspect: how useful the model in question is, if we test all possible ensemble combinations both including and excluding it, under the assumption of equal weights. Two variations of this strategy (posShap and maxShap) were implemented and empirically compared with the monolith model and other weighting strategies using two large banking-related datasets. A decision tree and logistic regression were used as our base learners for constructing homogenous and heterogeneous ensembles.

The highest performance gains over the monolith model in homogeneous ensembles were observed for the posShap strategy, with the largest ensemble size of 13 members: 4.8% and 1.9% for the BNG_credit-a and Bank Marketing datasets, respectively. If compared to traditional performance-based weighting (Perf), our approach improved the performance by 0.7%. The posShap was ranked as the best strategy, except for the case of the Python decision tree classifier using the Bank Marketing dataset. Similarly, the posShap weighting strategy in heterogeneous ensembles achieved the best performance with the largest ensemble size of 16 members, achieving a performance gain of 1.4% over the monolith approach using the BNG-credit-a dataset. Using the Bank Marketing dataset, posShap featured a 0.4% gain, but the winner there was equal weighting with a 0.6% gain over the monolith model.

From the two variants tested only posShap was successful, whereas maxShap was outperformed by all other weighting strategies, implicating that efforts to correct model outputs could not improve the performance of a resulting ensemble and simply eliminating such models from the ensemble works better. While the performance of posShap differs with respect to the dataset and the base learner used, the experiments demonstrate that it performs better or at least similarly when compared to other weighting strategies, including the most similar Shapley vote-based strategy (Roz).

Author Contributions: Conceptualization, V.D. and E.V.; methodology, V.D. and E.V.; software, V.D.; validation, L.A.; formal analysis, V.D., E.V., and L.A; investigation, V.D.; resources, L.Č. and E.V.; data curation, E.V. and V.D.; writing—original draft preparation, V.D. and L.A.; writing—review and editing E.V. and L.Č.; visualization, V.D. and E.V.; supervision, E.V. and L.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found online: Bank Marketing Dataset—(https://www.openml.org/search?type=data&status=active&id=1461 accessed on 15 March 2023) and BNG_credit-a dataset—(https://www.openml.org/search?type=data&status=active&id=258 accessed on 15 March 2023).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vopson, M.M. The information catastrophe. *AIP Adv.* **2020**, *10*, 085014. [CrossRef]
2. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]
3. Schapire, R.E. The strength of weak learnability. *Mach. Learn.* **1990**, *5*, 197–227. [CrossRef]
4. González, S.; García, S.; Del Ser, J.; Rokach, L.; Herrera, F. A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities. *Inf. Fusion* **2020**, *64*, 205–237. [CrossRef]
5. Fan, W.; Stolfo, S.J.; Zhang, J. The application of AdaBoost for distributed, scalable and on-line learning. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 15–18 August 1999.
6. Fern, A.; Givan, R. Online ensemble learning: An empirical study. *Mach. Learn.* **2003**, *53*, 71–109. [CrossRef]

7. Street, W.N.; Kim, Y. A streaming ensemble algorithm (SEA) for large-scale classification. In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 26 August 2001.
8. Wang, H.; Fan, W.; Yu, P.S.; Han, J. Mining concept-drifting data streams using ensemble classifiers. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24 August 2003.
9. Kolter, J.Z.; Maloof, M.A. Dynamic weighted majority: An ensemble method for drifting concepts. *J. Mach. Learn. Res.* **2007**, *8*, 2755–2790.
10. Bowser, A.; Wiggins, A.; Shanley, L.; Preece, J.; Henderson, S. Sharing Data While Protecting Privacy in Citizen Science. *Interactions* **2014**, *21*, 70–73. [CrossRef]
11. Verbraeken, J.; Wolting, M.; Katzy, J.; Kloppenburg, J.; Verbelen, T.; Rellermeyer, J.S. A survey on distributed machine learning. *ACM Comput. Surv. Csur* **2020**, *53*, 1–33. [CrossRef]
12. Wei, J.; Dai, W.; Qiao, A.; Ho, Q.; Cui, H.; Ganger, G.R.; Gibbons, P.B.; Gibson, G.A.; Xing, E.P. Managed communication and consistency for fast data-parallel iterative analytics. In Proceedings of the Sixth ACM Symposium on Cloud Computing, Kohala Coast, HI, USA, 27–29 August 2015.
13. Ma, C.; Li, J.; Shi, L.; Ding, M.; Wang, T.; Han, Z.; Poor, H.V. When federated learning meets blockchain: A new distributed learning paradigm. *IEEE Comput. Intell. Mag.* **2022**, *17*, 26–33. [CrossRef]
14. Tuladhar, A.; Gill, S.; Ismail, Z.; Forkert, N.D. Alzheimer's Disease Neuroimaging Initiative, Building machine learning models without sharing patient data: A simulation-based analysis of distributed learning by ensembling. *J. Biomed. Inform.* **2020**, *106*, 103424. [CrossRef] [PubMed]
15. Lu, Y.; Huang, X.; Dai, Y.; Maharjan, S.; Zhang, Y. Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4177–4186. [CrossRef]
16. Chen, X.; Ji, J.; Luo, C.; Liao, W.; Li, P. When machine learning meets blockchain: A decentralized, privacy-preserving and secure design. In Proceedings of the IEEE International Conference on Big Data (Big Data), Seattle, WA, USA, 10 December 2018.
17. Xu, R.; Baracaldo, N.; Joshi, J. Privacy-preserving machine learning: Methods, challenges and directions. *arXiv* **2021**, arXiv:2108.04417.
18. Shapley, L.S. A value for n-person games. In *Contributions to the Theory of Games*; Princeton University Press: Princeton, NJ, USA, 1953; Volume 2, pp. 307–318.
19. Rozemberczki, B.; Sarkar, R. The shapley value of classifiers in ensemble games. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. Virtual Event, Queensland, Australia, 1–5 November 2021.
20. Drungilas, V.; Vaičiukynas, E.; Ablonskis, L.; Čeponienė, L. Heterogeneous Models Inference Using Hyperledger Fabric Oracles. In Proceedings of the First Blockchain and Cryptocurrency Conference B2C' 2022, Barcelona, Spain, 9–11 November 2022.
21. Drungilas, V.; Vaičiukynas, E.; Jurgelaitis, M.; Butkienė, R.; Čeponienė, L. Towards blockchain-based federated machine learning: Smart contract for model inference. *Appl. Sci.* **2021**, *11*, 1010. [CrossRef]
22. Rokach, L. *Ensemble Learning: Pattern Classification Using Ensemble Methods*; World Scientific: Singapore, 2019.
23. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1249. [CrossRef]
24. Dogan, A.; Birant, D. A weighted majority voting ensemble approach for classification. In Proceedings of the 2019 4th International Conference on Computer Science and Engineering UBMK, Samsun, Turkey, 11–15 September 2019; pp. 1–6.
25. Prodromidis, A.L.; Stolfo, S.J.; Chan, P.K. Effective and efficient pruning of meta-classifiers in a distributed data mining system. *Knowl. Discov. Data Min. J.* **1999**, *32*, 1–29.
26. Shahhosseini, M.; Hu, G.; Pham, H. Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. *Mach. Learn. Appl.* **2022**, *7*, 100251. [CrossRef]
27. Štrumbelj, E.; Kononenko, I. Explaining prediction models and individual predictions with feature contributions. *Knowl. Inf. Syst.* **2014**, *41*, 647–665. [CrossRef]
28. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 1–10.
29. Rozemberczki, B.; Watson, L.; Bayer, P.; Yang, H.T.; Kiss, O.; Nilsson, S.; Sarkar, R. The shapley value in machine learning. *arXiv* **2022**, arXiv:2202.05594.
30. Tang, S.; Ghorbani, A.; Yamashita, R.; Rehman, S.; Dunnmon, J.A.; Zou, J.; Rubin, D.L. Data valuation for medical imaging using Shapley value and application to a large-scale chest X-ray dataset. *Sci. Rep.* **2021**, *11*, 1–9. [CrossRef]
31. Wang, T.; Rausch, J.; Zhang, C.; Jia, R.; Song, D. A principled approach to data valuation for federated learning. *Fed. Learn. Priv. Incent.* **2020**, *12500*, 153–167.
32. Ykhlef, H.; Bouchaffra, D. Induced subgraph game for ensemble selection. *Int. J. Artif. Intell. Tools* **2017**, *26*, 1760003. [CrossRef]
33. Chen, X.; Li, S.; Xu, X.; Meng, F.; Cao, W. A novel GSCI-based ensemble approach for credit scoring. *IEEE Access* **2020**, *8*, 222449–222465. [CrossRef]
34. Dumitrescu, E.; Hué, S.; Hurlin, C.; Tokpavi, S. Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects. *Eur. J. Oper. Res.* **2022**, *297*, 1178–1192. [CrossRef]
35. Laaksonen, J.; Oja, E. Classification with learning k-nearest neighbors. In Proceedings of the International Conference on Neural Networks (ICNN'96), Washington, DC, USA, 3–6 June 1996.
36. Karthik, S.; Bhadoria, R.S.; Lee, J.G.; Sivaraman, A.K.; Samanta, S.; Balasundaram, A.; Chaurasia, B.K.; Ashokkumar, S. Prognostic kalman filter based bayesian learning model for data accuracy prediction. *Comput. Mater. Contin* **2022**, *72*, 243–259.
37. Loh, W.Y. Classification and regression trees. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2011**, *1*, 14–23. [CrossRef]

38. King, J.E. *Binary Logistic Regression. Best Practices in Quantitative Methods*; Osborne, Ed.; Jason SAGE Publications, Inc.: Los Angeles, CA, USA, 2008.
39. Wang, Q.; Ma, Y.; Zhao, K.; Tian, Y. A comprehensive survey of loss functions in machine learning. *Ann. Data Sci.* **2020**, 1–26. [CrossRef]
40. Roth, A.E. *The Shapley Value: Essays in Honor of Lloyd S. Shapley*; Cambridge University Press: Cambridge, UK, 1988.
41. Zhang, Z.; Ho, K.M.; Hong, Y. Machine learning for the prediction of volume responsiveness in patients with oliguric acute kidney injury in critical care. *Crit. Care* **2019**, *23*, 112. [CrossRef] [PubMed]
42. Agarap, A.F. Deep Learning using Rectified Linear Units (ReLU). *arXiv* **2018**, arXiv:1803.08375.
43. Sergio, M.; Laureano, R.; Cortez, P. Using Data Mining for Bank Direct Marketing: An Application of the CRISP-DM Methodology. In Proceedings of the European Simulation and Modelling Conference-ESM'2011, Guimarães, Portugal, 24–26 October 2011.
44. Quinlan, J.R. Simplifying decision trees. *Int. J. Man-Mach. Stud.* **1987**, *27*, 221–234. [CrossRef]
45. van Rijn, J.N.; Holmes, G.; Pfahringer, B.; Vanschoren, J. Algorithm selection on data streams. In Proceedings of the Discovery Science: 17th International Conference, Bled, Slovenia, 8–10 October 2014.
46. Hsieh, K.; Phanishayee, A.; Mutlu, O.; Gibbons, P. The non-iid data quagmire of decentralized machine learning. In Proceedings of the International Conference on Machine Learning, Virtual Event, 12–18 July 2020.
47. Stripelis, D.; Thompson, P.M.; Ambite, J.L. Semi-synchronous federated learning for energy-efficient training and accelerated convergence in cross-silo settings. *ACM Trans. Intell. Syst. Technol. (TIST)* **2022**, *13*, 78. [CrossRef]
48. Michieli, U.; Ozay, M. Prototype guided federated learning of visual feature representations. *arXiv* **2021**, arXiv:2105.08982.
49. Arnold, S.; Yesilbas, D. Demystifying the effects of non-independence in federated learning. *arXiv* **2021**, arXiv:2103.11226.
50. Wadu, M.M.; Samarakoon, S.; Bennis, M. Joint client scheduling and resource allocation under channel uncertainty in federated learning. *IEEE Trans. Commun.* **2021**, *69*, 5962–5974. [CrossRef]
51. Wang, X.; Han, Y.; Wang, C.; Zhao, Q.; Chen, X.; Chen, M. In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning. *IEEE Netw.* **2019**, *33*, 156–165. [CrossRef]
52. Zhou, X.; Deng, Y.; Xia, H.; Wu, S.A.; Bennis, M. Time-triggered Federated Learning over Wireless Networks. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 11066–11079. [CrossRef]
53. Critical Difference Diagram with Wilcoxon-Holm Post-Hoc Analysis. Available online: https://github.com/hfawaz/cd-diagram (accessed on 15 March 2023).
54. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
55. Friedman, M. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* **1940**, *11*, 86–92. [CrossRef]
56. Benavoli, A.; Corani, G.; Mangili, F. Should we really use post-hoc tests based on mean-ranks? *J. Mach. Learn. Res.* **2016**, *17*, 152–161.
57. Wilcoxon, F. Individual comparisons of grouped data by ranking methods. *J. Econ. Entomol.* **1946**, *39*, 269–270. [CrossRef]
58. Holm, S. A simple sequentially rejective multiple test procedure. *Scand. J. Stat.* **1979**, *6*, 65–70.
59. The Official Implementation of "The Shapley Value of Classifiers in Ensemble Games" (CIKM 2021). Available online: https://github.com/benedekrozemberczki/shapley (accessed on 10 May 2023).
60. Experiment Results for All Tested Ensemble Sizes and Datasets. Available online: https://github.com/HurrisLT/ShapleyWeighting (accessed on 10 May 2023).
61. Heaton, J. An empirical analysis of feature engineering for predictive modeling. In Proceedings of the SoutheastCon, Norfolk, VA, USA, 30 March–3 April 2016.
62. Castro, J.; Gómez, D.; Tejada, J. Polynomial calculation of the Shapley value based on sampling. *Comput. Oper. Res.* **2009**, *36*, 1726–1730. [CrossRef]
63. Maleki, S.; Tran-Thanh, L.; Hines, G.; Rahwan, T.; Rogers, A. Bounding the estimation error of sampling-based Shapley value approximation. *arXiv* **2013**, arXiv:1306.4265.
64. Uzunoglu, B.; Fletcher, S.J.; Zupanski, M.; Navon, I.M. Adaptive ensemble reduction and inflation. *Q. J. R. Meteorol. Soc. A J. Atmos. Sci. Appl. Meteorol. Phys. Oceanogr.* **2007**, *133*, 1281–1294. [CrossRef]