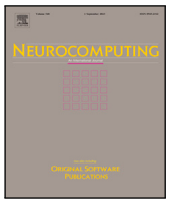
Neurocomputing 620 (2025) 129191

Contents lists available at ScienceDirect

Neurocomputing

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



Markowitz random forest: Weighting classification and regression trees with modern portfolio theory

Eleftherios Kouloumpris *, Ioannis Vlahavas

School of Informatics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece

ARTICLE INFO

Communicated by P.N. Suganthan

Keywords:
Random forest
Tree weighting
Ensemble methods
Modern portfolio theory
Mean-variance analysis

ABSTRACT

Tree-based ensembles such as random forest (RF) are essential methods for supervised learning. Whereas traditional RFs assign equal weights to their trees, significant evidence suggests that tree weighting schemes can enhance predictive performance. Previous works have focused solely on the predictive performance of individual trees, assigning greater weights to high-performing trees. However, the predictive power of RF arises not only from high performing trees but also from tree variety, a factor that has not been considered before. In this paper we propose Markowitz RF, a tree weighting method that considers both tree performance and variety, using a tree covariance matrix for risk regularization. Our method is formulated as an adapted optimization process inspired by financial mathematics. It can be applied to binary and multi-class classification, as well as regression tasks. Our experiments on 15 benchmark datasets indicate that MRF can significantly outperform previously proposed tree weighting approaches and other learning methods in terms of Precision-Recall AUC, mean absolute error and F_1_*macro*.

1. Introduction

The recent advances in machine learning (ML) constitute a major accomplishment of artificial intelligence (AI) research. The further development of supervised learning, a well-studied ML paradigm, has important implications for both academia and industry [1]. Tree-based ensembles (TBE) are currently among the most popular supervised learning methods [2].

Whereas the deep learning approach provides state-of-the-art results for multimedia data (e.g., image, video, sound), according to [3], TBEs are still among the best performing methods for tabular datasets. For instance, most Kaggle competitions have been won by a combination of feature engineering and a TBE solution. Two well known TBEs are random forest (RF) [4] and gradient boosting (GB) [5]. RF combines multiple low-bias/high-variance trees in parallel in order to decrease the total variance, while GB sequentially combines high-bias/low-variance trees to decrease the total bias. Further on, we will consider how these trees should be weighted for optimal results. Given that GB inherently decides the appropriate tree weights, we will only consider RF for the remainder of this work.

The traditional RF algorithm gives equal weights to every tree, a strategy that we will denote as equally weighted trees (EWT). Previous works claim that EWT is an arbitrary strategy that does not consider factors such as tree performance (e.g., [6,7]). The same works have proposed different weight allocation strategies that aim to improve upon EWT. Whereas the proposed weight allocation strategies take into account tree performance, to the best of our knowledge, information relating to the covariance between trees has not been considered before. Therefore, weighting methods that rely only on tree performance run the risk of giving significant weights to a small set of high performing, but also significantly correlated trees that make similar mistakes. This is an important issue because the predictive power of RF relies not only on the predictive performance of independent trees, but also on having a diverse set of uncorrelated trees (tree diversity).

In this paper, taking into account the influence of weights on tree diversity, we propose a novel method for estimating the optimal weights of RF trees. Our method is inspired by financial mathematics, in the same vein that neural networks and genetic algorithms are inspired by neuroscience and biology. We also note that while many ML methods are commonly used to solve problems in finance (i.e., ML-based finance), in the opposite direction, we propose the first finance-inspired ML method. More generally, our intention is to demonstrate that financial mathematics can lead to innovative AI research.

Although financial methods provide an initial inspiration, their direct application to supervised learning is not fully adequate. On the contrary, we propose several adaptations that are necessary to develop

* Corresponding author.
E-mail addresses: elefthenk@csd.auth.gr (E. Kouloumpris), vlahavas@csd.auth.gr (I. Vlahavas).

https://doi.org/10.1016/j.neucom.2024.129191
Received 2 February 2024; Received in revised form 10 November 2024; Accepted 14 December 2024
Available online 24 December 2024

methods that are suitable for the supervised learning setting. Furthermore, our ML-oriented treatment allows us to transform financial formulas into familiar ML concepts.

Specifically, we adapt a method from modern portfolio theory called mean–variance analysis (MVA), first introduced by H. Markowitz in 1952 [8], in order to optimize RF tree weights in both regression and classification settings. Hence, we name our method Markowitz random forest (MRF). MVA was originally designed for the construction of portfolios of financial assets (e.g., stocks) that have high expected returns, but that should also include a diverse set of assets. Therefore, the main contribution of the paper is that it proposes the first tree weighting method that takes into account both tree performance and tree variety, through a constrained joint optimization process. In alignment with standard ML terminology, the process is formulated as a tree performance optimization task, augmented with a regularization penalty that discourages the allocation of weight to trees that make too similar mistakes.

Our experimental work on five binary classification, five regression and five multi-class classification datasets show that MRF can provide improved predictive performance compared to the traditional RF and previously proposed tree weighting methods. Through statistical tests, several improvements are proven statistically significant across several datasets and learning tasks. These results provide substantial evidence that joint optimization of tree performance and variety can boost the accuracy of RF models. The proposed method is also favorably compared against several well-known tree-based ensembles, including XGBoost, Extremely Randomized Trees (ERT), Oblique Forest (OblF) and Rotation Forest (RotF) [9–15]. This comparison demonstrates that standard RF, with enhanced tree weighting alone, can rival advanced multivariate split methods like OblF and RotF.

The remainder of the paper has the following structure. Section 2 presents and discusses previously proposed tree weighting methods for RF. Section 3 provides the necessary background for RF and MVA. Section 4 presents the proposed method and the experimental setup. Section 5 reveals the experimental results and provides additional discussion. Section 6 presents a computational complexity analysis of MRF. Section 7 summarizes the advantages and disadvantage of the proposed method. Finally, Section 8 concludes the paper.

2. Related work

Supervised learning is a well researched ML paradigm and is commonly used in modern AI applications. One possible taxonomy of its methods includes generalized linear models [16], tree-based methods [2], probabilistic methods [17], kernel methods [18], instance-based methods [19] and deep learning [20]. Currently, deep learning and tree-based methods are among the leading fields of supervised learning research.

Deep learning offers superior capabilities for automated representation learning on large multimedia and corpus datasets. This is achieved through application-oriented artificial neural network (ANN) architectures, such as convolutional neural networks for computer vision tasks [20] and the transformer architecture for natural language processing [21].

In contrast, tree-based methods are frequently the best performing in problems with smaller tabular datasets that have heterogeneous features [3]. These data have a mixed representation of categorical and numerical features, and may also include missing or incorrect values. For the above reasons, the representation learning capabilities of deep learning are limited in this class of problems, while the datasets are often too small for training ANNs.

Such problems often arise in healthcare, business and engineering domains, in which data collection can be a time-consuming, costly and error-prone process. Tree-based methods are also relevant when model interpetability is essential. TBEs include randomized trees such as RF and extremely randomized trees (ERT) [9], and gradient boosting method such as XGBoost [10]. In this paper, we consider the potential of tree-weighted RF models, so the following paragraphs will present the corresponding literature.

Li et al. [22] proposed tree weighted random forest (TWRF) in order to weight trees according to their out-of-bag (OOB) performance in terms of classification accuracy. TWRF reduced the effect of noisy trees and was able to outperform RF and other traditional supervised learning methods. Winham et al. [23] introduced another tree weighted RF in which weights reflected tree accuracy. Besides improving the predictive performance of the ensemble, the latter work also emphasized the computation of feature importance. El Habib Daho et al. [24] introduced a variant of RF that, among other modifications, also used tree weighting. Similarly to the previously mentioned works, tree weights reflected OOB performance. The final ensemble was evaluated on several medical datasets.

Pham and Olafsson [25] considered a modified voting scheme that used Cesáro sequence averaging instead of the traditional average voting. Essentially, this voting scheme relies on a sorted sequence of the trees, with trees at the beginning of the sequence receiving larger weights. They used two criteria to sort the trees: (a) OOB error rates and (b) accuracy on another training set. They provided both theoretical and empirical proof that Cesáro RF can outperform RF under certain conditions. Devi et al. [6] proposed a tree weighting method in the context of imbalanced financial data. Specifically, they assigned the tree weights according to tree performance on the minority class. The proposed system outperforms traditional RF in an unbalanced fraud detection task.

Whereas the aforementioned works propose static weights that are estimated during training time, Jain et al. [26] proposed the exponentially weighted random forest (EWRF) scheme for dynamic weight allocation at prediction time. For this purpose, they defined an observation-tree similarity function based on the exponential function. When predicting a new example, the trees that are most similar to this test example receive the largest weights. The merit of dynamic weight allocation at prediction time was proven by multiple experiments. Gajowniczek et al. [27] experimented with weighting schemes for both observations and trees, with the latter relying on a combination of in-bag (INB) and OOB errors. Their solution was able to outperform traditional ensemble algorithms and decrease false alarms on data from Physionet/Computing in Cardiology Challenge, 2015.

Shahhosseini and Hu [7] proposed tree weighting methods based on constrained optimization of accuracy and area under curve (AUC), including also several stacking-based solutions. The stacked solution trained a second RF based on the first RF's OOB predictions or probabilities. More recently, Zhang et al. [28] worked with probability intervals to introduce the cautious weighted random forest (CWRF). Based on the theory of belief functions, they designed a convex optimization problem that takes into account both determinacy (i.e., the preciseness of a probability interval) and accuracy. Experiments on multiple benchmark datasets showed the appropriateness of the method for problems in which cautiousness is important or when data are of low quantity or quality.

In summary, our study of the literature reveals significant evidence that weighted tree RF methods can outperform the traditional algorithm. However, the majority of the reviewed methods rely only on independent tree performance and have a strong focus on binary classification. The most recent works tend to formulate tree weight allocation as a constrained optimization problem [7,28], which we also do in this work. However, our formulation is different as it has a double objective; instead of focusing only on independent tree performance, it also encourages tree diversity. Furthermore, we consider the performance of our method across different learning tasks, including multi-class classification.

3. Background

3.1. Random forest

RF is an integration of two ideas, namely the random subspaces method (RSM) and bootstrap aggregating (bagging). Ho [29] proposed RSM, a method that trains independent estimators on different feature subspaces (i.e., columns of a design matrix) that are sampled with replacement. Breiman [30] introduced bagging, which trains estimators on multiple datasets by sampling examples (i.e., rows of a design matrix) with replacement. He also coined the term RF for a method that combines RSM/bagging and is specifically applied to decision trees [4].

RF starts by creating a number of samples via bagging, and then fits a tree on each sample. The algorithm used to fit each tree differs from the traditional decision tree algorithm. Specifically, for each internal node, only a random subspace of features is considered for the best split. This randomization contributes to the decrease of the total variance of the ensemble, which results in an increased performance.

3.2. Mean–variance analysis

In portfolio management, an investor has to decide how to allocate a certain capital C to a number k of financial assets (e.g., stocks) with returns s. Assuming that for each asset there is an expected future return $r_i = \mathbb{E}\left[s_i\right], i \in [1,k]$, the allocation of capital is denoted by the portfolio weights $w_i, i \in [1,k]$ such that $w_i > 0$ and $\sum_{i=1}^{k} w_i = 1$. The vectors of expected returns and weights for all assets are denoted as $r \in \mathbb{R}^k$ and $w \in \mathbb{R}^k$.

MVA determines the optimal allocation of investment funds across a set of financial assets. This method requires two inputs: a vector of expected returns r, as previously described, and a risk matrix denoted K. The standard risk matrix is estimated as the symmetric covariance matrix of returns, $K = cov(r) \in \mathbb{R}^{kxk}$. The entries $K[i,j] = K[j,i] = \sigma_{ij}$ represent the risk associated with the covariance between stock returns s_i and s_j, as given in Eq. (1).

$$\sigma_{ij} = \mathbb{E}\left[(s_i - \mathbb{E}[s_i])(s_j - \mathbb{E}[s_j])\right] \tag{1}$$

A highly positive $K[j,i]$ implies that if one stock underperforms, the other is likely to underperform as well. When portfolio weights are overly concentrated in stocks with highly positive covariance, this indicates an increased risk of many stocks underperforming simultaneously.

According to modern portfolio theory, an investor typically aims for a portfolio with high expected returns and low risk. Given a set of expected returns r, the risk matrix K and the weights w, it is possible to estimate the expected performance and total risk of the entire portfolio. The expected performance M and the risk V of a portfolio with weights w are provided in Eqs. (2)–(3).

$$M = \sum_{i=1}^{k} w_i r_i = w^T r \tag{2}$$

$$V = \sum_{i=1}^{k} \sum_{j=1}^{k} w_i w_j \sigma_{ij} = w^T K w \tag{3}$$

The goal of MVA is to find portfolio weights w that maximize the expected portfolio performance (M) while keeping the total risk (V) low. These optimal portfolio weights can be estimated by optimizing Eq. (4) for different values of the parameter λ, which controls the return/risk trade-off.

$$\begin{aligned} \max_{w} \quad & w^T r - \lambda w^T K w \\ \text{s.t.} \quad & \sum_{i=0}^{k} w_i = 1 \\ & 0 \le w_i \le 1 \end{aligned} \tag{4}$$

The above is a convex optimization problem that can be solved with Quadratic Programming (QP) [31]. For a detailed explanation of optimization methods in finance, consider the work of Cornuéjols and Tutuncu [32].

In the upcoming Sections 4.3–4.5, a key step in adapting MVA for the supervised learning setting is to formulate r and K in a way that accurately reflects expected tree performance and the risk associated with tree similarity. These formulations vary depending on the specific task—whether binary classification, multiclass classification, or regression. MVA then uses these adapted inputs to optimize tree weights according to a two-fold objective: first, to assign greater weights to high-performing trees, and second, to avoid concentrating weights on trees that make similar mistakes, as this would diminish the ensemble's effectiveness.

4. Methods and data

4.1. Datasets and descriptive statistics

We performed our experiments on 15 datasets from the University of California Irvine (UCI) ML Repository [47] and Kaggle Datasets, provided in Table 1. The datasets are related to three different learning tasks. Among these, the five datasets that involve a binary classification task are Cylinder Bands Data Set (bands) [33], Online Shoppers Intentions (shoppers) [34], Default of Credit Card Clients (credit) [35], Bank Marketing (bank-marketing) [36] and AIDS Clinical Trials Group Study 175 (aids) [37]. The five datasets with a regression task are Online News Popularity (news) [38], Superconductivity [39], Insurance Company Benchmark (insurance) [40], Wine Quality (wine) [41] and Cars Price (cars) from Kaggle Datasets. The remaining five datasets related to multi-class classification problem are: Steel Plates Faults (plates) [42], Statlog Image Segmentation (statlog) [43], Student's Performance in Higher Education (student) [44], Room Occupancy Estimation (room) [45] and Pen-Based Recognition of Handwritten Digits (pen) [46].

Our pre-processing pipeline performs a removal of duplicate records, one-hot encoding of categorical variables and the filling of missing values with the mean value of each column. Statistics for the processed datasets appear in the Table 2.

4.2. Technologies and programming packages

All experiments have been conducted with the Python programming language, using the RF and ERT implementations of Scikit-learn and the XGBoost package. For MVA we used the implementation of the python package PyPortfolioOpt. Furthermore, PyPortfolioOpt relies on the cvxopt and cvxpy packages for the solution of convex optimization problems.

4.3. Markowitz random forest for binary classification

Our main goal is to apply a constrained joint optimization framework, originally intended to solve portfolio management problems, as a solution for the RF tree weighting problem. Specifically, we consider the MVA framework, which has been explained in Section 3.2. As mentioned in Section 1, we name this tree-weighted RF method as MRF. In order to apply this solution of financial mathematics to our supervised learning problem, which is a case of reducing one problem to another, we consider the mapping of concepts presented in Table 3.

We elaborate on why MVA is an attractive framework for tree weighting. Traditionally, investors are interested in portfolios with high return potentials. However, given two portfolios with equal return potentials, the investor will choose the most diversified portfolio (i.e., the one that has less correlated assets). The other, less diversified portfolio comes with additional and unnecessary risk (volatility) for the same expected return. MVA is designed to optimize towards portfolios with

Table 1
Datasets used for three learning tasks.

Problem	Data id	Dataset/Link	Related work
Binary clf.	bands	Bands Data Set	Evans and Fisher [33]
	shoppers	Online Shoppers Intentions	Sakar et al. [34]
	defaults	Default of Credit Card Clients	Yeh and Lien [35]
	bank-marketing	Bank Marketing	Moro et al. [36]
	aids	AIDS Clinical Trials Group Study 175	Hammer et al. [37]
Regression	news	Online News Popularity	[38]
	superconductivity	Superconductivity	Hamidieh [39]
	insurance	Insurance Company Benchmark	Putten et al. [40]
	cars	Cars Price Dataset	(Kaggle Dataset)
Multi-class clf.	wine	Wine Quality	Cortez et al. [41]
	plates	Steel Plates Faults	Buscema et al. [42]
	statlog	Statlog Image Segmentation	UCI Machine Learning Repository [43]
	students	Student's performance in higher education	[44]
	room	Room Occupancy Estimation	Singh et al. [45]
	pen	Pen-Based Recognition of Handwritten Digits	Reisizadeh et al. [46]

Table 2
Dataset statistics.

Dataset	Rows	Columns
bands	541	96
aids	2139	23
shoppers	12 330	28
credit	30 000	24
bank-marketing	41 176	63
insurance	5822	85
wine	6497	12
cars	19 237	783
superconduct	21 263	81
news	39 644	58
plates	1941	27
statlog	2310	19
students	4424	36
room	10 129	16
pen	10 992	16

Table 3
Mapping of concepts.

Portfolio management	Tree weighting
Financial assets	Decision trees
Assets weights	Tree weights
Portfolio	Ensemble (forest)
Portfolio total return	Ensemble accuracy
Portfolio total risk	Ensemble variance
Diversification	Uncorrelated mistakes

high returns, that are also well-diversified. Similarly, RF works best when it consists of accurate trees, which are also less correlated to each other. Consequently, our application of MVA to tree weighting can simultaneously consider both (a) tree performance and (b) promote the diversification of trees by assigning more weight to uncorrelated trees. Instead of a direct application of this financial method to tree weighting, it is better to make several adaptations that make it more suitable in the context of ML.

One first adaptation is that we consider only the correlation of trees on prediction mistakes, that are either False Positives (FPs) or False Negatives (FNs). The rationale is that the method should discourage the allocation of weights to trees that make similar mistakes. The second adaptation is that covariance minimization is formulated as a regularization penalty. This formulation is more common in the ML literature (e.g., Ridge or Lasso penalized regression).

As mentioned in Section 3.2, MVA requires an expected performance vector r and risk matrix K. We will explain how these are designed for the tree weighting problem. Initially, an RF model with k trees is fitted for a binary classification task on a training dataset X with m examples. Then, each tree RF_j predicts probabilities for each example X_i. The positive class probability prediction of the jth tree on the ith example is denoted as P_{ij}.

As a performance vector r for MVA, we consider the Precision-Recall Area Under Curve (PRAUC) of each tree. This has the benefit of being independent of the decision threshold and suitable for imbalanced data in binary classification. The assumption is that a weighted RF can achieve higher PRAUC by giving more weight to trees that independently achieve high PRAUC scores. The performance vector used in this learning task is shown in Eq. (5), in which y is the vector of actual labels.

$$r = [PRAUC(P_{:1}, y), \dots, PRAUC(P_{:n}, y)] \in [0, 1] \tag{5}$$

The risk estimation process involves a few more steps. First, for each combination of example and tree we use the actual labels $y_i \in \{0, 1\}$ to compute the probability given to the correct class (denoted as *score*), as in Eq. (6).

$$score(y_i, P_{ij}) = P_{ij}^{y_i}(1 - P_{ij})^{(1-y_i)} \in [0, 1] \tag{6}$$

The scores for each example/tree combination are collected in a performance matrix Q, as depicted in Eq. (7):

$$Q = \begin{bmatrix} score(1,1) & \dots & score(1,k) \\ \vdots & score(i,j) & \vdots \\ score(m,1) & \dots & score(m,k) \end{bmatrix} \in \mathbb{R}^{m \times k} \tag{7}$$

To compute the risk estimation for a pair of trees (a, b), we find the set of all examples which are FP for at least one tree, denoted as $FP(a, b)$ (Eq. (8)). The risk related to FPs is denoted as $risk_{FP}(a, b)$ and is computed as the Pearson correlation coefficient (ρ) of scores across the FP set (Eq. (9)). A similar process is performed to estimate the FN set $FN(a, b)$ and the corresponding risk $risk_{FN}(a, b)$ (Eqs. (10)–(11)). The final risk $risk(a, b)$ of the pair is estimated by averaging $risk_{FP}(a, b)$ and $risk_{FN}(a, b)$, which gives equal weight to the two types of risk (Eq. (12)).

$$FP(a, b) = \{i | y_i = 0 \wedge min(Q(i, a), Q(i, b)) < 0.5\} \tag{8}$$

$$risk_{FP}(a, b) = \rho(Q[i, a], Q[i, b]) | i \in FP(a, b) \tag{9}$$

$$FN(a, b) = \{i | y_i = 1 \wedge min(Q(i, a), Q(i, b)) < 0.5\} \tag{10}$$

$$risk_{FN}(a, b) = \rho(Q[i, a], Q[i, b]) | i \in FN(a, b) \tag{11}$$

$$risk(a, b) = \frac{risk_{FP}(a, b) + risk_{FN}(a, b)}{2} \tag{12}$$

Having computed all the pairwise risk estimations, a symmetric risk matrix K is simply formed by arranging all the pairwise risk estimations, as in Eq. (13).

$$K = \begin{bmatrix} risk(1,1) & \dots & risk(1,k) \\ \vdots & risk(a,b) & \vdots \\ risk(k,1) & \dots & risk(k,k) \end{bmatrix} \in \mathbb{R}^{k \times k} \tag{13}$$

Assuming a vector of tree weights w, the total expected performance $M = w^T r$ indicates the participation of high performing trees. Similarly, the total risk $V = w^T K w$ captures the participation of trees that tend to make similar FPs or FNs. We design a utility function that encourages high values of M and penalizes large values of V. Then, quadratic programming is applied on this utility function to obtain the tree weights w that solve the optimization problem in Eq. (14). This formalization is common in ML literature and can be considered as a regularized optimization task. Specifically, the main objective of this process is to allocate the largest portion of weight to high performing trees (by maximizing $w^T r$), with a regularization penalty that discourages the existence of strong correlations between trees (by minimizing $\lambda w^T K w$). The parameter λ controls the relative strength of the regularizer.

$$\begin{aligned} \max_{w} \quad & w^T r - \lambda w^T K w \\ \text{s.t.} \quad & \sum_{i=0}^{k} w_i = 1 \\ & 0 \leq w_i \leq 1 \end{aligned} \tag{14}$$

A pseudocode illustration of the proposed method for binary classification is presented in Algorithm 1 of Appendix C.

4.4. Markowitz random forest for regression

The general outline for the MRF regression method is similar to the one presented for classification. We define $\hat{Y}_{ij}$ as the numerical prediction of the jth tree on the ith example. The first difference is the estimation of performance vector r. For this, negExpMAPE is defined the negative exponential of the mean absolute percentage error (MAPE) (Eq. (15)). This performance definition offers three benefits: it is scale independent, ranges from 0 to 1 and higher values are more preferable. The performance vector r contains the negExpMAPE scores for all trees (Eq. (16)). If $\hat{Y}_{ij} = y_i$, the corresponding absolute percentage error is estimated as 0.

$$negExpMAPE(y, \hat{Y}_{:j}) = exp(-MAPE(y, \hat{Y}_{:j})) \in [0, 1] \tag{15}$$

$$M = [negExpMAPE(y, \hat{Y}_{:1}), \dots, negExpMAPE(y, \hat{Y}_{:n})] \in [0, 1]^k \tag{16}$$

The risk matrix for regression is computed as follows. For every entry of tree/sample, we compute the regression residual (RS) presented in Eq. (17). The signed residuals are used instead of the absolute residuals, in order to capture if two trees tend to simultaneously overestimate or underestimate the actual targets. The RS values are used to populate a performance matrix Q, as in Eq. (18). Then, the risk matrix K is computed as the covariance matrix of Q (Eq. (19)).

$$RS(y_i, \hat{Y}_{ij}) = (y_i - \hat{Y}_{ij}) \in \mathbb{R} \tag{17}$$

$$Q = \begin{bmatrix} RS(1,1) & \dots & RS(1,k) \\ \vdots & RS(i,j) & \vdots \\ RS(m,1) & \dots & RS(m,k) \end{bmatrix} \in \mathbb{R}^{m \times k} \tag{18}$$

$$K = cov(Q) \in \mathbb{R}^{k \times k} \tag{19}$$

Having computed the necessary inputs r and K for MVA, the remaining steps are identical to those of the classification scenario presented in the previous section. In the regression scenario, the optimization method allocates high weights to trees with low MAPE errors (by maximizing $w^T r$), whereas the regularization penalty discourages trees that simultaneously overestimate or underestimate the actual targets (by minimizing $\lambda w^T K w$). A pseudocode illustration of the proposed method for regression is presented in Algorithm 2 of Appendix C.

4.5. Markowitz random forest for multi-class classification

Section 4.3 developed the MRF formalization for binary classification problems. It is possible to extend this approach to handle multi-class classifications problems. For this purpose, we redefine the expected performance vector and the risk matrix. As an expected performance vector r, we consider the F_1_macro score of each tree, which indicates its predictive ability equally among the different classes. This is shown in Eq. (20). We define $P(i, j, c)$ as the predicted probability for class c on the i_{th} example by the j_{th} tree. Note that $P(:, j, :)$ corresponds to all predictions made by the jth tree.

$$r = [F_1\_macro(P(:, 1, :), y), \dots, F_1\_macro(P(:, k, :), y)] \in [0, 1]^k \tag{20}$$

Assume also that $D(i, j)$ is the class that receives the highest probability on the i_{th} example by the j_{th} tree, as in Eq. (21).

$$D(i, j) = arg_c max\{P(i, j, c)\} \in [1, 2, \dots, C] \tag{21}$$

The binary setting considered two types of risk based on FP and FN examples, which are the mistakes related to two classes (positive and negative) for a pair of trees. To generalize this for more than two classes, we consider that each class carries a different type of risk. For each class $c \in \{1, 2, \dots, C\}$ and every pair of trees (a, b), we detect the set of mistakes on this class from either of the trees in the pair, denoted as $FS_c(a, b)$ (Eq. (22)). Following this, the risk related to each class c is estimated by computing the correlation coefficient (ρ) on the class mistakes across all predicted probabilities (Eq. (23)).

$$FS_c(a, b) = \{i | (D(i, a) \neq c \vee D(i, b) \neq c) \wedge y_i = c\} \tag{22}$$

$$risk_c(a, b) = \rho_{i \in FS_c(a,b)}\{P(i, a, :), P(i, b, :)\} \tag{23}$$

The risks related to the classes are averaged in a single scalar $risk(a, b)$, a process that gives equal weight to the different types of risk (Eq. (24)). Then, these can be arranged in a matrix to form the risk matrix K, as in Eq. (25).

$$risk(a, b) = \mathbb{E}_c[risk_c(a, b)] \tag{24}$$

$$K = \begin{bmatrix} risk(1,1) & \dots & risk(1,k) \\ \vdots & risk(a,b) & \vdots \\ risk(k,1) & \dots & risk(k,k) \end{bmatrix} \in \mathbb{R}^{k \times k} \tag{25}$$

The remaining optimization steps are equivalent to those describe in the binary classification case (Section 4.3). In the multi-class setting, MVA attempts to allocate high weights to trees that achieve high performance, considering all classes equally (by maximizing $w^T r$). This is regularized with a penalty that aims to discourage trees that make highly correlated mistakes (by minimizing $\lambda w^T K w$).

This penalty influences the ensemble in two ways. First, it discourages trees from misclassifying the same examples. Second, even when trees do misclassify the same examples, it encourages them to predict different incorrect classes, therby reducing the concentration of predicted probability on a single mistaken class. A pseudocode illustration of the proposed method for multi-class classification is presented in Algorithm 1 of Appendix C.

4.6. Supervised methods and tree weighting benchmarks

Whereas the baseline method in this work is the traditional RF, it is also important to evaluate how MRF fairs against previously proposed tree weighting schemes. Therefore, we also compare our new method with the score-weighted forest (SWRF) and the OOB-weighted random forest (OWRF), based on the works of Li et al. [22] and Gajowniczek et al. [27], respectively. Generally, assuming a performance score has been given to each tree, the computed weights for the two methods

are estimated as in Eqs. (26)–(27). In these Equations, w_j denotes the weight for the jth tree, $score_{\langle train \rangle}(j)$ refers to the performance of the tree on the entire training set (including INB and OOB portions), whereas $score_{\langle OOB \rangle}(j)$ refers only to the OOB score of the particular tree.

$$w_j = \frac{score_{\langle train \rangle}(j)}{\sum_{z=1}^{k} score_{\langle train \rangle}(z)} \tag{26}$$

$$w_j = \frac{score_{\langle OOB \rangle}(j)}{\sum_{z=1}^{k} score_{\langle OOB \rangle}(z)} \tag{27}$$

Subsequently, in order to specify the above methods for the cases of classification and regression, we only need to determine the scoring functions. Based on the aforementioned works, for classification tasks we use accuracy score and for regression we consider the reciprocal of the mean absolute error (MAE) as a score $1/(MAE + \epsilon)$, where ϵ is a small floating number to avoid division by zero.

We also introduce another benchmark method named Independent Variance Random Forest (IVRF). IVRF is a simplification of MRF that considers only the independent variance of each tree and ignores the pairwise tree covariances. Assuming that the expected performance of a tree is r_j and the corresponding variance is σ_j^2, IVRF assigns this tree a score $score_j = r_j/\sigma_j^2$. The final weights are derived by scaling the tree scores so that they sum to 1. The comparisons between MRF and IVRF in Section 5 provide evidence that the consideration of pairwise tree covariance can lead to superior results.

Furthermore, we compare the performance of MRF against five well-received supervised learning models : XGBoost, ERT, OblF and RotF. XGBoost is an advanced gradient boosting model that achieves state-of-the-art results in several problems. ERT is an extremely randomized version of RF that reduces the tendency of trees to overfit the data. Therefore, these comparisons allow us to investigate the potential of MRF to achieve competitive results against methods favored by the ML community.

The remaining supervised methods consist of recently proposed variants of Random Forest (RF) that rely on multivariate splitting. OblF employs Support Vector Machines (SVM) at each node to create multivariate splits. RotF applies Principal Component Analysis (PCA) to randomly partitioned feature subsets, which enhances data diversity and makes it an implicitly multivariate method.

4.7. Hyper-parameter tuning

Our hyper-parameter tuning process is similar for the regression and classification tasks. We use the Tree-structured Parzen Estimator (TPE) optimizer from the hyperopt package to tune both RF, XGBoost and ERT using three-fold CV (3-CV) on the training portion of the data. The search space distribution that was considered for each hyper-parameter is presented in Table B.4 of Appendix B.

The optimization objectives for binary classification, regression and multi-class classification are PRAUC, MSE and F_1_macro, respectively. A sequence of 20 optimization steps is considered in each learning task. For classification tasks, class weights that are inversely proportional to class frequencies are considered to handle class imbalance. The randomized ensembles RF and ERT that are trained during 3-CV are combined through model aggregation, whereas the boosting ensemble XGBoost is retrained on the full training set.

The four weighting methods are applied on the same standard RF model that is tuned once for each task. This makes the results of tree weighting more comparable, as the forest is fixed and only the tree weights differ in each variant. MRF considers only the out-of-sample portion of the training set of each tree to estimate expected performance, while the entire training set is used for the estimation of the risk matrix. Furthermore, MRF has an additional regularization coefficient λ, that is fixed to a reasonable value of 0.5 for every dataset and learning task. Additionally, as suggested in the PyPortfolioOpt manual, an inverse L_2 norm penalty with a coefficient of 0.1 is applied to encourage non-zero weights.

5. Experimental results

This section describes the supervised learning experiments conducted in this work. First, it presents the processes of cross-validation, hyperparameter optimization, and statistical significance testing. Then, it details the experimental results for each method on the three learning tasks, elaborating on the most significant findings.

5.1. Experimental design

Experiments relating to three different learning tasks were performed on the 15 datasets mentioned in Section 4.1. For every dataset, the main experiment is repeated for 20 random trials, in which the data are always randomly split into train/test at the respective ratios of 80% and 20%. Furthermore, a 3-fold cross validation (3-CV) process is applied for each random trial to the training portion of the data, which enables hyper-parameter optimization. For the classification tasks, both the train/test split and the 3-CV on training data are stratified with respect to the classes. The reported performance metrics are estimated as averages over the random trial test results.

Additionally, the Friedman and post-hoc Nemenyi statistical tests [48] are applied across the random trials to investigate whether there are statistical differences in performance (Friedman), and for which pairs of learning methods (Nemenyi). Every Friedman test that was performed revealed a statistically significant difference, so the corresponding post-hoc Nemenyi tests for all learning tasks and datasets are provided in Tables A.1–A.3 of Appendix A.

5.2. Tree weighting for binary classification

The test results for the binary classification tasks are provided in Table 4, with the performance estimates retrieved by averaging across 20 random trials. For each dataset, the methods are sorted by their PRAUC score, which is deemed the most critical metric. PRAUC indicates the ability of the method to detect a positive minority class and is independent to the selection of a decision threshold.

XGBoost achieves the highest PRAUC scores in two datasets (aids and bank), MRF is the best performing in two other datasets (credit and shop), and RotF is the best performing in one dataset (bands). This suggests that the optimal tree ensemble approach varies by dataset, indicating that practitioners should experiment with multiple tree ensemble methods. Furthermore, MRF is second best performing for the aids and banks dataset, whereas XGBoost is among the lowest performing for credit and shop. Therefore, MRF achieves the best average rank in terms of PRAUC, and is followed by XGBoost. ERT and OblF are outperformed across all metrics by other methods for every binary classification task.

The Nemenyi post-hoc tests provided in Table A.1 of Appendix A reveal which performance differences should be considered statistically significant. Notably, MRF is able to outperform other supervised learning methods (RotF, OblF, ERT and XGBoost) with statistical significance on two datasets (credit and shop).

In terms of the threshold sensitive metrics, XGBoost generally achieves the best precision in all datasets, except for credit, in which MRF is better. Regarding recall, MRF achieves notable improvements for aids and banks at 84.18% and 92.28%, respectively, as well as the best average recall. F_1 seems to be highly data dependent, with MRF leading in bank and shop, IVRF in credit and aids, and RotF in bands.

MRF achieves low performance on the smallest dataset (bands), which we presume has too few examples for an accurate estimation of the risk matrix. However, the pairwise tests indicate that there are no statistically significant differences for this dataset among the best performing methods.

ROC and accuracy are two metrics that do not consider the effect of class imbalance. XGBoost presents the highest ROC and accuracy, but at a significant cost of a low recall compared to other methods. Regarding

Table 4
Predictive performance on test set for binary classification.

Data	Method	PRAUC	ROC	F_1	Precision	Recall	acc.	Time
aids	XGB [10]	**0.8644**	**0.9414**	0.7570	0.8148	0.7125	**0.8901**	**0.0502**
	MRF	0.8587	0.9410	0.7720	0.7136	**0.8418**	0.8792	2.3863
	IVRF	0.8499	0.9384	**0.7744**	0.7240	0.8332	0.8820	1.2378
	SWRF [22]	0.8464	0.9373	0.7739	0.7269	0.8284	0.8824	0.3250
	OWRF [27]	0.8464	0.9372	0.7737	0.7267	0.8279	0.8822	0.3509
	RF [4]	0.8455	0.9369	0.7733	0.7274	0.8264	0.8822	0.2554
	RotF [13]	0.8108	0.9164	0.7406	**0.8341**	0.6668	0.8867	2.7725
	ERT [9]	0.7974	0.9143	0.7362	0.6951	0.7841	0.8633	0.2105
	ObIF [11]	0.7080	0.8971	0.7158	0.6059	0.8769	0.8305	0.4659
bands	RotF [13]	**0.8133**	**0.8473**	**0.7201**	0.7691	0.6815	**0.7789**	3.7676
	XGB [10]	0.7964	0.8215	0.6327	**0.7708**	0.5554	0.7381	**0.0451**
	OWRF [27]	0.7916	0.8179	0.6541	0.7078	0.6196	0.7266	0.4885
	SWRF [22]	0.7914	0.8177	0.6556	0.7093	0.6207	0.7280	0.4387
	RF [4]	0.7914	0.8177	0.6548	0.7067	0.6217	0.7261	0.3545
	IVRF	0.7910	0.8163	0.6424	0.7072	0.6043	0.7211	1.6720
	MRF	0.7889	0.8153	0.6477	0.7037	0.6109	0.7225	4.7822
	ERT [9]	0.7821	0.8088	**0.6823**	0.6858	**0.6859**	0.7317	0.2343
	ObIF [11]	0.7089	0.7364	0.6305	0.5839	0.6967	0.6564	0.2010
bank	XGB [10]	**0.6590**	**0.9461**	0.5224	**0.6916**	0.4216	**0.9135**	**0.1856**
	MRF	0.6474	0.9443	**0.5929**	0.4364	**0.9248**	0.8569	26.3127
	IVRF	0.6291	0.9383	0.5841	0.4310	0.9059	0.8546	11.3719
	SWRF [22]	0.6283	0.9380	0.5834	0.4303	0.9059	0.8542	2.6749
	OWRF [27]	0.6283	0.9380	0.5835	0.4304	0.9059	0.8543	2.4041
	RF [4]	0.6280	0.9379	0.5833	0.4304	0.9052	0.8543	0.8174
	RotF [13]	0.5600	0.9184	0.5134	0.6201	0.4384	0.9064	49.2010
	ERT [9]	0.5389	0.8970	0.5173	0.4153	0.6857	0.8558	0.7604
	ObIF [11]	0.4972	0.8921	0.4924	0.3573	0.8067	0.8134	0.8926
credit	MRF	**0.5580**	0.7813	0.5448	0.5232	0.5685	0.7899	21.1384
	IVRF	0.5566	**0.7818**	**0.5455**	0.5210	0.5725	0.7890	9.2606
	SWRF [22]	0.5566	**0.7818**	0.5453	0.5207	0.5724	0.7888	2.5570
	OWRF [27]	0.5566	**0.7818**	0.5454	0.5208	0.5725	0.7889	2.1465
	RF [4]	0.5565	**0.7818**	0.5453	0.5207	0.5724	0.7888	1.4164
	XGB [10]	0.5424	0.7745	0.2439	0.4197	0.1739	0.8004	**0.0667**
	ERT [9]	0.5398	0.7756	0.5379	0.5017	**0.5801**	0.7796	0.4329
	RotF [13]	0.4682	0.7292	0.4276	**0.6075**	0.3300	**0.8046**	23.2335
	ObIF [11]	0.4488	0.7216	0.5149	0.4915	0.5425	0.7741	0.3956
shop	MRF	**0.7448**	**0.9336**	**0.6636**	0.5572	0.8209	0.8709	7.8874
	IVRF	0.7312	0.9301	0.6617	0.5530	0.8241	0.8693	4.2798
	SWRF [22]	0.7264	0.9289	0.6608	0.5519	0.8240	0.8689	1.9827
	OWRF [27]	0.7264	0.9289	0.6609	0.5520	0.8240	0.8689	1.2350
	RF [4]	0.7244	0.9284	0.6607	0.5516	**0.8242**	0.8688	0.4046
	XGB [10]	0.7194	0.9218	0.6136	**0.7411**	0.5406	**0.8982**	**0.0507**
	RotF [13]	0.6340	0.8884	0.5725	0.7013	0.4847	0.8881	11.5590
	ObIF [11]	0.5855	0.8798	0.6620	0.5638	0.8020	0.8731	0.7362
	ERT [9]	0.5648	0.8691	0.5087	0.3699	0.8151	0.7559	0.3801

the required training execution times, MRF adds a small overhead that varies between 1 and 20 s, which indicates the applicability of the method on real datasets. Due to the repeated application of PCA on feature partitions, RotF presents significantly higher execution times compared to other methods. A computational complexity analysis for the asymptotic behavior of MRF is presented in Section 6.

An additional investigation was conducted regarding the distribution of tree weights by each method. The findings were consistent across all experiments, and provide insight regarding the weighting behavior of MRF. Fig. 1 illustrates the case for the shoppers dataset. While the weights of SWRF and OWRF are concentrated near to the original weights, MRF spreads the tree weights much further. The weighting behavior of IVRF falls somewhere in between.

We presume that SWRF and OWRF are limited by individual tree performance estimates that do not vary much. In contrast, the risk matrix used by MRF provides more information, guiding it to better determine the unique contribution of each tree. Furthermore, MRF can eliminate trees that do not contribute to the ensemble, those that are not only low-performing but also similar to each other, by setting their weights to zero. Similarly, high-performing trees that also offer variety by making different mistakes receive the upper-end of weights.

Table 5 presents the ranking statistics for each model across five binary classification tasks. MRF achieved the best average rank at 2.6,

Table 5
Model ranking statistics across five binary classification tasks.

Method	MRF	IVRF	XGB [10]	SWRF [22]	OWRF [27]	RF [4]	RotF [13]	ERT [9]	ObF
mean	**2.60**	3.20	3.20	3.60	4.20	5.40	6.00	8.00	8.80
min	**1.00**	2.00	**1.00**	3.00	3.00	5.00	**1.00**	7.00	8.00
max	7.00	6.00	6.00	**4.00**	5.00	6.00	8.00	9.00	9.00
std	2.51	1.64	2.59	0.55	0.84	0.55	2.83	0.71	0.45

followed by XGBoost and IVRF at 3.2. However, contrary to IVRF, XGBoost is the best performing for several datasets. Furthermore, MRF achieves a lower rank standard deviation compared to XGBoost (2.51 vs 2.59).

SWRF and OWRF achieve average rankings of 3.6 and 4.2. These are followed by RF, RotF, ERT and ObIF at the worst positions. RotF presents the highest standard deviation in performance (2.83). Interestingly, MRF is the only tree weighting method that outperforms XGBoost in terms of average ranking. Similarly, it outperformed forests that are based on multivariate splits, such RotF and ObIF. This evidence highlights the potential for further research into enhanced RF models that leverage advanced tree weighting techniques. The interpretation of the aforementioned results leads to some practical guidelines regarding the advantages and limitations of MRF, which are discussed in Section 7.

5.3. Tree weighting for regression

The test results for the regression tasks are provided in Table 6. As a first observation, weighted versions of the RF generally outperform the other supervised learning methods. This suggests the potential of MRF to achieve competitive predictive performance in specific domains. As the following paragraphs will reveal, the performance improvements of MRF are pronounced in the regression setting.

Specifically, MRF achieves the best MAE score for every dataset, and the best MAPE in 3 datasets (insurance, news and wine). It is also on the high-end of RMSE scores, being the best for car and superconduct. R-squared is more depended on the dataset, but seems to favor RF variants.

The Nemenyi tests in Table A.2 of Appendix A reveal that MRF offers a statistically significant performance improvement in MAE against RF and XGBoost for most datasets. Furthermore, as was the case for binary classification, the training time overhead for MRF in regression tasks remain minor and should not pose an issue for practical applications.

Table 7 presents the ranking statistics for regression, in which the benefits of MRF are more pronounced compared to classification. Specifically, MRF achieves the highest ranking in all problems with an average rank of 1. It is followed by the remaining weighted versions of RF, which achieve average rankings between 3 and 3.8. The next model is ObIF and the traditional RF, at average ranks of 5.4. ERT and XGB are trailing behind with the lowest performance in these tasks. Once again, these results underscore the potential of RF models when enhanced by tree weighting schemes, allowing them to surpass even multivariate-split based forests.

5.4. Multi-class classification experiments

The results for the multi-class experiments are shown in Table 8. Multi-class predictive models are typically evaluated through F_1_macro and F_1_micro scores. The former weights each class equally, whereas the latter weights each example equally. In the context of class imbalance we emphasize F_1_macro, as it indicates the model's ability to achieve balanced performance across all classes, regardless of their sizes.

In these experiments, RotF achieves the best F_1_macro for three datasets (pen, plates, statlog image), with MRF being the best in the remaining datasets (room, student). However, whereas RotF shows high variability in performance (e.g., achieving the 5_{th} position in

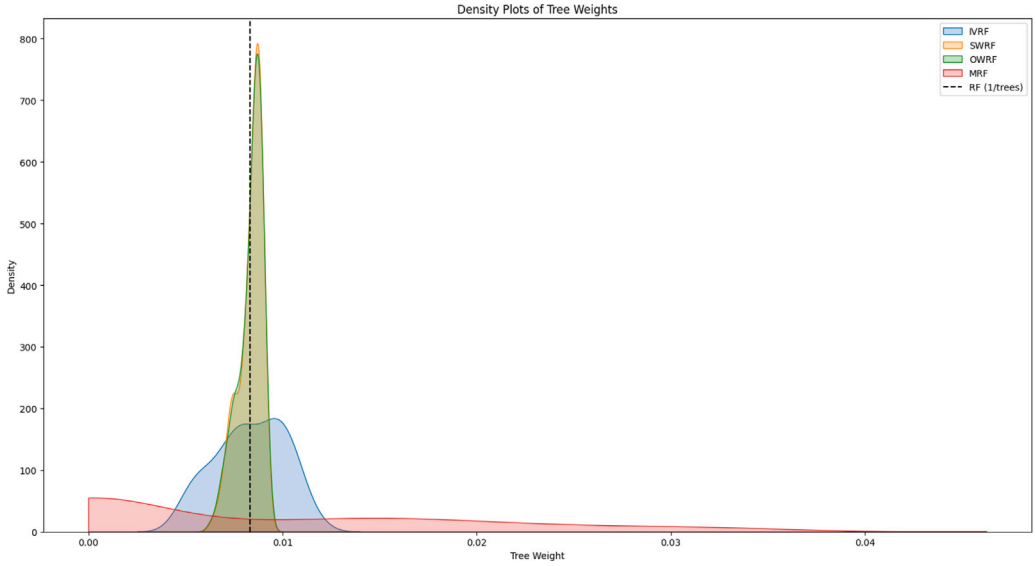


Fig. 1. Distribution of weights in the shoppers classification task.

Table 6
Predictive performance on test set for regression.

Dataset	Method	MAE	RMSE	MAPE	r^2	Time
car	MRF	**13 538.9370**	201 374.5060	1.763920e+01	0.0749	5.1124
	IVRF	13 540.0357	201 427.7928	1.709210e+01	0.0689	4.2450
	OblF [11]	13587.0829	**201160.0708**	**1.681560e+01**	**0.0957**	0.0000
	SWRF [22]	13 963.6273	201 497.5219	1.807640e+01	0.0641	4.6618
	OWRF [27]	13 965.1169	201 499.8911	1.807910e+01	0.0639	6.0294
	ERT [9]	13 989.5642	201 648.1633	1.757700e+01	0.0503	0.0000
	RF [4]	14 062.3812	201 593.8683	1.822060e+01	0.0549	0.0000
	XGB [10]	15 502.6324	202 184.9734	2.061720e+01	−0.0016	0.2997
insur.	MRF	**0.1029**	0.2306	**2.211050e+14**	0.0432	1.9240
	OblF [11]	0.1068	**0.2296**	2.414450e+14	**0.0511**	0.0000
	SWRF [22]	0.1069	0.2297	2.419776e+14	0.0506	0.8595
	OWRF [27]	0.1070	0.2297	2.424910e+14	0.0506	0.9645
	IVRF	0.1070	0.2297	2.425425e+14	**0.0506**	1.0932
	RF [4]	0.1070	0.2297	2.425425e+14	0.0506	0.0000
	ERT [9]	0.1076	0.2300	2.433680e+14	0.0480	0.0000
	XGB [10]	0.1106	0.2337	2.501693e+14	0.0174	0.0538
news	MRF	**3002.4437**	11 710.5283	**1.858200e+00**	0.0243	4.7174
	IVRF	3031.9519	11 705.9553	1.888400e+00	**0.0250**	3.5569
	OWRF [27]	3034.4942	11 705.9091	1.891200e+00	**0.0250**	2.9762
	SWRF [22]	3034.4989	**11 705.8789**	1.891200e+00	**0.0250**	3.2542
	RF [4]	3035.4693	11 705.9881	1.892000e+00	**0.0250**	0.0000
	OblF [11]	3057.9386	11729.8550	1.938500e+00	0.0208	0.0000
	ERT [9]	3058.8740	11 736.2838	1.951000e+00	0.0196	0.0000
	XGB [10]	3155.9293	11 827.7871	2.016000e+00	0.0031	0.1126
supercond.	MRF	**7.1624**	**11.3945**	7.419200e+00	**0.8893**	2.8350
	SWRF [22]	7.1765	11.4113	7.439100e+00	0.8890	2.0192
	OWRF [27]	7.1767	11.4115	7.439800e+00	0.8890	2.1513
	RF	7.1781	11.4131	7.441500e+00	0.8889	0.0000
	IVRF	7.1896	11.4410	**7.065300e+00**	0.8884	2.2087
	XGB [10]	7.5077	11.5672	7.987000e+00	0.8859	0.2474
	ERT [9]	8.8304	13.1963	9.060900e+00	0.8515	0.0000
	OblF [11]	12.3505	17.0593	1.468720e+01	0.7519	0.0000
wine	MRF	**0.5242**	0.6710	**9.350000e−02**	0.4076	0.4569
	SWRF [22]	0.5252	0.6703	9.360000e−02	**0.4089**	0.2819
	IVRF	0.5252	**0.6702**	9.360000e−02	**0.4089**	0.3755
	OWRF [27]	0.5253	0.6703	9.370000e−02	0.4088	0.3256
	RF [4]	0.5253	0.6703	9.370000e−02	0.4088	0.0000
	OblF [11]	0.5823	0.7365	1.038000e−01	0.2864	0.0000
	XGB [10]	0.5537	0.7031	9.850000e−02	0.3496	0.0629
	ERT [9]	0.5791	0.7312	1.034000e−01	0.2967	0.0000

Table 7
Model ranking statistics across the five regression tasks.

Method	MRF	SWRF [22]	IVRF	OWRF [27]	OblF [11]	RF [4]	ERT [9]	XGB [10]
mean	**1.0**	3.0	3.4000	3.8000	5.4000	5.4000	6.8000	7.2000
min	**1.0**	2.0	2.0000	3.0000	2.0000	4.0000	6.0000	6.0000
max	**1.0**	4.0	5.0000	5.0000	8.0000	7.0000	7.0000	8.0000
std	0.0	1.0	1.5166	0.8367	2.7928	1.1402	0.4472	1.0954

Table 8
Predictive performance on test set for multi-class classification.

Dataset	Method	F_1_macro	F_1_micro	Accuracy	Elapsed
pen	RotF [13]	**0.9927**	**0.9927**	**0.9927**	4.3560
	MRF	0.9823	0.9821	0.9821	5.4343
	IVRF	0.9819	0.9817	0.9817	1.3910
	RF [4]	0.9819	0.9817	0.9817	0.3130
	XGB [10]	0.9809	0.9807	0.9807	0.3618
	ERT [9]	0.9732	0.9730	0.9730	0.4247
	OblF [11]	0.8276	0.8339	0.8339	0.6581
	OWRF [27]	0.1140	0.1964	0.1964	0.5028
	SWRF [22]	0.1140	0.1964	0.1964	0.4991
plates	RotF [13]	**0.8052**	**0.7731**	**0.7731**	8.8178
	MRF	0.7806	0.7478	0.7478	2.8049
	RF [4]	0.7729	0.7366	0.7366	0.2457
	IVRF	0.7726	0.7361	0.7361	0.6966
	XGB [10]	0.7589	0.7514	0.7514	0.1781
	ERT [9]	0.7100	0.6937	0.6937	0.1975
	OblF [11]	0.4623	0.4744	0.4744	0.7359
	OWRF [27]	0.1586	0.4148	0.4148	0.3440
	SWRF [22]	0.1584	0.4145	0.4145	0.3163
room	MRF	**0.9890**	**0.9969**	**0.9969**	2.7024
	RotF [13]	0.9889	0.9968	0.9968	5.7058
	ERT [9]	0.9876	0.9963	0.9963	0.2216
	RF [4]	0.9876	0.9965	0.9965	0.2699
	IVRF	0.9874	0.9964	0.9964	0.9756
	XGB [10]	0.9831	0.9950	0.9950	0.2239
	OblF [11]	0.8538	0.9530	0.9530	0.1468
	OWRF [27]	0.4786	0.8575	0.8575	0.4060
	SWRF [22]	0.4786	0.8575	0.8575	0.3887
statlog image	RotF [13]	**0.9820**	**0.9820**	**0.9820**	4.1006
	MRF	0.9654	0.9655	0.9655	2.6908
	IVRF	0.9625	0.9626	0.9626	0.7117
	RF [4]	0.9624	0.9624	0.9624	0.2449
	XGB [10]	0.9616	0.9615	0.9615	0.2126
	ERT [9]	0.9442	0.9444	0.9444	0.1977
	OblF [11]	0.8796	0.8826	0.8826	0.7236
	OWRF [27]	0.1727	0.2677	0.2677	0.3315
	SWRF [22]	0.1727	0.2677	0.2677	0.3093
student	MRF	**0.6993**	0.7489	0.7489	2.2699
	IVRF	0.6988	0.7481	0.7481	0.7894
	RF [4]	0.6988	0.7481	0.7481	0.2486
	ERT [9]	0.6853	0.7295	0.7295	0.3216
	RotF [13]	0.6820	0.7594	0.7594	18.4609
	XGB [10]	0.6685	**0.7603**	**0.7603**	0.0739
	OblF [11]	0.6346	0.7165	0.7165	0.2020
	OWRF [27]	0.5093	0.7015	0.7015	0.3602
	SWRF [22]	0.5092	0.7014	0.7014	0.3440

the student dataset), MRF presents lower variance in performance by achieving at least 2_{nd} position in every dataset. For this reason, it will be shown later that MRF achieves the best average ranking.

The results are similar for F_1_micro and accuracy, although XGBoost achieves the best in the student dataset. However, XGBoost deteriorates significantly the performance of minority classes in order to achieve an increased performance of the majority classes.

Although MRF consistently achieves high scores in F_1_macro, more evidence is required to prove that it can outperform RF, XGBoost and IVRF with statistical significance. However, according to the Nemenyi results in Table A.3, it is proven superior to previously proposed tree weighting methods (SWRF, OWRF) with statistical significance in every dataset.

As was the case for binary classification and regression, MRF achieves the highest average ranking (1.6) in the multi-class setting (Table 9). RotF is the next best performing method, with an average ranking of 2, while traditional RF, ERT and XGB outperform the previously proposed tree weighting approaches OWRF and SWRF. Particularly, the tree weighting approaches OWRF and SWRF achieve the lowest two ranks across all datasets. This evidence suggests that the effectiveness of previously proposed tree-weighting methods is significantly lower than that of MRF in the multi-class setting. Furthermore, the fact that MRF can outperform multivariate-split based forests like RotF solely through optimal tree weighting within an RF model highlights the potential of this research direction.

The analysis reveals significant performance differences between tree-weighting methods that focus solely on individual tree performance and those that incorporate risk estimates based on tree covariance. While OWRF and SWRF fall short of traditional RF and occupy the lowest rankings, MRF and IVRF consistently rank among the top-performing methods, providing evidence that tree diversity is crucial for effective tree weighting in multi-class problems. Whereas standard tree-weighting methods fail to outperform an equally weighted RF, the proposed tree-weighting approach not only surpasses RF but also outperforms advanced multivariate-split based forests like RotF and OblF, demonstrating its superior potential to enhance RF performance.

6. Computational complexity analysis

The previous Section established that MRF can achieve statistically significant improvements in predictive performance. In this section, we consider the additional computational complexity that is required by the proposed tree weighting method.

MRF is applied on an existing RF, so it inherits the complexity of RF. For fully grown trees, the time complexity of RF is $TC(RF) = \mathcal{O}(Tkmlog(m))$, where m is the number of rows, k is the number of features considered in each split and T is the number of estimators. Note that if a maximum depth d is set, the complexity of RF is $TC(RF) = \mathcal{O}(Tkmd)$. The next paragraphs will derive the additional complexity due to tree weight allocation.

MRF initially must compute a covariance matrix of an $(n \times T)$ matrix of probabilities (or residuals), as well as an expected performance vector. The correlation matrix computation adds a complexity of $\mathcal{O}(T^2\, m)$, which overshadows the complexity of computing the expected performance vector, that is $\mathcal{O}(Tm)$. Following this, the mean–variance algorithm solves a Quadratic Programming (QP) problem in polynomial time, with an additional complexity factor of approximately $\mathcal{O}(T^3)$ [49]. The combination of the above yields the total time complexity for MRF, as presented in Eq. (28).

$$tc(MRF) = \mathcal{O}\left(\underbrace{Tkmlog(m)}_{\text{Random Forest}} + \underbrace{T^2m}_{\text{Matrix Estimation}} + \underbrace{T^3}_{\text{MVA}}\right) \tag{28}$$

The above complexity is equivalent for the worst, average and best case analysis. This result indicates that the additional time complexity depends mostly on the number of trees, and to a lesser extent on the number of rows. Considering that the number of trees is often a small constant, ranging from 10 to 1000, the complexity reduces to the same as RF. Therefore, the additional time complexity of MRF should not pose a significant issue for medium to high term computing systems that can train RF models. This is also in alignment with our empirical computation time estimations presented in Section 5.

Considering also the multi-class setting, the only difference is that the estimation of the covariance matrix uses predicted probabilities for all classes (c). Like the number of trees, the number of classes is also typically small for many applications.

$$tc(MRF) = \mathcal{O}\left(\underbrace{Tkmlog(m)}_{\text{Random Forest}} + \underbrace{T^2mc}_{\text{Matrix Estimation}} + \underbrace{T^3}_{\text{MVA}}\right) \tag{29}$$

Table 9
Model ranking statistics across the five multi-class classification task.

Method	MRF	RotF [13]	IVRF	RF [4]	ERT [9]	XGB [10]	OblF [11]	OWRF [27]	SWRF [22]
mean	**1.600**	2.000	3.40	3.600	5.000	5.400	7.0	8.0	9.0
min	**1.000**	**1.000**	2.00	3.000	3.000	5.000	7.0	8.0	9.0
max	**2.000**	5.000	5.00	4.000	6.000	6.000	7.0	8.0	9.0
std	0.548	1.732	1.14	0.548	1.414	0.548	0.0	0.0	0.0

The other weighting methods also require additional complexity, as they need to compute scores across all trees and examples. This is shown in Eq. (30).

$$tc(SWRF) = tc(OWRF) = tc(IVRF) = \mathcal{O}\left(\underbrace{Tkmlog(m)}_{\text{Random Forest}} + \underbrace{Tm}_{\text{Scoring}}\right) \quad (30)$$

For reference, we briefly overview the time complexities for the remaining methods, XGBoost, ERT, OblF and RotF. Assuming that there are no missing values in the data, the time complexity of XGBoost is presented in Eq. (31), in which d denotes the max depth of trees and n the number of features. For ERT, the asymptotic time complexity is equivalent to that RF Eq. (32).

$$tc(XGBoost) = \mathcal{O}(Tdnm\log(m)) \quad (31)$$

$$tc(ERT) = \mathcal{O}(Tkm\log(m)) \quad (32)$$

The algorithm of OblF is similar to RF, but it uses SVM to create multivariate splits at each node. The complexity of training an SVM on a dataset with m rows and k features is approximately $\mathcal{O}(m^2k)$. Due to the quadratic complexity, the root node's split dominates the overall complexity in the case of balanced splits. Based on this hypothesis, the average-case complexity for OblF is given in Eq. (33).

$$tc(\text{OblF}) = \mathcal{O}(Tm^2k) \quad (33)$$

Proceeding to the complexity of RotF, we assume that the dataset is partitioned into p equal groups of m rows and $f = n/p$ features. PCA is applied to each group, and this process is repeated for every tree before applying the random forest algorithm. We consider the SVD implementation of PCA, which for a dataset with m rows and f features has a complexity of $\mathcal{O}(\min(f^2\ m, fm^2))$. The final complexity for RotF is provided in Eq. (34). As shown, this is the most computationally intensive method, which aligns with the increased computation times observed in the experimental section.

$$tc(\text{RotF}) = \mathcal{O}\left(Tp\min\left(f^2\ m, fm^2\right) + Tkm\log(m)\right) \quad (34)$$

7. Discussion of research findings

By summarizing the findings of Section 5, we recommend general guidelines regarding the problem types and conditions for which MRF can be a potentially high performing model. Regardless of task, MRF always depends on an underlying RF and aims to improve its performance. Therefore, the first guideline is to consider MRF when the standard RF method achieves high performance. Second, datasets with very few examples may not provide enough rows for an accurate computation of the risk matrix, so the potential of MRF would be limited in this case. RotF was proven the best option in the case of a small dataset. IVRF may also occasionally outperform MRF for similar reasons, as it does not rely on a risk matrix estimation. The following paragraphs will elaborate on the task-specific guidelines.

In binary classification, MRF can be potentially useful when it is desirable to achieve a high PRAUC score, particularly in class imbalanced scenarios. This is achieved through accurate probability predictions that can discriminate classes at varying decision thresholds. It should also be considered when recall is more important than precision, in problems for which False Negatives (FNs) are more costly than False Positives (FPs).

For instance, there are problems in finance for which FNs are more expensive, such as fraud detection. Specifically, an undetected fraud (FN) has more severe effects compared to the costs of misreporting a transaction as fraudulent (FP). In medical diagnosis as well, undetected disease (FN) can often be more costly than an incorrect positive diagnosis (FP). On the contrary and according to our experiments, XGBoost is more likely to outperform MRF for problems that precision is more important. Examples in this category include email spam detection and classification-based recommender systems.

The regression experiments proved that MRF can offer significantly improvements in MAE and MAPE scores, and is very likely to outperform other ensemble models. This is relevant for problems in which all errors should be weighted equally, regardless of their magnitude. If high errors should be more penalized, different tree-weighting variants should be tested for optimal MSE score.

The previously proposed tree weighting methods SWRF and OWRF did not perform well on multi-class problems, indicating that this area should receive more attention. In this direction, MRF was proven significantly better in multi-class problems, particularly when it is important to achieve balanced predictive performance across all classes. RotF should be considered in the multi-class setting as it was highly effective, provided that the increased computational cost is not a limiting factor.

8. Conclusion

Initially, the traditional RF algorithm was designed to assign equal weights to the decision trees that constitute the ensemble. Previous works have attempted to improve upon this by assigning non-equal weights, based on the predictive performance of each independent tree. However, to the best of our knowledge, previous works have not considered the opportunity to promote tree variety through their weighting schemes, which is essential for the success of RF.

In this paper we propose MRF, an improved tree-weighted version of RF that relies on a novel weighting method. Along with individual tree performance, MRF also considers a regularization factor that promotes tree variety. Through a tree covariance matrix, it discourages the allocation of weight to trees that make similar mistakes, which facilitates the participation of diversified trees in the ensemble. Our method is based on MVA, a method from financial mathematics which aims to maximize portfolio return, while at the same time to minimize portfolio variance. Therefore, this also makes MRF a finance-inspired ML method.

Our experimental results on 15 public datasets provide evidence that our proposed method can provide notable improvements against the traditional RF algorithm and previously proposed tree weighting methods. This is achieved in the context of three supervised learning tasks: binary classification, regression and multi-class classification. Specifically, MRF provides significant improvements in PRAUC, MAE, F_1_macro. Additionally, Friedman and Nemenyi statistical tests revealed that MRF can outperform other weighted variants of RF, as well as XGBoost, ERT, OblF and RotF across several learning tasks and datasets with statistical signficance.

We plan several directions for future work. First, to investigate the application of tree weight optimization on RF variants that apply multivariate splits, such as double oblique and double rotation

Table A.1
Nemenyi post-hoc tests for binary classification.

Data		ERT	IVRF	MRF	OWRF	ObF	RF	RotF	SWRF	XGB
	ERT	1.0	**0.001**	**0.001**	**0.0014**	0.9	0.107	0.9	**0.0014**	**0.001**
	IVRF	**0.001**	1.0	0.9	0.6646	**0.001**	0.0566	**0.001**	0.6646	0.9
	MRF	**0.001**	0.9	1.0	0.5245	**0.001**	0.0278	**0.001**	0.5245	0.9
	OWRF	**0.0014**	0.6646	0.5245	1.0	**0.001**	0.9	0.0477	0.9	0.3737
aids	ObF	0.9	**0.001**	**0.001**	**0.001**	1.0	**0.0014**	0.3737	**0.001**	**0.001**
	RF	0.107	0.0566	0.0278	0.9	**0.0014**	1.0	0.6296	0.9	0.0128
	RotF	0.9	**0.001**	**0.001**	0.0477	0.3737	0.6296	1.0	0.0477	**0.001**
	SWRF	**0.0014**	0.6646	0.5245	0.9	**0.001**	0.9	0.0477	1.0	0.3737
	XGB	**0.001**	0.9	0.9	0.3737	**0.001**	0.0128	**0.001**	0.3737	1.0
	ERT	1.0	0.6646	0.8223	0.1641	0.0789	0.2549	0.3364	0.4512	0.5245
	IVRF	0.6646	1.0	0.9	0.9	**0.001**	0.9	0.9	0.9	0.9
	MRF	0.8223	0.9	1.0	0.9	**0.001**	0.9	0.9	0.9	0.9
	OWRF	0.1641	0.9	0.9	1.0	**0.001**	0.9	0.9	0.9	0.9
bands	ObF	0.0789	**0.001**	**0.001**	**0.001**	1.0	**0.001**	**0.001**	**0.001**	**0.001**
	RF	0.2549	0.9	0.9	0.9	**0.001**	1.0	0.9	0.9	0.9
	RotF	0.3364	0.9	0.9	0.9	**0.001**	0.9	1.0	0.9	0.9
	SWRF	0.4512	0.9	0.9	0.9	**0.001**	0.9	0.9	1.0	0.9
	XGB	0.5245	0.9	0.9	0.9	**0.001**	0.9	0.9	0.9	1.0
	ERT	1.0	**0.001**	**0.001**	**0.0022**	0.9	0.3024	0.9	**0.001**	**0.001**
	IVRF	**0.001**	1.0	0.9	0.6296	**0.001**	0.0156	**0.001**	0.7697	0.3737
	MRF	**0.001**	0.9	1.0	0.0566	**0.001**	**0.001**	**0.001**	0.107	0.9
	OWRF	**0.0022**	0.6296	0.0566	1.0	**0.001**	0.7697	0.0916	0.9	**0.0014**
bank	ObF	0.9	**0.001**	**0.001**	**0.001**	1.0	0.0278	0.5245	**0.001**	**0.001**
	RF	0.3024	0.0156	**0.001**	0.7697	0.0278	1.0	0.9	0.6296	**0.001**
	RotF	0.9	**0.001**	**0.001**	0.0916	0.5245	0.9	1.0	0.0477	**0.001**
	SWRF	**0.001**	0.7697	0.107	0.9	**0.001**	0.6296	0.0477	1.0	0.0035
	XGB	**0.001**	0.3737	0.9	**0.0014**	**0.001**	**0.001**	**0.001**	0.0035	1.0
	ERT	1.0	**0.001**	**0.001**	**0.0055**	0.2126	0.2126	0.6996	**0.0068**	0.9
	IVRF	**0.001**	1.0	0.9	0.9	**0.001**	0.2401	**0.001**	0.9	**0.001**
	MRF	**0.001**	0.9	1.0	0.5595	**0.001**	0.0401	**0.001**	0.5245	**0.001**
	OWRF	**0.0055**	0.9	0.5595	1.0	**0.001**	0.9	**0.001**	0.9	0.0566
defaults	ObF	0.2126	**0.001**	**0.001**	**0.001**	1.0	**0.001**	0.9	**0.001**	0.0335
	RF	0.2126	0.2401	0.0401	0.9	**0.001**	1.0	**0.001**	0.9	0.6296
	RotF	0.6996	**0.001**	**0.001**	**0.001**	0.9	**0.001**	1.0	**0.001**	0.2703
	SWRF	**0.0068**	0.9	0.5245	0.9	**0.001**	0.9	**0.001**	1.0	0.067
	XGB	0.9	**0.001**	**0.001**	0.0566	0.0335	0.6296	0.2703	0.067	1.0
	ERT	1.0	**0.001**	**0.001**	**0.001**	0.9	0.0044	0.6646	**0.001**	**0.001**
	IVRF	**0.001**	1.0	0.9	0.6646	**0.001**	0.0104	**0.001**	0.5945	0.0566
	MRF	**0.001**	0.9	1.0	0.0477	**0.001**	**0.001**	**0.001**	0.0335	**0.001**
	OWRF	**0.001**	0.6646	0.0477	1.0	**0.001**	0.6646	0.0044	0.9	0.9
shoppers	ObF	0.9	**0.001**	**0.001**	**0.001**	1.0	0.0128	0.8398	**0.001**	0.0017
	RF	0.0044	0.0104	**0.001**	0.6646	0.0128	1.0	0.5245	0.7347	0.9
	RotF	0.6646	**0.001**	**0.001**	0.0044	0.8398	0.5245	1.0	0.0068	0.2126
	SWRF	**0.001**	0.5945	0.0335	0.9	**0.001**	0.7347	0.0068	1.0	0.9
	XGB	**0.001**	0.0566	**0.001**	0.9	0.0017	0.9	0.2126	0.9	1.0

RF [50,51]. Second, to apply similar weight optimization techniques on deep learning ensembles, including ensemble random vector functional link neural networks (edRVFL) [52]. Fourth, to modify the static MRF weighting mechanism towards a dynamic weighting scheme that depends on the observation. Third, to compare the potential of other portfolio management methods for tree weight allocation. Finally, to investigate other ideas in financial mathematics that can lead to finance-inspired ML methods.

CRediT authorship contribution statement

Eleftherios Kouloumpris: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Data curation, Conceptualization. **Ioannis Vlahavas:** Writing – review & editing, Supervision, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Appendix A. Nemenyi post-hoc statistical tests

See Tables A.1–A.3.

Appendix B. Hyper-parameter search space

See Table B.4.

Appendix C. Proposed method algorithms

See Algorithms 1 and 2.

Table A.2
Nemenyi post-hoc tests for regression task.

Data		ERT	IVRF	MRF	OWRF	OblF	RF	SWRF	XGB
insur.	ERT	1.0	0.6568	**0.001**	0.0092	**0.0016**	0.7725	**0.001**	0.5603
	IVRF	0.6568	1.0	**0.001**	0.5796	0.2983	0.9	0.0161	0.0082
	MRF	**0.001**	**0.001**	1.0	0.0073	0.0336	**0.001**	0.4402	**0.001**
	OWRF	0.0092	0.5796	0.0073	1.0	0.9	0.4613	0.7532	**0.001**
	OblF	**0.0016**	0.2983	0.0336	0.9	1.0	0.2015	0.9	**0.001**
	RF	0.7725	0.9	**0.001**	0.4613	0.2015	1.0	0.0082	0.0161
	SWRF	**0.001**	0.0161	0.4402	0.7532	0.9	0.0082	1.0	**0.001**
	XGB	0.5603	0.0082	**0.001**	**0.001**	**0.001**	0.0161	**0.001**	1.0
car	ERT	1.0	0.018	0.0599	0.9	0.0599	0.0336	0.9	**0.001**
	IVRF	0.018	1.0	0.9	**0.001**	0.9	**0.001**	**0.001**	**0.001**
	MRF	0.0599	0.9	1.0	**0.0016**	0.9	**0.001**	**0.0016**	**0.001**
	OWRF	0.9	**0.001**	**0.0016**	1.0	**0.0016**	0.3975	0.9	0.0092
	OblF	0.0599	0.9	0.9	**0.0016**	1.0	**0.001**	**0.0016**	**0.001**
	RF	0.0336	**0.001**	**0.001**	0.3975	**0.001**	1.0	0.3975	0.8304
	SWRF	0.9	**0.001**	**0.0016**	0.9	**0.0016**	0.3975	1.0	0.0092
	XGB	**0.001**	**0.001**	**0.001**	0.0092	**0.001**	0.8304	0.0092	1.0
wine	ERT	1.0	**0.001**	**0.001**	**0.001**	0.9	0.0274	**0.001**	0.6761
	IVRF	**0.001**	1.0	0.9	0.6375	**0.001**	0.0496	0.9	**0.001**
	MRF	**0.001**	0.9	1.0	0.1625	**0.001**	0.0027	0.9	**0.001**
	OWRF	**0.001**	0.6375	0.1625	1.0	**0.001**	0.9	0.3975	0.0856
	OblF	0.9	**0.001**	**0.001**	**0.001**	1.0	0.0045	**0.001**	0.3556
	RF	0.0274	0.0496	0.0027	0.9	0.0045	1.0	0.0144	0.7532
	SWRF	**0.001**	0.9	0.9	0.3975	**0.001**	0.0144	1.0	**0.001**
	XGB	0.6761	**0.001**	**0.001**	0.0856	0.3556	0.7532	**0.001**	1.0
s. cond.	ERT	1.0	**0.001**	**0.001**	**0.001**	0.9	0.018	**0.001**	0.9
	IVRF	**0.001**	1.0	0.1195	0.9	**0.001**	0.9	0.5217	0.0599
	MRF	**0.001**	0.1195	1.0	0.5217	**0.001**	0.0073	0.9	**0.001**
	OWRF	**0.001**	0.9	0.5217	1.0	**0.001**	0.6761	0.9	0.0045
	OblF	0.9	**0.001**	**0.001**	**0.001**	1.0	**0.001**	**0.001**	0.1625
	RF	0.018	0.9	0.0073	0.6761	**0.001**	1.0	0.0856	0.4402
	SWRF	**0.001**	0.5217	0.9	0.9	**0.001**	0.0856	1.0	**0.001**
	XGB	0.9	0.0599	**0.001**	0.0045	0.1625	0.4402	**0.001**	1.0
news	ERT	1.0	**0.001**	**0.001**	**0.0016**	0.9	0.5217	0.0045	0.5603
	IVRF	**0.001**	1.0	0.9	0.5603	**0.001**	0.0021	0.3975	**0.001**
	MRF	**0.001**	0.9	1.0	0.0336	**0.001**	**0.001**	0.0144	**0.001**
	OWRF	**0.0016**	0.5603	0.0336	1.0	0.0057	0.4402	0.9	**0.001**
	OblF	0.9	**0.001**	**0.001**	0.0057	1.0	0.7147	0.0144	0.3556
	RF	0.5217	0.0021	**0.001**	0.4402	0.7147	1.0	0.5989	0.0035
	SWRF	0.0045	0.3975	0.0144	0.9	0.0144	0.5989	1.0	**0.001**
	XGB	0.5603	**0.001**	**0.001**	**0.001**	0.3556	0.0035	**0.001**	1.0

Algorithm 1 Markowitz Random Forest(n,λ) for classification

function meanVarianceAnalysis(r, K, λ)
  Objective: Solve quadratic optimization to maximize $w^\top r - \lambda w^\top K w$ with constraints $w^\top \mathbf{1} = 1$ and $w \geq 0$
  Return optimal weights w
rf ← Train standard random forest with n trees
▷ Replace F1-macro with PRAUC for binary classification tasks
for $j = 1, \ldots, n$ **do**
  treeScores[j] ← Estimate out-of-bag F1-macro of the jth tree
▷ Estimate a performance score for each example-tree pair
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, n$ **do**
    exampleScores[i, j] ← Predicted probability for correct class of ith example by the jth tree
▷ Estimate the risk for every class c and every pair of trees (a, b)
▷ Note that for binary classification this will estimate the average of FP and FN risk.
for $a = 1, \ldots, n$ **do**
  for $b = 1, \ldots, n$ **do**
    for $c = 1, \ldots, C$ **do**
      F(c, a, b) ← Set of examples that are falsely classified in class c by either tree a or b
      RiskF(c, a, b) ← Correlation between exampleScores[i,a] and exampleScores[i,b], $i \in$ F(c, a, b)
    riskMatrix[a,b] ← $\mathbb{E}_c$[RiskF(c, a, b)]
▷ Solve the quadratic optimization problem: maximize $w^\top r - \lambda w^\top S w$
treeWeights ← meanVarianceAnalysis(treeScores, riskMatrix, λ)
Output rf, treeWeights

Algorithm 2 Markowitz Random Forest(n,λ) for regression

function meanVarianceAnalysis(r, K, λ)
  Objective: Solve quadratic optimization to maximize $w^\top r - \lambda w^\top K w$ with constraints $w^\top \mathbf{1} = 1$ and $w \geq 0$
  Return optimal weights w
rf ← Train standard random forest with n trees
▷ Estimate a performance score for each tree that ranges between 0 and 1
for $j = 1, \ldots, n$ **do**
  error[j] ← Mean Absolute Percentage Error (MAPE) of jth tree
  treeScores[j] ← exp(- error[j])
▷ Estimate a performance score for each example-tree pair
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, n$ **do**
    exampleResidual[i, j] ← (actual target of ith example) - (prediction of jth tree on ith example)
▷ Estimate the nxn tree covariance matrix
riskMatrix ← Covariance(exampleResidual)
▷ Solve the quadratic optimization problem: maximize $w^\top r - \lambda w^\top S w$
treeWeights ← meanVarianceAnalysis(treeScores, riskMatrix, λ)
Output rf, treeWeights

Data availability

All datasets used in this work are publicly available.

Table A.3
Nemenyi post-hoc test results for multi-class classification task.

Data		ERT	IVRF	MRF	OWRF	ObF	RF	RotF	SWRF	XGB
	ERT	1.0	0.3024	0.3364	**0.001**	0.0477	0.2401	0.9	**0.001**	0.7347
	IVRF	0.3024	1.0	0.9	**0.001**	**0.001**	0.9	0.3364	**0.001**	**0.0017**
	MRF	0.3364	0.9	1.0	**0.001**	**0.001**	0.9	0.3737	**0.001**	**0.0022**
	OWRF	**0.001**	**0.001**	**0.001**	1.0	0.7347	**0.001**	**0.001**	0.9	0.0477
student	ObF	0.0477	**0.001**	**0.001**	0.7347	1.0	**0.001**	0.0401	0.6646	0.8748
	RF	0.2401	0.9	0.9	**0.001**	**0.001**	1.0	0.2703	**0.001**	**0.0011**
	RotF	0.9	0.3364	0.3737	**0.001**	0.0401	0.2703	1.0	**0.001**	0.6996
	SWRF	**0.001**	**0.001**	**0.001**	0.9	0.6646	**0.001**	**0.001**	1.0	0.0335
	XGB	0.7347	**0.0017**	**0.0022**	0.0477	0.8748	**0.0011**	0.6996	0.0335	1.0
	ERT	1.0	0.0566	0.023	0.0916	0.9	0.107	**0.001**	0.0916	0.3364
	IVRF	0.0566	1.0	0.9	**0.001**	**0.001**	0.9	0.1426	**0.001**	0.9
	MRF	0.023	0.9	1.0	**0.001**	**0.001**	0.9	0.2703	**0.001**	0.9
	OWRF	0.0916	**0.001**	**0.001**	1.0	0.6996	**0.001**	**0.001**	0.9	**0.001**
pen	ObF	0.9	**0.001**	**0.001**	0.6996	1.0	**0.0022**	**0.001**	0.6996	0.0156
	RF	0.107	0.9	0.9	**0.001**	**0.0022**	1.0	0.0789	**0.001**	0.9
	RotF	**0.001**	0.1426	0.2703	**0.001**	**0.001**	0.0789	1.0	**0.001**	0.0156
	SWRF	0.0916	**0.001**	**0.001**	0.9	0.6996	**0.001**	**0.001**	1.0	**0.001**
	XGB	0.3364	0.9	0.9	**0.001**	0.0156	0.9	0.0156	**0.001**	1.0
	ERT	1.0	0.9	0.9	**0.001**	**0.001**	0.9	0.9	**0.001**	0.1155
	IVRF	0.9	1.0	0.7172	**0.001**	**0.0039**	0.9	0.9	**0.001**	0.3189
	MRF	0.9	0.7172	1.0	**0.001**	**0.001**	0.8398	0.9	**0.001**	**0.0017**
	OWRF	**0.001**	**0.001**	**0.001**	1.0	0.6996	**0.001**	**0.001**	0.9	0.0335
room	ObF	**0.001**	**0.0039**	**0.001**	0.6996	1.0	**0.0017**	**0.001**	0.6996	0.8398
	RF	0.9	0.9	0.8398	**0.001**	**0.0017**	1.0	0.9	**0.001**	0.2126
	RotF	0.9	0.9	0.9	**0.001**	**0.001**	0.9	1.0	**0.001**	**0.0128**
	SWRF	**0.001**	**0.001**	**0.001**	0.9	0.6996	**0.001**	**0.001**	1.0	0.0335
	XGB	0.1155	0.3189	**0.0017**	0.0335	0.8398	0.2126	**0.0128**	0.0335	1.0
	ERT	1.0	0.0916	**0.0068**	0.107	0.9	0.0401	**0.001**	0.0789	0.5245
	IVRF	0.0916	1.0	0.9	**0.001**	**0.0017**	0.9	0.1641	**0.001**	0.9
	MRF	**0.0068**	0.9	1.0	**0.001**	**0.001**	0.9	0.6296	**0.001**	0.7347
	OWRF	0.107	**0.001**	**0.001**	1.0	0.7347	**0.001**	**0.001**	0.9	**0.001**
plates	ObF	0.9	**0.0017**	**0.001**	0.7347	1.0	**0.001**	**0.001**	0.6646	0.0401
	RF	0.0401	0.9	0.9	**0.001**	**0.001**	1.0	0.3024	**0.001**	0.9
	RotF	**0.001**	0.1641	0.6296	**0.001**	**0.001**	0.3024	1.0	**0.001**	**0.0128**
	SWRF	0.0789	**0.001**	**0.001**	0.9	0.6646	**0.001**	**0.001**	1.0	**0.001**
	XGB	0.5245	0.9	0.7347	**0.001**	0.0401	0.9	**0.0128**	**0.001**	1.0
	ERT	1.0	0.3024	**0.0055**	0.067	0.9	0.3364	**0.001**	0.067	0.2126
	IVRF	0.3024	1.0	0.9	**0.001**	**0.0084**	0.9	0.0278	**0.001**	0.9
	MRF	**0.0055**	0.9	1.0	**0.001**	**0.001**	0.8748	0.5945	**0.001**	0.9
	OWRF	0.067	**0.001**	**0.001**	1.0	0.6996	**0.001**	**0.001**	0.9	**0.001**
statlog image	ObF	0.9	**0.0084**	**0.001**	0.6996	1.0	**0.0104**	**0.001**	0.6996	**0.0044**
	RF	0.3364	0.9	0.8748	**0.001**	**0.0104**	1.0	**0.023**	**0.001**	0.9
	RotF	**0.001**	0.0278	0.5945	**0.001**	**0.001**	**0.023**	1.0	**0.001**	0.0477
	SWRF	0.067	**0.001**	**0.001**	0.9	0.6996	**0.001**	**0.001**	1.0	**0.001**
	XGB	0.2126	0.9	0.9	**0.001**	**0.0044**	0.9	0.0477	**0.001**	1.0

Table B.4
Hyperparameter Search Spaces.

Model	Parameter	Search space
RF, ERT, OblF, RotF	n_estimators	$\{10, 20, \dots, 200\}$
	max_depth	$\{\text{None}, \{2, 3, \dots, 10\}\}$
	min_samples_split	$\{2, 3, \dots, 20\} \cup [0.01, 0.5]$
	min_samples_leaf	$\{1, 2, \dots, 20\} \cup [0.01, 0.2]$
	max_features	{sqrt, log2, None}
	criterion (clf)	{gini, entropy}
	criterion (reg)	{squared_error}
Oblique RF (OblF)	min_weight_fraction_leaf	$[0.0, 0.5]$
	max_leaf_nodes	$\{\text{None}, \{10, 20, \dots, 100\}\}$
	min_impurity_decrease	$[0.0, 0.2]$
	max_samples	$\{\text{None}, \{10, 20, \dots, 100\}\} \cup [0.1, 1.0]$
Rotation Forest (RotF)	remove_proportion	$[0.5, 0.9]$
	min_max_group	$\{10, 100\}$
XGBoost	objective (multi clf)	{multi:softmax}
	objective (binary clf)	{reg:squarederror}
	objective (multi)	{reg:squarederror}
	learning_rate	$e^{U(-5,0)}$
	gamma	$U(0, 5)$
	max_depth	$\{1, 2, \dots, 7\}$
	min_child_weight	$U(1, 10)$
	subsample	$U(0.5, 1)$
	colsample_bytree	$U(0.3, 1)$
	lambda	$U(1, 5)$
	alpha	$U(0, 1)$
	max_bin	$\{64, 128, 192, \dots, 768\}$

References

[1] I.H. Sarker, Machine learning: Algorithms, real-world applications and research directions, SN Comput. Sci. 2 (3) (2021) 160.

[2] S. González, S. García, J. Del Ser, L. Rokach, F. Herrera, A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities, Inf. Fusion 64 (2020) 205–237, http://dx.doi.org/10.1016/j.inffus.2020.07.007, URL: https://www.sciencedirect.com/science/article/pii/S1566253520303195.

[3] R. Shwartz-Ziv, A. Armon, Tabular data: Deep learning is not all you need, Inf. Fusion 81 (2022) 84–90, http://dx.doi.org/10.1016/j.inffus.2021.11.011, URL: https://www.sciencedirect.com/science/article/pii/S1566253521002360.

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

[5] J.H. Friedman, Stochastic gradient boosting, Comput. Stat. Data Anal. 38 (4) (2002) 367–378.

[6] D. Devi, S.K. Biswas, B. Purkayastha, A cost-sensitive weighted random forest technique for credit card fraud detection, in: 2019 10th International Conference on Computing, Communication and Networking Technologies, ICCCNT, 2019, pp. 1–6, http://dx.doi.org/10.1109/ICCCNT45670.2019.8944885.

[7] M. Shahhosseini, G. Hu, Improved weighted random forest for classification problems, 2020, ArXiv, arXiv:2009.00534.

[8] H. Markowitz, Portfolio selection, J. Finance 7 (1) (1952) 77–91, URL: http://www.jstor.org/stable/2975974.

[9] P. Geurts, D. Ernst, L. Wehenkel, Extremely randomized trees, Mach. Learn. 63 (2006) 3–42, http://dx.doi.org/10.1007/s10994-006-6226-1.

[10] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 785–794, http://dx.doi.org/10.1145/2939672.2939785.

[11] T.M. Tomita, J. Browne, C. Shen, J. Chung, J.L. Patsolic, B. Falk, C.E. Priebe, J. Yim, R. Burns, M. Maggioni, J.T. Vogelstein, Sparse projection oblique randomer forests, J. Mach. Learn. Res. 21 (104) (2020) 1–39, URL: http://jmlr.org/papers/v21/18-664.html.

[12] A. Li, R. Perry, C. Huynh, T.M. Tomita, R. Mehta, J. Arroyo, J. Patsolic, B. Falk, S. Sarma, J. Vogelstein, Manifold oblique random forests: Towards closing the gap on convolutional deep networks, SIAM J. Math. Data Sci. 5 (1) (2023) 77–96, http://dx.doi.org/10.1137/21M1449117.

[13] J.J.R. Diez, L.I. Kuncheva, C.J. Alonso, Rotation forest: A new classifier ensemble method, IEEE Trans. Pattern Anal. Mach. Intell. 28 (2006) 1619–1630, URL: https://api.semanticscholar.org/CorpusID:6847493.

[14] M. Juez-Gil, Á. Arnaiz-González, J.J. Rodríguez, C. López-Nozal, C. García-Osorio, Rotation forest for big data, Inf. Fusion 74 (2021) 39–49, http://dx.doi.org/10.1016/j.inffus.2021.03.007, URL: https://www.sciencedirect.com/science/article/pii/S1566253521000634.

[15] S. Han, H. Kim, Y.S. Lee, Double random forest, Mach. Learn. 109 (2020) http://dx.doi.org/10.1007/s10994-020-05889-1.

[16] A.J. Dobson, A.G. Barnett, An Introduction to Generalized Linear Models, Chapman and Hall/CRC, 2018.

[17] K.P. Murphy, Probabilistic Machine Learning: An introduction, MIT Press, 2022, URL:.

[18] T. Hofmann, B. Schölkopf, A.J. Smola, Kernel methods in machine learning, Ann. Statist. 36 (3) (2008) 1171–1220, http://dx.doi.org/10.1214/009053607000000677.

[19] L.E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.

[20] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, http://www.deeplearningbook.org.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), in: Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017, URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[22] H.B. Li, W. Wang, H.W. Ding, J. Dong, Trees weighting random forest method for classifying high-dimensional noisy data, in: 2010 IEEE 7th International Conference on E-Business Engineering, 2010, pp. 160–163, http://dx.doi.org/10.1109/ICEBE.2010.99.

[23] S.J. Winham, R.R. Freimuth, J.M. Biernacka, A weighted random forests approach to improve predictive performance, Stat. Anal. Data Min. ASA Data Sci. J. 6 (6) (2013) 496–505, http://dx.doi.org/10.1002/sam.11196, URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11196.

[24] M. El Habib Daho, N. Settouti, M. El Amine Lazouni, M. El Amine Chikh, Weighted vote for trees aggregation in random forest, in: 2014 International Conference on Multimedia Computing and Systems, ICMCS, 2014, pp. 438–443, http://dx.doi.org/10.1109/ICMCS.2014.6911187.

[25] H. Pham, S. Olafsson, On cesáro averages for weighted trees in the random forest, J. Classification 37 (2019) http://dx.doi.org/10.1007/s00357-019-09322-8.

[26] V. Jain, J. Sharma, K. Singhal, A. Phophalia, Exponentially weighted random forest, in: Pattern Recognition and Machine Intelligence, Springer International Publishing, Cham, 2019, pp. 170–178.

[27] K. Gajowniczek, I. Grzegorczyk, T.S. Zabkowski, C.L. Bajaj, Weighted random forests to improve arrhythmia classification, Electronics 9 (2020).

[28] H. Zhang, B. Quost, M.H. Masson, Cautious weighted random forests, Expert Syst. Appl. 213 (2023) 118883, http://dx.doi.org/10.1016/j.eswa.2022.118883, URL: https://www.sciencedirect.com/science/article/pii/S0957417422019017.

[29] T.K. Ho, The random subspace method for constructing decision forests, IEEE Trans. Pattern Anal. Mach. Intell. 20 (8) (1998) 832–844, http://dx.doi.org/10.1109/34.709601.

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

[31] W.F. Sharpe, The sharpe ratio, in: Streetwise–the Best of the Journal of Portfolio Management, Princeton University Press NJ, 1998, pp. 169–185.

[32] G. Cornuéjols, R. Tutuncu, Optimization methods in finance, 2007, http://dx.doi.org/10.1017/CBO9780511753886.

[33] B. Evans, D. Fisher, Overcoming process delays with decision tree induction, IEEE Expert 9 (1) (1994) 60–66, http://dx.doi.org/10.1109/64.295130.

[34] C.O. Sakar, S. Polat, M. Katircioglu, Y. Kastro, Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks, Neural Comput. Appl. 31 (2019) http://dx.doi.org/10.1007/s00521-018-3523-0.

[35] I.C. Yeh, C.h. Lien, The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients, Expert Syst. Appl. 36 (2, Part 1) (2009) 2473–2480, http://dx.doi.org/10.1016/j.eswa.2007.12.020, URL: https://www.sciencedirect.com/science/article/pii/S0957417407006719.

[36] S. Moro, P. Cortez, P. Rita, A data-driven approach to predict the success of bank telemarketing, Decis. Support Syst. 62 (2014) 22–31, http://dx.doi.org/10.1016/j.dss.2014.03.001, URL: https://www.sciencedirect.com/science/article/pii/S016792361400061X.

[37] S.M. Hammer, D.A. Katzenstein, M.D. Hughes, H. Gundacker, R.T. Schooley, R.H. Haubrich, W.K. Henry, M.M. Lederman, J.P. Phair, M. Niu, M.S. Hirsch, T.C. Merigan, A trial comparing nucleoside monotherapy with combination therapy in HIV-infected adults with CD4 cell counts from 200 to 500 per cubic millimeter. AIDS clinical trials group study 175 study team, N. Engl. J. Med. 335 15 (1996) 1081–1090, URL: https://api.semanticscholar.org/CorpusID:40754467.

[38] K. Fernandes, A proactive intelligent decision support system for predicting the popularity of online news, ISBN: 978-3-319-23484-7, 2015, http://dx.doi.org/10.1007/978-3-319-23485-4_53.

[39] K. Hamidieh, A data-driven statistical model for predicting the critical temperature of a superconductor, Comput. Mater. Sci. 154 (2018) 346–354, http://dx.doi.org/10.1016/j.commatsci.2018.07.052, URL: https://www.sciencedirect.com/science/article/pii/S0927025618304877.

[40] P. Putten, M. de Ruiter, M. Someren, CoIL challenge 2000 tasks and results: Predicting and explaining caravan policy ownership, 2000.

[41] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decis. Support Syst. 47 (4) (2009) 547–553, http://dx.doi.org/10.1016/j.dss.2009.05.016, URL: https://www.sciencedirect.com/science/article/pii/S0167923609001377. Smart Business Networks: Concepts and Empirical Evidence.

[42] M. Buscema, S. Terzi, W. Tastle, Steel Plates Faults, UCI Machine Learning Repository, 2010, http://dx.doi.org/10.24432/C5J88N.
[43] UCI Machine Learning Repository, Statlog (Image Segmentation), 1990, http://dx.doi.org/10.24432/C5P01G.
[44] M.V. Martins, D. Tolledo, J. Machado, L. Baptista, V. Realinho, Early prediction of student's performance in higher education: A case study, 2021, http://dx.doi.org/10.1007/978-3-030-72657-7_16.
[45] A.P. Singh, V. Jain, S. Chaudhari, F.A. Kraemer, S. Werner, V. Garg, Machine learning-based occupancy estimation using multivariate sensor nodes, in: 2018 IEEE Globecom Workshops, GC Wkshps, 2018, pp. 1–6, URL: https://api.semanticscholar.org/CorpusID:56336847.
[46] A. Reisizadeh, A. Mokhtari, H. Hassani, R. Pedarsani, An exact quantized decentralized gradient descent algorithm, IEEE Trans. Signal Process. 67 (19) (2019) 4934–4947, http://dx.doi.org/10.1109/TSP.2019.2932876.
[47] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2017, URL: http://archive.ics.uci.edu/ml.
[48] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[49] S.P. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[50] B.H. Menze, B.M. Kelm, D.N. Splitthoff, U. Koethe, F.A. Hamprecht, On oblique random forests, in: D. Gunopulos, T. Hofmann, D. Malerba, M. Vazirgiannis (Eds.), Machine Learning and Knowledge Discovery in Databases, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 453–469.
[51] M. Ganaie, M. Tanveer, P. Suganthan, V. Snasel, Oblique and rotation double random forest, Neural Netw. 153 (2022) 496–517, http://dx.doi.org/10.1016/j.neunet.2022.06.012, URL: https://www.sciencedirect.com/science/article/pii/S0893608022002258.
[52] Q. Shi, R. Katuwal, P. Suganthan, M. Tanveer, Random vector functional link neural network based ensemble deep learning, Pattern Recognit. 117 (2021) 107978, http://dx.doi.org/10.1016/j.patcog.2021.107978, URL: https://www.sciencedirect.com/science/article/pii/S0031320321001655.

Eleftherios Kouloumpris is an M.Sc. graduate from the Department of Informatics of the Aristotle University of Thessaloniki. He is currently pursuing his PhD in the same university, where he specializes in the intriguing intersection of machine learning and finance. He has made significant contributions to the field through the development of various models for portfolio management and algorithmic trading. Additionally, he explores the innovative applications of machine learning models that draw inspiration from financial mathematics.

Professor Ioannis Vlahavas is a faculty member in the Department of Informatics of the Aristotle University of Thessaloniki. He has extensive research experience in artificial intelligence, with numerous publications and citations. Since May 2017, he has been an active partner of EuroAI. Additionally, he actively participates in the design of AI-based products, with prior experience in establishing successful academic spin-off companies.