Ensemble learning for portfolio valuation and risk management

Lotfi Boudabsa & Damir Filipović

Quantitative Finance, 2025
Vol. 25, No. 3, 421–442, https://doi.org/10.1080/14697688.2024.2430299

Ensemble learning for portfolio valuation and risk management

LOTFI BOUDABSA*† and DAMIR FILIPOVIƆ‡

†Department of Mathematics, EPFL, Lausanne, Switzerland
‡Swiss Finance Institute @ EPFL, EPFL and Swiss Finance Institute, Lausanne, VD, Switzerland

(*Received 14 May 2022; accepted 9 November 2024*)

We introduce an ensemble learning method for dynamic portfolio valuation and risk management building on regression trees. We learn the dynamic value process of a derivative portfolio from a finite sample of its cumulative cash flow. The estimator is given in closed form. The method is fast and accurate, and scales well with sample size and path space dimension. It can also be applied to Bermudan options. Numerical experiments show good results for examples in dimensions 12 and 36.

Keywords: Dynamic portfolio valuation; Ensemble learning; Gradient boosting; Random forest; Regression trees; Risk management; Bermudan options;

1. Introduction

The valuation and risk management of derivative portfolios form a challenging task in banks, insurance companies, and other financial institutions. issues are formally explained as follows. Most economic scenario generators can be represented as stochastic models with finitely many time periods $t = 0, 1, \dots, T$, where randomness is generated by some underlying stochastic driver $X = (X_1, \dots, X_T)$, and the financial variables of interest are dynamic functionals of X. The components X_t are mutually independent, but not necessarily identically distributed, taking values in $\mathbb{R}^d$ endowed with the Borel σ-algebra $\mathcal{B}(\mathbb{R}^d)$, for some $d \in \mathbb{N}$. We assume that X is realized on the path space $\mathbb{R}^{d\times T}$ such that $X_t(x) = x_t$ for a generic sample point $x = (x_1, \dots, x_T) \in \mathbb{R}^{d\times T}$. We denote the distribution of X by $\mathbb{Q} = \mathbb{Q}_1 \times \cdots \times \mathbb{Q}_T$, and we assume that $\mathbb{Q}$ represents the risk-neutral pricing measure with respect to some fixed numeraire, such as the money market account. All financial values and cash flows henceforth are discounted by this numeraire, if not otherwise stated. The stochastic driver X generates the filtration $\mathcal{F}_t = \mathcal{B}(\mathbb{R}^d)^{\otimes t}$ which represents the flow of information.[†]

We consider a portfolio whose cumulative cash flow is modeled by some measurable function $f : \mathbb{R}^{d\times T} \to \mathbb{R}$ such that $f \in L^2_{\mathbb{Q}}$. Its dynamic value process V is then given by the martingale

$$V_t = \mathbb{E}_{\mathbb{Q}}[f(X) \mid \mathcal{F}_t], \quad t = 0, \dots, T. \tag{1}$$

Computing V is challenging, because the conditional expectations in (1) are not available in closed form in general.

For illustration, let us consider the multivariate Black–Scholes model, where X_t are i.i.d. standard normal on $\mathbb{R}^d$. There are m nominal stock prices given by

$$S_{i,t} = S_{i,t-1} \exp[\sigma_i^\top X_t \sqrt{\Delta_t} + (r - \|\sigma_i\|^2/2)\Delta_t], \quad t = 1, \dots, T, \tag{2}$$

for some initial values $S_{i,0}$, volatility vectors $\sigma_i \in \mathbb{R}^d$, $i = 1, \dots, m$, constant risk-free rate r, and where Δ_t denotes the time step size in units of a year between $t-1$ and t. Note that while the components of X_t are mutually independent, the stock prices are not. Indeed, the conditional covariances of the log returns are encoded by the volatility vectors and given by

$$\mathrm{Cov}_{\mathbb{Q}}[\log S_{i,t}, \log S_{j,t} \mid \mathcal{F}_{t-1}] = \sigma_i^\top \sigma_j \Delta_t. \tag{3}$$

Even if these covariances were zero, there exists no closed-form expression for the value process V of a structured

*Corresponding author. Email: lotfi.boudabsa@epfl.ch, boudabsalotfi@live.fr
† Henceforth, we let $\mathcal{F}_0$ denote the trivial σ-algebra. However, we could easily extend the setup to include randomness at $t = 0$, by setting $X = (X_0, X_1, \dots, X_T)$. Here X_0 could include cashflow specific values that parametrize the cumulative cashflow function $f(X)$, such as the strike price of an embedded option or the initial values of underlying financial instruments. We could then sample X_0 from a Bayesian prior $\mathbb{Q}_0$.

product such as the barrier reverse convertible option whose discounted payoff at T is

$$f(X) = \mathrm{e}^{-r\sum_{t=1}^{T}\Delta_t}\left(C + F\left(1 - 1_{\{\min_{i,t} S_{i,t} \le B\}}\right.\right.$$
$$\left.\left.\times\left(1 - \min_i \frac{S_{i,T}}{S_{i,0}K}\right)^+\right)\right), \qquad (4)$$

for some strike price K, barrier $B < K$, coupon C, and face value F.

We solve this issue via a novel method to learn the portfolio value process V. First, we use ensemble estimators with regression trees to learn the function f from a finite sample $\boldsymbol{X} = (X^{(1)}, \dots, X^{(n)})$, drawn from $\mathbb{Q}$, along with the corresponding function values $\boldsymbol{f} = (f(X^{(1)}), \dots, f(X^{(n)}))$.[†] Here we consider the two most popular ensemble estimators, namely Random Forest defined in Breiman (2001), and Gradient Boosting defined in Friedman (2001). We denote these estimators of f by $f_{\boldsymbol{X}}$. In either case, the expression of $f_{\boldsymbol{X}}$ is of the form

$$f_{\boldsymbol{X}} = \sum_{i=1}^{N} \beta_i \mathbb{1}_{\boldsymbol{A}_i}, \qquad (5)$$

for some $N \in \mathbb{N}$, where β_i are real coefficients, and $\boldsymbol{A}_i$ are hyperrectangles of $\mathbb{R}^{d\times T}$ that cover $\mathbb{R}^{d\times T}$, but are not necessarily disjoint. See (8) for the definition of a hyperrectangle. Second, we use $f_{\boldsymbol{X}}$ to define the process $V_{\boldsymbol{X}}$ as follows,

$$V_{\boldsymbol{X},t} = \mathbb{E}_{\mathbb{Q}}[f_{\boldsymbol{X}}(X) \mid \mathcal{F}_t], \quad t = 0, \dots, T. \qquad (6)$$

The process $V_{\boldsymbol{X}}$ in (6) is an estimator of the value process V in (1). This estimator is fast to construct for two reasons. First, the training of $f_{\boldsymbol{X}}$ is fast, which is due to the use of readily accessible and highly optimized implementations of Random Forest and Gradient Boosting. Specifically, for Random Forest we use the RandomForestRegressor class in scikit-learn (Pedregosa *et al.* 2011), and for Gradient Boosting we use the XGBRegressor class in XGBoost (eXtreme Gradient Boosting) (Chen and Guestrin 2016). Second, the conditional expectations in (6) are given in closed form, in the sense that they can be efficiently evaluated at very low computational cost, see Section 3 for details. Furthermore, there is empirical evidence showing that $f_{\boldsymbol{X}}$ is an accurate estimator of f, i.e. $f_{\boldsymbol{X}}$ achieves a small $L^2_{\mathbb{Q}}$-error $\|f - f_{\boldsymbol{X}}\|_{2,\mathbb{Q}}$, in many problems arising from different fields (scientific fields, and machine learning and data mining challenges). See, e.g. Biau and Scornet (2016) and references therein for Random Forest, and Chen and Guestrin (2016) for Gradient Boosting. This implies that $V_{\boldsymbol{X}}$ is an accurate estimator of V. Indeed, thanks to Doob's maximal inequality, see, e.g. Revuz and Yor (1994, Corollary II.1.6), the path-wise maximum $L^2_{\mathbb{Q}}$-error is bounded by

$$\| \max_{t=0,\dots,T} |V_t - V_{\boldsymbol{X},t}| \|_{2,\mathbb{Q}} \le 2\|f - f_{\boldsymbol{X}}\|_{2,\mathbb{Q}}. \qquad (7)$$

There are many risk management tasks building on the dynamic value process V. In this paper, we focus on risk measurement as a generic example.[‡] For two dates $t_0 < t_1$, we denote by $\Delta V_{t_0,t_1} = V_{t_1} - V_{t_0}$ the gain from holding the portfolio over the period $[t_0, t_1]$. Portfolio risk managers and financial market regulators alike quantify the risk of the portfolio V over $[t_0, t_1]$ by means of an $\mathcal{F}_{t_0}$-conditional risk measure, such as value at risk or expected shortfall, evaluated at $\Delta V_{t_0,t_1}$.[§] In practice these risk measures are applied under the equivalent real-world measure $\mathbb{P} \sim \mathbb{Q}$. Using the Cauchy–Schwarz inequality and (7), we obtain $\| \max_{t=0,\dots,T} |V_t - V_{\boldsymbol{X},t}| \|_{1,\mathbb{P}} \le \|\frac{d\mathbb{P}}{d\mathbb{Q}}\|_{2,\mathbb{Q}} \| \max_{t=0,\dots,T} |V_t - V_{\boldsymbol{X},t}| \|_{2,\mathbb{Q}} \le 2\|\frac{d\mathbb{P}}{d\mathbb{Q}}\|_{2,\mathbb{Q}} \|f - f_{\boldsymbol{X}}\|_{2,\mathbb{Q}}$, so that $V_{\boldsymbol{X}}$ and V are close in $L^1_{\mathbb{P}}$ as soon as $f_{\boldsymbol{X}}$ and f are close in $L^2_{\mathbb{Q}}$. Hence risk measures that are continuous with respect to the $L^1_{\mathbb{P}}$-norm, such as value at risk (under mild technical conditions) and expected shortfall, see, e.g. Cambou and Filipović (2017, Section 6), return similar values when applied to $V_{\boldsymbol{X}}$ instead of V.

The two main categories of methods for the estimation of V_t in (1), at some fixed time t, are nested Monte Carlo simulations and least squares Monte Carlo methods. Nested Monte Carlo simulations consist of a two-stage simulation procedure. At the outer stage, several scenarios of $(X_1, \dots, X_t)$ are drawn. At the inner stage, for each scenario of $(X_1, \dots, X_t)$, the conditional expectation $\mathbb{E}_{\mathbb{Q}}[f(X) \mid X_1, \dots, X_t]$ is estimated by Monte Carlo simulation. It is known that in practice nested Monte Carlo simulations are computationally very costly, see, e.g. Broadie *et al.* (2015). Nevertheless, the computational burden due to the inner stage is mitigated in Gordy and Juneja (2010). In fact, the authors show that a relatively small number of simulations of $f(X)$ in the inner stage can give accurate estimations of risk measures evaluated at V_t. Least squares Monte Carlo methods estimate V_t by a function $\hat{V}_t$ that can be evaluated at any scenario of $(X_1, \dots, X_t)$. Glasserman and Yu (2004) introduce the distinction between two types of least squares Monte Carlo methods. In the first class of methods, termed 'regress-now', V_t is estimated by means of a projection on a finite number of basis functions that depend solely on the variable of interest $(x_1, \dots, x_t) \in \mathbb{R}^{d\times t}$. In the second class of methods, termed 'regress-later', it is the payoff function f that is estimated by means of a projection on a finite number of *well chosen* basis functions that depend on the state variable $(x_1, \dots, x_T) \in \mathbb{R}^{d\times T}$. By *well chosen* we mean basis functions whose conditional expectations with respect to the random variable $(X_1, \dots, X_t)$ are given in closed form.

In portfolio risk management, Broadie *et al.* (2015) use regress-now for the estimation of the unconditional expectation of a Lipschitz continuous function $g(L)$ of the 1-year loss $L = -\Delta V_{0,1}$. They assume various sets of basis functions, such as polynomial functions, and plain vanilla derivatives. Little was known on the optimal choice of the basis functions in regress-now, until the paper (Ha and Bauer 2022). They introduce a family of basis functions,

† More precisely, $\boldsymbol{X}$ consists of i.i.d. $\mathbb{R}^{d\times T}$-valued random variables $X^{(i)} \sim \mathbb{Q}$ defined on the product probability space $(\boldsymbol{E}, \boldsymbol{\mathcal{E}}, \boldsymbol{Q})$ with $\boldsymbol{E} = \mathbb{R}^{d\times T} \otimes \mathbb{R}^{d\times T} \otimes \cdots$, $\boldsymbol{\mathcal{E}} = \mathcal{B}(\mathbb{R}^d)^{\otimes T} \otimes \mathcal{B}(\mathbb{R}^d)^{\otimes T} \otimes \cdots$, and $\boldsymbol{Q} = \mathbb{Q} \otimes \mathbb{Q} \otimes \cdots$.

‡ Another important task in portfolio risk management is hedging, which is sketched in more detail in Boudabsa and Filipović (2022).

§ For the definition of value at risk and expected shortfall (also called conditional value at risk or average value at risk), we refer to Föllmer and Schied (2004, Section 4.4), and Section 4 below.

called ‘singular functions’, which can be tied to a model framework rather than to a particular payoff function f, that are robust. Non-parametric regression techniques have also been used as a regress-now method for the estimation of risk measures of portfolios, e.g. neural networks in Cheridito *et al.* (2020), kernel smoothing and k-nearest neighbors in Hong *et al.* (2017), stochastic Kriging in Liu and Staum (2010), and Gaussian Process Regression in Risk and Ludkovski (2018).

Regress-later, also known as the replicating portfolio method, is studied, and compared to regress-now, in Pelsser and Schweizer (2016). They show that regress-later presents several advantages over regress-now. Other works related to the replicating portfolio method include Natolski and Werner (2014) and Cambou and Filipović (2018), where they give a mathematical study of replicating portfolios.

As mentioned in Fernandez-Arjona and Filipović (2022, Section 2), in insurance regulation, there are cases where portfolio risk measurement is required not only for the first year loss, but also for the second and third year losses. For these cases, regress-now methods would have to be applied three times to estimate, separately, three losses. However, regress-later solutions presented in Fernandez-Arjona and Filipović (2022) and Boudabsa and Filipović (2022) give closed form estimation of the entire value process V, so that estimation of portfolio loss at any time is readily available. Fernandez-Arjona and Filipović (2022) use orthogonal polynomials and shallow neural networks to estimate V. Boudabsa and Filipović (2022) apply kernel ridge regression to derive a closed-form estimator of V. Their kernel-based estimator satisfies asymptotic consistency and finite sample guarantees. However, due to cubic training time complexity, it cannot be applied to high dimensional problems, i.e. problems where the sample size n or the path space dimension $d \times T$ are very large. As discussed in Section 5.1, the ensemble estimators in the present paper scale better.

Another strand of the literature deals with the pricing of Bermudan options. Least squares Monte Carlo methods were first applied by Carriere (1996), Tsitsiklis and van Roy (1999), and Longstaff and Schwartz (2001). Becker *et al.* (2019) use neural networks to learn the optimal stopping rule. Goudenège *et al.* (2020) introduce the GPR-EI (Gaussian Process Regression-Exact Integration) method, which is a regress-later solution. The GPR-EI method gives a closed-form estimator of the entire value process V of an American option under the Black–Scholes and Rough–Bergomi models. Ech-Chafiq *et al.* (2021) present a regress-now solution, where they use regression trees and Random Forest to estimate the continuation values of Bermudan options. While our focus is on the estimation of the portfolio value process (1), we illustrate in Appendix 2 how our method can be applied to price Bermudan options, both with regress-later and regress-now ensemble estimators.

Here and throughout we use the following conventions and notation. For any $p \in [1,\infty)$ and measurable function $f : \mathbb{R}^{d\times T} \to \mathbb{R}$, we denote $\|f\|_{p,\mathbb{Q}} = (\int_{\mathbb{R}^{d\times T}} |f(x)|^p \mathbb{Q}(dx))^{1/p}$. We denote by $L^p_{\mathbb{Q}}$ the space of $\mathbb{Q}$-*equivalence classes* of measurable functions $f : \mathbb{R}^{d\times T} \to \mathbb{R}$ with $\|f\|_{p,\mathbb{Q}} < \infty$. If not otherwise stated, we will use the same symbol, e.g. f, for a function and its equivalence class. Let $a = (a_1, \dots, a_T)$ and $b = (b_1, \dots, b_T)$, where $a_t, b_t \in \overline{\mathbb{R}^d}$, so that $a, b \in \overline{\mathbb{R}^{d\times T}}$.[†] Assume that $a_t < b_t$, i.e. $a_{j,t} < b_{j,t}$ for every $j = 1, \dots, d$, for every $t = 1, \dots, T$. A hyperrectangle $\boldsymbol{A} = (a, b]$ of $\mathbb{R}^{d\times T}$ is a subset of $\mathbb{R}^{d\times T}$ of the form

$$\boldsymbol{A} = \{(x_1, \dots, x_T) \in \mathbb{R}^{d\times T} \mid a_t < x_t \le b_t,\ t = 1, \dots, T\}. \quad (8)$$

It is convenient to write $\boldsymbol{A} = \prod_{t=1}^T (a_t, b_t]$, where $(a_t, b_t] = \prod_{j=1}^d (a_{j,t}, b_{j,t}]$ is a subset of $\mathbb{R}^d$.[‡]

The remainder of the paper is as follows. Section 2 presents the ensemble estimators we use to learn the function f. Section 3 shows that the value process estimator V_X is in closed form for a large family of stochastic drivers. Section 4 provides numerical examples for the valuation of exotic and path-dependent options in the multivariate Black–Scholes model. Section 5 discusses future research directions. Section 6 concludes. Appendix 1 compares our method to its regress-now variant. Appendix 2 shows how our method can be applied to Bermudan options.

2. Ensemble estimators based on regression trees

Following up on Section 1, we let $f \in L^2_{\mathbb{Q}}$. We now present the construction of the estimator f_X in (5), which is used to define the value process estimator V_X in (6). As mentioned above, f_X is either a Random Forest or a Gradient Boosting. Random Forest is defined in Breiman (2001) using CART regression trees (CART stands for Classification And Regression Trees). The CART method is defined in Breiman *et al.* (1984). Gradient Boosting is defined in Friedman (2001) using ‘a small regression tree, such as those produced by CART’. In order to make the paper self-contained, we first recap the CART method.[§] We then recap the construction of a Random Forest and Gradient Boosting. Throughout we assume as given a finite i.i.d. sample $\boldsymbol{X} = (X^{(1)}, \dots, X^{(n)})$ drawn from $\mathbb{Q}$, along with the function values $\boldsymbol{f} = (f(X^{(1)}), \dots, f(X^{(n)}))$. We denote the corresponding empirical distribution on the path space $\mathbb{R}^{d\times T}$ by

$$\mathbb{Q}_X = \frac{1}{n}\sum_{i=1}^n \delta_{X^{(i)}}. \quad (9)$$

2.1. *CART regression tree*

The CART method gives a piece-wise constant function that has the following form,

$$f_{X,\Pi} = \sum_{\boldsymbol{A}\in\Pi} \mathbb{E}_{\mathbb{Q}_X}[f(X) \mid X \in \boldsymbol{A}]\mathbb{1}_{\boldsymbol{A}}, \quad (10)$$

[†] We denote $\overline{\mathbb{R}^d} = \Pi_{j=1}^d [-\infty, \infty]$.

[‡] If $b_{j,t} = \infty$, then $(a_{j,t}, b_{j,t}]$ is defined as $(a_{j,t}, \infty)$.

[§] According to the survey (Loh 2014), the first regression tree method is called AID (Automatic Interaction Detector) and was defined in Morgan and Sonquist (1963). However, it is Breiman *et al.* (1984) that has been the most influential in what we now call the regression tree literature. In this literature there are many different methods that produce regression trees. To the best of our knowledge, today the two most popular regression tree methods are CART defined in Breiman *et al.* (1984), and C4.5 defined in Quinlan (1993).

where Π is a finite hyperrectangle partition of $\mathbb{R}^{d\times T}$.† The conditional expectations in (10) are given by $\mathbb{E}_{\mathbb{Q}_X}[f(X) \mid X \in \boldsymbol{A}] = \mathbb{E}_{\mathbb{Q}_X}[f(X)\mathbb{1}_{\boldsymbol{A}}(X)]/\mathbb{Q}_X[\boldsymbol{A}]$ if $\mathbb{Q}_X[\boldsymbol{A}] > 0$, and we use the convention $\mathbb{E}_{\mathbb{Q}_X}[f(X) \mid X \in \boldsymbol{A}] = 0$ if $\mathbb{Q}_X[\boldsymbol{A}] = 0$ or, equivalently, if no sample point $X^{(i)}$ lies in $\boldsymbol{A}$. The partition Π is constructed recursively by refining the trivial partition $\Pi_0 = \{\mathbb{R}^{d\times T}\}$ in the following way. At step $t \geq 0$, the partition Π_t is of size $t+1$. Pick $\boldsymbol{A} \in \Pi_t$ and perform an axis-aligned split, denoted by (j, s, z), to obtain the two hyperrectangles $\boldsymbol{A}_L = \{x \mid x \in \boldsymbol{A},\ x_{j,s} \leq z\}$ and $\boldsymbol{A}_R = \{x \mid x \in \boldsymbol{A},\ x_{j,s} > z\}$. This gives a new partition $\Pi_{t+1} = (\Pi_t \setminus \boldsymbol{A}) \cup \{\boldsymbol{A}_L, \boldsymbol{A}_R\}$ of size $t+2$. To find the optimal split (j, s, z) of $\boldsymbol{A}$, one minimizes the within-group variance

$$\begin{aligned}(j,s,z) &\mapsto V_{\boldsymbol{A}}(j,s,z)\\ &= \mathbb{E}_{\mathbb{Q}_X}[\mathbb{1}_{\boldsymbol{A}_L}(X)(f(X) - \mathbb{E}_{\mathbb{Q}_X}[f(X) \mid X \in \boldsymbol{A}_L])^2]\\ &\quad + \mathbb{E}_{\mathbb{Q}_X}[\mathbb{1}_{\boldsymbol{A}_R}(X)(f(X) - \mathbb{E}_{\mathbb{Q}_X}[f(X) \mid X \in \boldsymbol{A}_R])^2] \qquad (11)\end{aligned}$$

over $\mathcal{S} = \{(j,s,z) \mid j = 1,\dots,d,\ s = 1,\dots,T,\ z \in \mathbb{R}\}$.

When to stop refining the partition Π_t?‡ In practice, standard stopping rules include not to split a hyperrectangle $\boldsymbol{A}$ if it contains less than a certain number of points **nodesize** $\in \mathbb{N}$. Another rule is to stop refining Π_t when its size reaches a certain fixed number in $\mathbb{N}$. Then, Π_t should be pruned by looking for an optimal sub-partition $\widetilde{\Pi} \subseteq \Pi_t$, so that $f_{X,\Pi}$ in (10) is defined with $\Pi = \widetilde{\Pi}$, see Breiman *et al.* (1984) and Genuer and Poggi (2017). However we omit this step and define $f_{X,\Pi}$ with $\Pi = \Pi_t$. In fact, as discussed in Breiman (2001, Section 4) and Friedman *et al.* (2000, Section 8), a CART regression tree should not be pruned when it is used to define a Random Forest or Gradient Boosting.

The CART regression tree is known for its interpretability, and ability to perform dimensionality reduction and handle outliers. However this estimator is very sensitive to the sample $\boldsymbol{X}$: small perturbations of the sample $\boldsymbol{X}$ can lead to large changes in $f_{X,\Pi}$. In response to this issue, Breiman (1996) introduced Bagging (from bootstrap and aggregating). Bagging is the aggregation, i.e. the average, of M CART regression trees. The m-th tree is constructed using a sample $\boldsymbol{X}_m = (X_m^{(1)}, \dots, X_m^{(n)})$, obtained by bootstrapping from $\boldsymbol{X}$, and the corresponding function values $\boldsymbol{f}_m = (f(X_m^{(1)}), \dots, f(X_m^{(n)}))$. Bagging gives significantly better results than a single CART regression tree. Five years later, Breiman (2001) introduced Random Forest, which is an enhancement of Bagging. This will be our first ensemble estimator.

2.2. *Random forest*

Breiman (2001) defines a class of estimators called Random Forest. The same paper gives an example of Random Forest termed Random Forest-RI, where RI stands for Random Inputs. As highlighted in the survey (Genuer and Poggi 2017), nowadays the name Random Forest very often refers to Random Forest-RI. Therefore, we will call Random Forest-RI simply Random Forest. This estimator is the aggregation of M regression trees, which are grown slightly differently than the CART regression trees. Below we detail its construction.

Fix $\widetilde{n} \leq n$ and let $(\boldsymbol{X}_1, \dots, \boldsymbol{X}_M)$ be M samples, where each sample $\boldsymbol{X}_m = (X_m^{(1)}, \dots, X_m^{(\widetilde{n})})$ is constructed by resampling $\widetilde{n}$ points from $\boldsymbol{X}$. The resampling can be with or without replacement. The resampling is called bootstrapping when it is done with replacement and $\widetilde{n} = n$, otherwise it is called subsampling (with or without replacement). Fix $p \in \{1, \dots, d \times T\}$, and grow M regression trees, where the m-th tree f_{X_m,Π_m} is constructed as follows. Instead of $\boldsymbol{X}$ and $\boldsymbol{f}$, use the sample $\boldsymbol{X}_m$ and the corresponding function values $\boldsymbol{f}_m = (f(X_m^{(1)}), \dots, f(X_m^{(\widetilde{n})}))$. And for every hyperrectangle $\boldsymbol{A}$ to split, draw uniformly p coordinates $(j_1, s_1), \dots, (j_p, s_p)$ from $\{1, \dots, d\} \times \{1, \dots, T\}$, and minimize $(j,s,z) \mapsto V_{\boldsymbol{A}}(j,s,z)$ in (11) over $\{(j_i, s_i, z) \mid i = 1, \dots, p,\ z \in \mathbb{R}\}$ instead of $\mathcal{S}$. Then Random Forest, denoted by $f_{X,\boldsymbol{\Pi}}$ with $\boldsymbol{\Pi} = (\Pi_1, \dots, \Pi_M)$, is the aggregation of the M regression trees f_{X_m,Π_m},

$$f_{X,\boldsymbol{\Pi}} = \frac{1}{M}\sum_{m=1}^{M} f_{X_m,\Pi_m}. \qquad (12)$$

In the case where $p = d \times T$ and the sampling scheme is bootstrapping, Random Forest is just Bagging.

2.3. *Gradient boosting*

The idea of Boosting goes back to a theoretical question posed in Kearns (1988), called the 'Hypothesis Boosting Problem'. In the context of binary classification problems, the author asked whether there exist a process able to turn a weak learner into a strong one. Such a process would be called Boosting. Here a weak learner is a classifier that performs only slightly better than random guessing. And a strong learner is a classifier that achieves a nearly perfect classification. A positive answer to this question was given in Schapire (1990). However the algorithm in Schapire (1990) could not be implemented in practice, and it is AdaBoost, the algorithm defined in Freund and Schapire (1996), that is usually considered as the first workable Boosting algorithm. The success of AdaBoost with classification trees was such that Breiman called it 'best off-the-shelf classifier in the world', see Friedman *et al.* (2000). In order to better understand the performance of AdaBoost, a lot of research has been done. This includes the statistical framework developed in Friedman *et al.* (2000), which was further developed in Friedman (2001) to cover both classification and regression problems. In the later paper, Gradient Boosting is defined. This estimator is based on CART regression trees, and is constructed recursively, by setting $f_{X,0} = \mathbb{E}_{\mathbb{Q}_X}[f(X)]$, and for $t \geq 1$,

$$\begin{cases} f_{X,t}(x) & = f_{X,t-1}(x) - \gamma_t g_t(x), \quad x \in \mathbb{R}^{d\times T},\\ \gamma_t & \in \arg\min_{\gamma \in \mathbb{R}_+} \mathbb{E}_{\mathbb{Q}_X}\left[\psi(f(X), f_{X,t-1}(X) - \gamma g_t(X))\right], \end{cases} \qquad (13)$$

where $\psi : \mathbb{R}^2 \mapsto \mathbb{R}$, $(x,y) \mapsto \psi(x,y)$ is a given loss function. The function g_t is a CART regression tree that estimates the

† Recall that X stands for the canonical process $X_t(x) = x_t$, for $x = (x_1, \dots, x_T)$ in $\mathbb{R}^{d\times T}$.

‡ Other regression tree methods construct the partition Π_t relying on other minimization criteria than the within-group variance in (11). For instance, C4.5 (Quinlan 1993) relies on the gain ratio.

function $x \mapsto \partial_y \psi(f(x), f_{X,t-1}(x))$ using the sample $\boldsymbol{X}$, along with the function values $(\partial_y \psi(f(X^{(1)}), f_{X,t-1}(X^{(1)})), \ldots, \partial_y \psi(f(X^{(n)}), f_{X,t-1}(X^{(n)})))$.[†] And γ_t is called the optimal step-size.

Often in regression problems, one picks the squared error loss function $\psi(x,y) = \frac{1}{2}(x-y)^2$, which is what we do in Section 4. In this case, $x \mapsto \partial_y \psi(f(x), f_{X,t-1}(x)) = f_{X,t-1}(x) - f(x)$. Thus at step $t \geq 1$ of Gradient Boosting, a CART regression tree is used to estimate the residual function $f_{X,t-1} - f$. In practice other loss functions ψ could be considered. The only requirement is that ψ be differentiable with respect to its second variable, see Friedman (2001).[‡]

When to stop increasing the number of boosting iterations t? A standard approach to find the optimal t is to use early stopping techniques, which is what we do in Section 4. However, in practice Gradient Boosting is known to be resistant to overfitting, see, e.g. Bartlett *et al.* (1998). This means that, in general, the $L^2_{\mathbb{Q}}$-error $\|f_{X,t} - f\|_{2,\mathbb{Q}}$ does not increase as t becomes very large.

Henceforth and throughout, f_X is a placeholder for either the Random Forest $f_{X,\Pi}$ in (12), or the Gradient Boosting $f_{X,t}$ in (13). The generic expression of f_X is given in (5).

3. Closed-form estimators for *V*

In the previous section we presented the construction of an ensemble estimator f_X of f of the form (5). Now we use f_X to define an estimator V_X of V of the form (6). From (5) and (6), we derive the following expression

$$V_{X,t} = \sum_{i=1}^{N} \beta_i \mathbb{E}_{\mathbb{Q}}[\mathbb{1}_{A_i}(X) \mid \mathcal{F}_t], \quad t = 0, \ldots, T. \tag{14}$$

Now recall that X has distribution $\mathbb{Q}(dx) = \mathbb{Q}_1(dx_1) \times \cdots \times \mathbb{Q}_T(dx_T)$. Thus for a hyperrectangle $\boldsymbol{A} = \prod_{t=1}^{T}(a_t, b_t]$ in (8), we have[§]

$$\mathbb{E}_{\mathbb{Q}}[\mathbb{1}_{\boldsymbol{A}}(X) \mid \mathcal{F}_t] = \prod_{s=1}^{t} \mathbb{1}_{(a_s,b_s]}(X_s) \prod_{s=t+1}^{T} \mathbb{Q}_s[(a_s, b_s]]. \tag{15}$$

Accordingly, we deduce that the value process estimator V_X is in closed form as soon as the probability

$$\mathbb{Q}_s[(a_s, b_s]] \text{ is in closed form for every } a_s < b_s \in \overline{\mathbb{R}^d}, \quad s = 1, \ldots, T. \tag{16}$$

We say an expression is in closed form if it can be efficiently evaluated at very low computational cost. Below, we present two common cases of stochastic drivers X where property (16) is satisfied.

3.1. *Cross-sectional independence*

The first case is when $\mathbb{Q}_s$ can be factorized as $\mathbb{Q}_s(dx_s) = \mathbb{Q}_{1,s}(dx_{1,s}) \times \cdots \times \mathbb{Q}_{d,s}(dx_{d,s})$. This means that the stochastic driver $X = (X_1, \ldots, X_T)$ is such that X_s has cross-sectional independence, for every $s = 1, \ldots, T$. In this case, for $a_s < b_s \in \overline{\mathbb{R}^d}$, we have $\mathbb{Q}_s[(a_s, b_s]] = \mathbb{Q}_s[\prod_{j=1}^{d}(a_{j,s}, b_{j,s}]] = \prod_{j=1}^{d}(F_{j,s}(b_{j,s}) - F_{j,s}(a_{j,s}))$, where $F_{j,s}$ denotes the cumulative distribution function of $\mathbb{Q}_{j,s}$. Thus property (16) holds as soon as $F_{j,s}$ is in closed form for every $j = 1, \ldots, d$. There are many such examples. For its extensive use in financial modeling we mention the standard normal distribution $\mathbb{Q}_s = \mathcal{N}(0, I_d)$. Examples include the discrete-time multivariate Black–Scholes model in (2) and many more time-series models, such as the multivariate GARCH models, see Bollerslev (1986) and Bauwens *et al.* (2006).

3.2. *Closed-form copulas*

The second case, generalizing the above, is when $\mathbb{Q}_s$ is defined in terms of a copula. In this case, the stochastic driver $X = (X_1, \ldots, X_T)$ is such that X_s has cross-sectional dependence, for every $s = 1, \ldots, T$. As above, we denote by $F_{j,s}$ the cumulative distribution function of the marginal $\mathbb{Q}_{j,s}$. We now assume as given a copula C_s on $[0,1]^d$ such that

$$\mathbb{Q}_s[(-\infty, x_s]] = C_s(F_{1,s}(x_{1,s}), \ldots, F_{d,s}(x_{d,s})).$$

In fact, it is well known that any multivariate distribution on $\mathbb{R}^d$ can be expressed in terms of a copula, see Sklar (1959) and Embrechts (2009, Theorem 1). Moreover, the copula is unique if the marginals $F_{j,s}$ are continuous.

Now property (16) holds as soon as the copula C_s and the marginals $F_{j,s}$ are in closed form. Indeed, for any hyperrectangle $(a,b] = \prod_{j=1}^{d}(a_j, b_j]$ of $\mathbb{R}^d$, we have

$$\mathbb{Q}_s[(a,b]] = \sum_{z \in \prod_{j=1}^{d}\{a_j,b_j\}} (-1)^{N(z)} C_s(F_{1,s}(z_1), \ldots, F_{d,s}(z_d)), \quad N(z) = \mathrm{card}(\{k \mid z_k = a_k\}).$$

The first case corresponds to the independence copula $C_s(y) = \prod_{j=1}^{d} y_j$. Copula models are widespread in financial risk management, as they allow to design tailor-made dependence structures between the underlying assets. See, e.g. McNeil *et al.* (2015) for a thorough discussion.

4. Numerical experiments

We follow up on the introductory example with the multivariate Black–Scholes model with m nominal stock price processes $S_{i,t}$ given by (2). In particular, we assume that X_t are i.i.d. standard normal on $\mathbb{R}^d$.

† For the sake of brevity, we write $\partial_y \psi(f(x), f_{X,t-1}(x))$ instead of $\frac{\partial \psi(\cdot,\cdot)}{\partial y}\big|_{(f(x), f_{X,t-1}(x))}$.

‡ There are several popular open-source software libraries that provide implementations of Gradient Boosting (Friedman 2001). The most popular ones are XGBoost (for eXtreme Gradient Boosting) (Chen and Guestrin 2016), LightGBM (for Light Gradient Boosting Machine) (Ke *et al.* 2017), and CatBoost (for Categorical Boosting) (Prokhorenkova *et al.* 2018). Since these libraries are based on several engineering optimizations, they provide estimators that are not exactly as $f_{X,t}$ in (13). In these libraries, there is a wide range of loss functions available, and it is also possible to implement one's own loss function.

§ For $t = 0$, we set $\prod_{s=1}^{0} \cdot = 1$.

As for the portfolios, we fix a strike price K and consider the following European style exotic options with payoff functions

- Min-put $f(X) = e^{-r\sum_{t=1}^{T}\Delta_t}(K - \min_i S_{i,T})^+$;
- Max-call $f(X) = e^{-r\sum_{t=1}^{T}\Delta_t}(\max_i S_{i,T} - K)^+$.

We also consider a genuinely path-dependent product with the payoff function

- Barrier reverse convertible (BRC) $f(X) = e^{-r\sum_{t=1}^{T}\Delta_t}(C + F(1 - 1_{\{\min_{i,t} S_{i,t} \le B\}}(1 - \min_i \frac{S_{i,T}}{S_{i,0}K})^+))$,

for some barrier $B < K$, coupon C, and face value F. At maturity T, the holder of this structured product receives the coupon C. She also receives the face value F if none of the nominal stock prices falls below the barrier B at any time $t = 1, \ldots, T$. Otherwise, the face value F is reduced by the payoff of F/K min-puts on the normalized stocks $S_{i,T}/S_{i,0}$ with strike price K. These payoff functions are inspired from those given in Becker *et al.* (2019). Note that the payoff functions of the min-put and BRC are bounded, while the payoff of the max-call is unbounded.†

For our numerical experiments we choose the following parameter values: risk-free rate $r = 0$, number of stocks $m = d$, initial stock prices $S_{i,0} = 1$, volatility vectors $\sigma_i = 0.2\boldsymbol{e}_i$, where $\boldsymbol{e}_i$ denote the standard basis vectors in $\mathbb{R}^d$, so that stock prices are independent.‡ Further, we choose strike price $K = 1$ (at the money), barrier $B = 0.6$, coupon $C = 0$, and face value $F = 1$. For the min-put and max-call, $(d, T) = (6, 2)$ and $(\Delta_1, \Delta_2) = (1/12, 11/12)$; for the BRC, $(d, T) = (3, 12)$ and $(\Delta_1, \ldots, \Delta_{12}) = (1/12, \ldots, 1/12)$. Thus the path space $\mathbb{R}^{d\times T}$ is of dimension 12 for the min-put and max-call, and it is of dimension 36 for the BRC.§ Note that for the BRC example, the numbers (d, T) can be considered realistic, see the listing notice of the barrier reverse convertible product issued by the Banque cantonale vaudoise (BCV 2023).

Under the parameter specification above, we generate a training sample $\boldsymbol{X}$ of size $n = 20\,000$. We use $\boldsymbol{X}$, along with the corresponding function values $\boldsymbol{f}$, to construct the ensemble estimator f_X in (5). To find the optimal hyperparameter value for this estimator we use a validation sample $\boldsymbol{X}_{\text{valid}}$ of size $0.4 \times n = 8000$, along with its corresponding function values $\boldsymbol{f}_{\text{valid}}$. Both the optimal hyperparameter value search and the construction of f_X are done using the programming language Python and readily accessible machine learning libraries.

Specifically, when f_X is the Random Forest $f_{X,\Pi}$ in (12), we use the RandomForestRegressor class of the library scikit-learn (Pedregosa *et al.* 2011). We find the optimal hyperparameter value by validation on the set of hyperparameter values $\mathcal{P}_{\text{RF}} = \{(M, \textbf{nodesize}, p) \mid M \in \{100, 250, 500\}, \textbf{nodesize} \in \{2, 3, 5\}, p \in \{\lceil d \times T/3\rceil, d \times T\}\}$ using $\boldsymbol{X}_{\text{valid}}$ and $\boldsymbol{f}_{\text{valid}}$. In $\mathcal{P}_{\text{RF}}$ there are three default hyperparameter values. The RandomForestRegressor (Python) default hyperparameter value $(100, 2, d \times T)$, the randomForest (Liaw and Wiener 2002) (R programming language) default hyperparameter value in regression $(500, 5, \lceil d \times T/3\rceil)$, and our default hyperparameter value $(100, 5, d \times T)$.¶ Table 1 shows the normalized $L^2_{\mathbb{Q}}$-error $\|f_X - f\|_{2,\mathbb{Q}}/V_0$, computed using the validation sample $\boldsymbol{X}_{\text{valid}}$, and the number of hyperrectangles N in the Random Forest f_X in (5) for these three default hyperparameter values as well as the optimal hyperparameter value in $\mathcal{P}_{\text{RF}}$. For the min-put and max-call, we observe that our default hyperparameter value gives normalized $L^2_{\mathbb{Q}}$-error comparable to that given by the optimal hyperparameter value. Besides it has the advantage to give, on average, 8 times less hyperrectangles than the optimal hyperparameter value. This implies that the evaluation of V_X is 8 times faster with our default hyperparameter value than with the optimal hyperparameter value for the min-put and max-call examples. Thus for computational reason we use our default hyperparameter value $(100, 5, 12)$ for min-put and max-call. However for BRC we use the optimal hyperparameter value $(500, 5, 12)$, because here the number of hyperrectangles is relatively small ($N < 500\,000$). For the three payoff functions we use **samplingregime** =bootstrapping. In the class RandomForestRegressor, the variables M, **nodesize**, p, **sampling regime** correspond to **n_estimators**, **min_samples_split**, **max_features**, **bootstrap**, respectively.

When f_X is the Gradient Boosting $f_{X,t}$ in (13), we use the XGBRegressor class of XGBoost (Chen and Guestrin 2016). Similarly to what we did for Random Forest, we use $\boldsymbol{X}_{\text{valid}}$ and $\boldsymbol{f}_{\text{valid}}$ to perform a validation on the set of hyperparameter values $\mathcal{P}_{\text{XGB}} = \{(t_{\text{optimal\_stopping}}(\textbf{nodesize}, \textbf{maxdepth}), \textbf{nodesize}, \textbf{maxdepth}) \mid \textbf{nodesize} \in \{5, 15, 25, 35, 45\}, \textbf{maxdepth} \in \{40, 50, \ldots, 90\}\}$ to find the optimal hyperparameter value. The hyperparameter **max depth** controls the number of hyperrectangles in the regression tree g_t in (13). Given the values **nodesize**, **max depth**, the number of iterations $t_{\text{optimal\_stopping}}(\textbf{nodesize}, \textbf{maxdepth})$ is determined by early stopping using the validation sample $\boldsymbol{X}_{\text{valid}}$. Table 2 shows the normalized $L^2_{\mathbb{Q}}$-error $\|f_X - f\|_{2,\mathbb{Q}}/V_0$, computed using the validation sample $\boldsymbol{X}_{\text{valid}}$, and the number of hyperrectangles N in the Gradient Boosting f_X in (5) for the optimal hyperparameter value in $\mathcal{P}_{\text{XGB}}$. Furthermore, for these three payoff functions we also considered other hyperparameters in XGBRegressor for which we took standard values: **booster** = gbtree, **learning_rate** = 0.1, **tree_method** = hist, **objective** = reg:squarederror, and **base_score** = 0.5. Note that in XGBRegressor, the variables **nodesize** and **max depth** correspond to **min_child_weight** and **max_depth**, respectively.

† A portfolio of derivatives is the sum of several derivatives. Therefore, we can assume that the discounted payoff at T of a portfolio of derivatives is given by $f(X) = \sum_{i=1}^{m} f_i(X)$, where $f_i(X)$ denotes the discounted payoff of the ith derivative at its maturity T_i. The random variable $X \in \mathbb{R}^{d\times T}$ is the underlying stochastic driver, where $T = \max\{T_1, \ldots, T_m\}$. Thus we could apply our method either directly to f, or to each f_i individually.

‡ This choice of model parameters is similar to what is used in other papers, e.g. Becker *et al.* (2019). A more practical setup would assume correlated stocks. However, this can be easily encoded by choosing different volatility vectors, as can be seen from (3). Our numerical experiments are for illustrative purpose only.

§ We could easily scale the number of stocks m, as long as we keep d fixed. This would correspond to a factor model, where a large number m of asset returns is driven by a small number d of factors.

¶ Our default hyperparameter value is an intermediary choice between the default hyperparameter values in RandomForestRegressor and randomForest.

Table 1. Random Forest validation step: normalized $L^2_{\mathbb{Q}}$-error $\|f_X - f\|_{2,\mathbb{Q}}/V_0$, computed using the validation sample $\boldsymbol{X}_{\text{valid}}$ and expressed in %, and number of hyperrectangles N in the Random Forest f_X in (5), for the optimal hyperparameter value in $\mathcal{P}_{\text{RF}}$, and three default hyperparameter values, for the payoff functions min-put, BRC, and max-call.

	Min-put	BRC	Max-call
Optimal hyperparameter value	(500, 2, 12)	(500, 5, 12)	(250, 3, 12)
Normalized $L^2_{\mathbb{Q}}$-error	6.864%	6.884%	10.26%
Number of hyperrectangles	6 279 290	187 710	2 027 347
Default hyperparameter value in RandomForestRegressor (Python)	(100, 2, 12)	(100, 2, 36)	(100, 2, 12)
Normalized $L^2_{\mathbb{Q}}$-error	6.894%	6.973%	10.36%
Number of hyperrectangles	1 255 344	52 805	1 237 737
Default hyperparameter value in randomForest (R)	(500, 5, 4)	(500, 5, 12)	(500, 5, 4)
Normalized $L^2_{\mathbb{Q}}$-error	8.124%	6.884%	12.61%
Number of hyperrectangles	2 575 215	187 710	2 556 448
Our default hyperparameter value	(100, 5, 12)	(100, 5, 36)	(100, 5, 12)
Normalized $L^2_{\mathbb{Q}}$-error	6.917%	6.965%	10.39%
Number of hyperrectangles	494 118	34 807	489 747

Table 2. XGBoost validation step: normalized $L^2_{\mathbb{Q}}$-error $\|f_X - f\|_{2,\mathbb{Q}}/V_0$, computed using the validation sample $\boldsymbol{X}_{\text{valid}}$ and expressed in %, and number of hyperrectangles N in the Gradient Boosting f_X in (5), for the optimal hyperparameter value in $\mathcal{P}_{\text{XGB}}$, for the payoff functions min-put, BRC, and max-call.

	Min-put	BRC	Max-call
Optimal hyperparameter value	(120, 40, 15)	(256, 50, 15)	(152, 60, 35)
Normalized $L^2_{\mathbb{Q}}$-error	5.856%	6.284%	9.753%
Number of hyperrectangles	88 127	189 864	66 225

Table 3. Normalized $L^2_{\mathbb{Q}}$-error $\|V_t - V_{X,t}\|_{2,\mathbb{Q}}/V_0$, computed using the test sample and expressed in %, at steps $t = 0, 1, T$, using XGBoost and Random Forest, for the payoff functions min-put, BRC, and max-call.

Payoff	Estimator	$V_{X,0}$	$V_{X,1}$	$V_{X,T}$
Min-put	XGBoost	**0.1701%**	**1.525%**	**5.814%**
	Random Forest	0.2933%	2.300%	6.811%
	Kernel-based method	0.1942%	1.827%	10.05%
BRC	XGBoost	0.05519%	0.3530%	6.276%
	Random Forest	0.2008%	0.5276%	6.660%
	Kernel-based method	**0.02198%**	**0.2506%**	**5.745%**
Max-call	XGBoost	**0.08016%**	**2.217%**	9.923%
	Random Forest	0.3845%	3.155%	**9.868%**
	Kernel-based method	0.1031%	2.315%	11.65%

Next we use our ensemble estimator f_X to construct V_X in (6). As discussed in Section 3, V_X is given in closed form. We then evaluate $V_{X,t}$ at times $t \in \{0, 1, T\}$ on a test sample $\boldsymbol{X}_{\text{test}}$ of size $n_{\text{test}} = 100\,000$. We benchmark V_X to the ground truth value process V, which we obtain by means of Monte Carlo schemes using $\boldsymbol{X}_{\text{test}}$. More specifically, we obtain V_0 as simple Monte Carlo estimate of $\{f(X) \mid X \in \boldsymbol{X}_{\text{test}}\}$. For V_1, we use a nested Monte Carlo scheme, where we estimate $V_1(X_1)$ using $n_{\text{inner}} = 1000$ inner simulations of $(X_2, \dots, X_T)$, for each X_1 in $\boldsymbol{X}_{\text{test}}$. Then we carry out the following three evaluation tasks.

First, we compute the absolute relative error of $V_{X,0}$, $|V_0 - V_{X,0}|/V_0$, and the normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,t}$, $\|V_t - V_{X,t}\|_{2,\mathbb{Q}}/V_0$, for $t = 1,\ T$. Table 3 shows that normalized $L^2_{\mathbb{Q}}$-error of $V_{X,t}$ decreases substantially for increasing time-to-maturity $T - t$. More specifically, for XGBoost, the normalized $L^2_{\mathbb{Q}}$-error of $V_{X,1}$ is on average 9-times smaller than that of $V_{X,T}$, and the relative absolute error of $V_{X,0}$ is on average 14-times smaller than the normalized $L^2_{\mathbb{Q}}$-error of $V_{X,1}$. For Random Forest these values are 6 and 6, respectively. These findings are in line with (7), which has useful practical implications. Indeed, despite the lack of theoretical bounds on the error $\|V_t - V_{X,t}\|_{2,\mathbb{Q}}$, in concrete applications one can always estimate the normalized $L^2_{\mathbb{Q}}$-error of $V_{X,T}$ by a simple Monte Carlo scheme as we do here. This error then serves as upper bound on the normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,t}$, for any $t < T$. Table 3 also reveals that XGBoost outperforms Random Forest in the estimation of V_0, V_1 and V_T in most cases (8 cases out of 9). In this table we also report the normalized $L^2_{\mathbb{Q}}$-errors obtained with the kernel-based method in Boudabsa and Filipović (2022, Table 2). We see that the kernel-based method outperforms our ensemble learning method only in the BRC example. Figures 1(a), 2(a), and 3(a) show the decrease of the normalized $L^2_{\mathbb{Q}}$-error of $V_{X,1}$ with respect to the training sample size n. Figures 1(b), 2(b), and 3(c) illustrate the same phenomenon for $V_{X,T}$. In these figures we also recognize the outperformance of XGBoost over Random Forest.

Second, we compute and compare quantiles of $V_{X,1}$ and V_1, and $V_{X,T}$ and V_T using the test and training sample. Thereto, for $t \in \{1, T\}$, we compute the empirical left quantiles of $V_{X,t}$ and V_t at levels $\{0.001\%, 0.002\%, \dots, 0.009\%\}$,

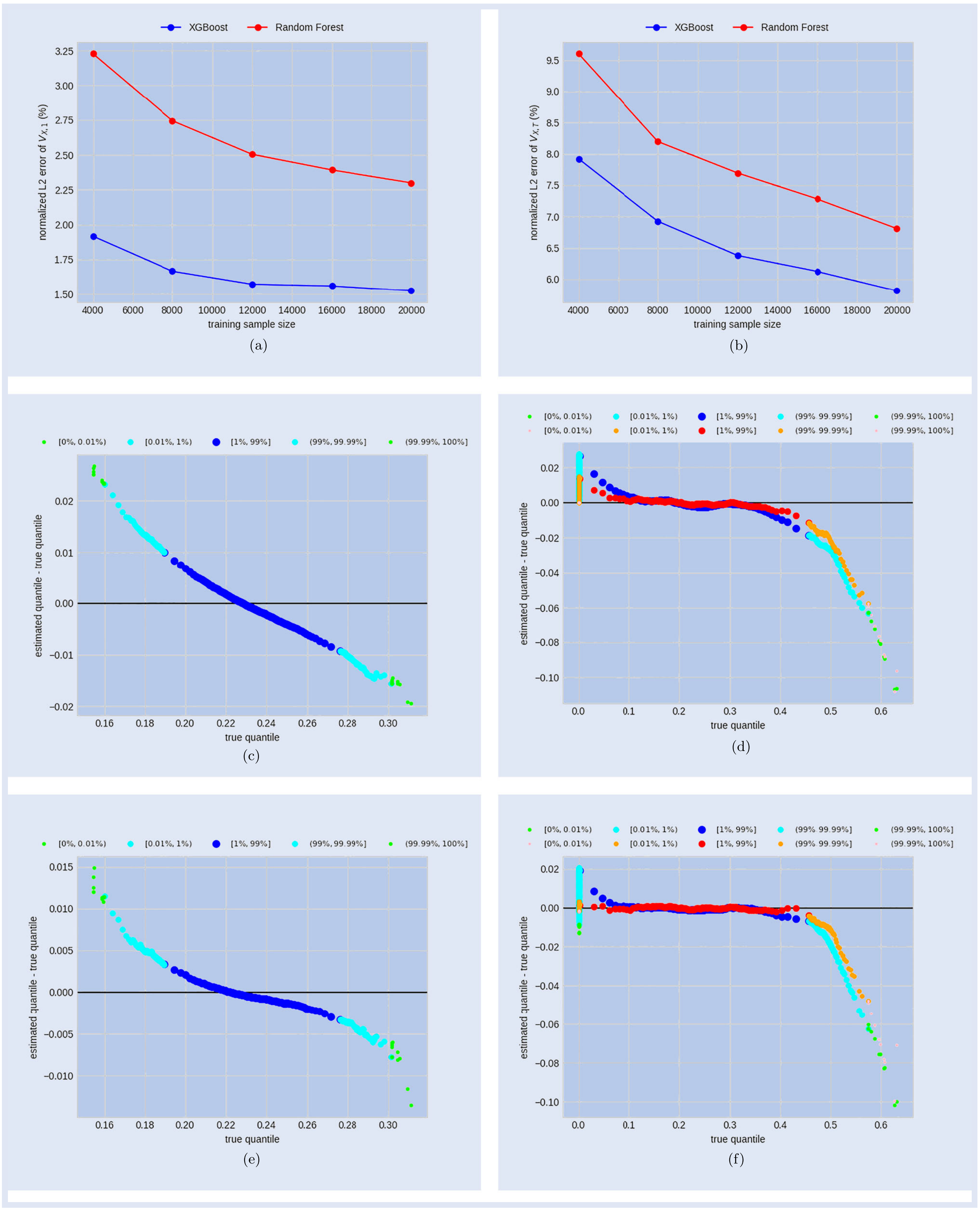

Figure 1. Results for the min-put with Random Forest and XGBoost. The normalized $L^2_{\mathbb{Q}}$-error of $V_{X,1}$, $\|V_1 - V_{X,1}\|_{2,\mathbb{Q}}/V_0$, is computed using the test sample and expressed in %. In the detrended Q-Q plots, the blue, cyan, and lawngreen (red, orange, and pink) dots are built using the test (training) sample. $[0\%, 0.01\%)$ refers to the quantiles of levels $\{0.001\%, 0.002\%, \dots, 0.009\%\}$, $[0.01\%, 1\%)$ refers to the quantiles of levels $\{0.01\%, 0.02\%, \dots, 0.99\%\}$, $[1\%, 99\%]$ refers to the quantiles of levels $\{1\%, 2\%, \dots, 99\%\}$, $(99\%, 99.99\%]$ refers to the quantiles of levels $\{99.01\%, 99.02\%, \dots, 99.99\%\}$, and $(99.99\%, 100\%]$ refers to the quantiles of levels $\{99.991\%, 99.992\%, \dots, 100\%\}$. (a) Normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ in % with Random Forest and XGBoost. (b) Normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,T}$ in % with Random Forest and XGBoost. (c) Detrended Q-Q plot of $V_{X,1}$ with Random Forest. (d) Detrended Q-Q plot of $V_{X,T}$ with Random Forest. (e) Detrended Q-Q plot of $V_{X,1}$ with XGBoost and (f) Detrended Q-Q plot of $V_{X,T}$ with XGBoost.

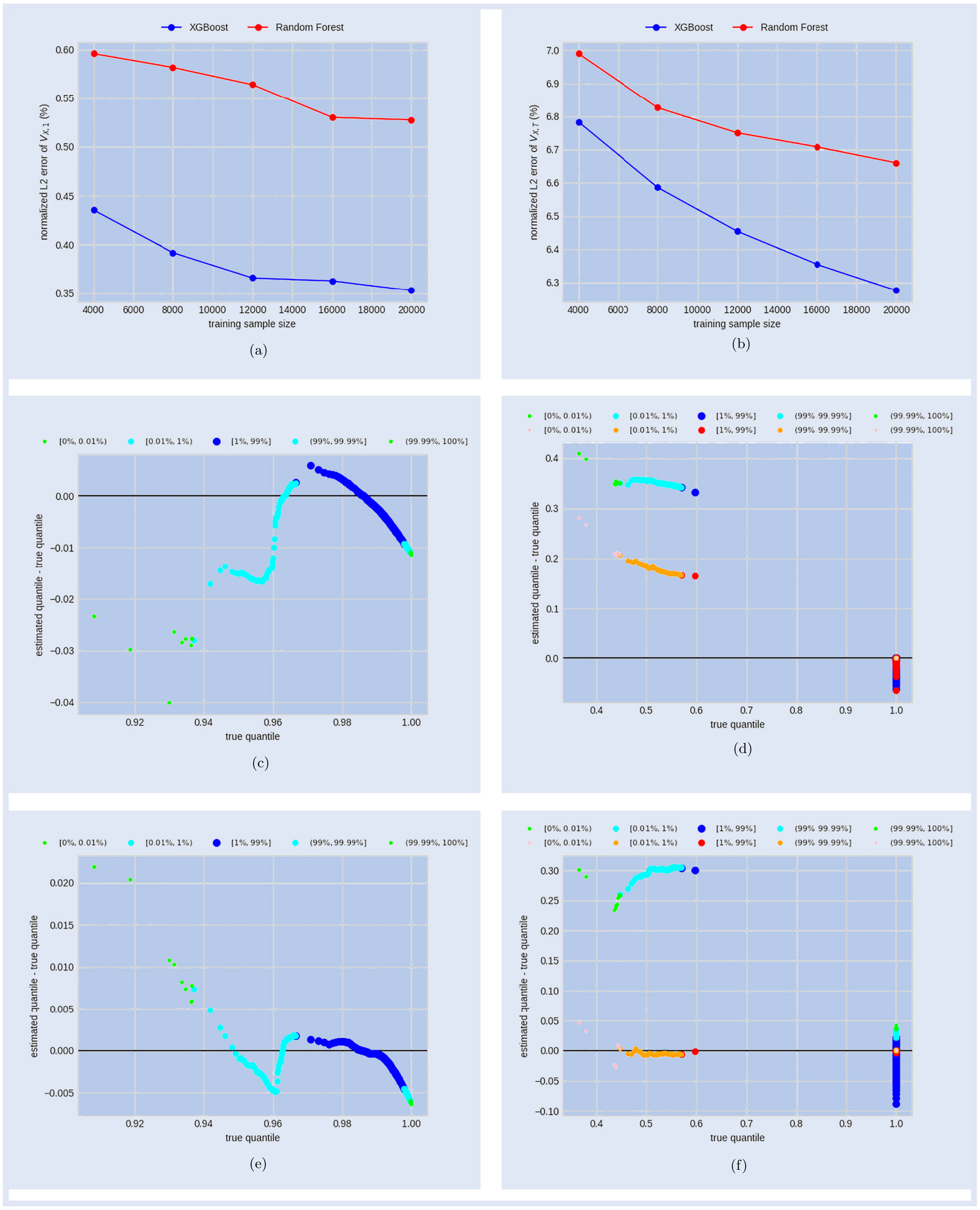

Figure 2. Results for the BRC with Random Forest and XGBoost. The normalized $L^2_{\mathbb{Q}}$-error of $V_{X,1}$, $\|V_1 - V_{X,1}\|_{2,\mathbb{Q}}/V_0$, is computed using the test sample and expressed in %. In the detrended Q-Q plots, the blue, cyan, and lawngreen (red, orange, and pink) dots are built using the test (training) sample. $[0\%, 0.01\%)$ refers to the quantiles of levels $\{0.001\%, 0.002\%, \ldots, 0.009\%\}$, $[0.01\%, 1\%)$ refers to the quantiles of levels $\{0.01\%, 0.02\%, \ldots, 0.99\%\}$, $[1\%, 99\%]$ refers to the quantiles of levels $\{1\%, 2\%, \ldots, 99\%\}$, $(99\%, 99.99\%]$ refers to the quantiles of levels $\{99.01\%, 99.02\%, \ldots, 99.99\%\}$, and $(99.99\%, 100\%]$ refers to the quantiles of levels $\{99.991\%, 99.992\%, \ldots, 100\%\}$. (a) Normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ in % with Random Forest and XGBoost. (b) Normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,T}$ in % with Random Forest and XGBoost. (c) Detrended Q-Q plot of $V_{X,1}$ with Random Forest. (d) Detrended Q-Q plot of $V_{X,T}$ with Random Forest. (e) Detrended Q-Q plot of $V_{X,1}$ with XGBoost and (f) Detrended Q-Q plot of $V_{X,T}$ with XGBoost.

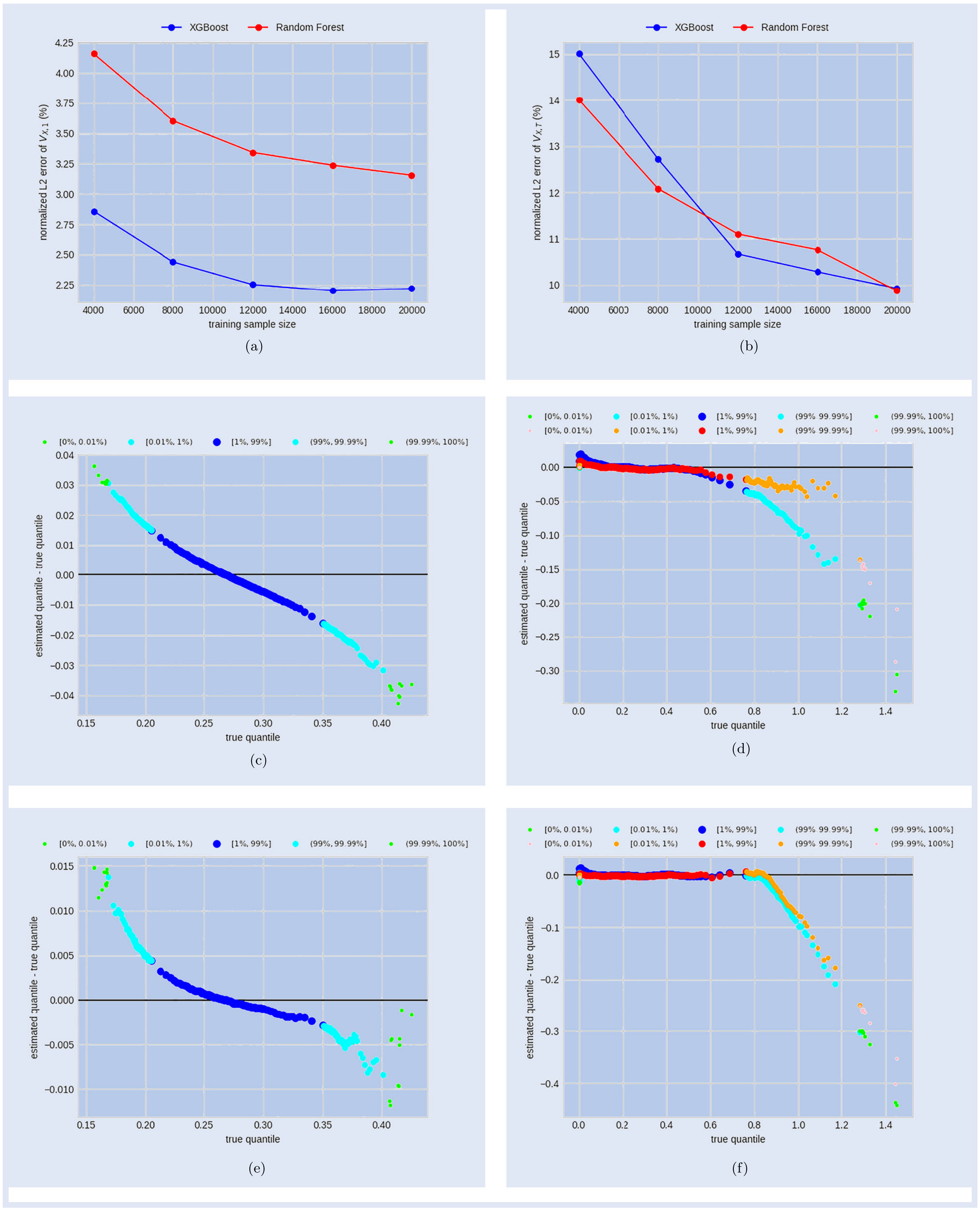

Figure 3. Results for the max-call with Random Forest and XGBoost. The normalized $L^2_{\mathbb{Q}}$-error of $V_{X,1}$, $\|V_1 - V_{X,1}\|_{2,\mathbb{Q}}/V_0$, is computed using the test sample and expressed in %. In the detrended Q-Q plots, the blue, cyan, and lawngreen (red, orange, and pink) dots are built using the test (training) sample. $[0\%, 0.01\%)$ refers to the quantiles of levels $\{0.001\%, 0.002\%, \ldots, 0.009\%\}$, $[0.01\%, 1\%)$ refers to the quantiles of levels $\{0.01\%, 0.02\%, \ldots, 0.99\%\}$, $[1\%, 99\%]$ refers to the quantiles of levels $\{1\%, 2\%, \ldots, 99\%\}$, $(99\%, 99.99\%]$ refers to the quantiles of levels $\{99.01\%, 99.02\%, \ldots, 99.99\%\}$, and $(99.99\%, 100\%]$ refers to the quantiles of levels $\{99.991\%, 99.992\%, \ldots, 100\%\}$. (a) Normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ in % with Random Forest and XGBoost. (b) Normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,T}$ in % with Random Forest and XGBoost. (c) Detrended Q-Q plot of $V_{X,1}$ with Random Forest. (d) Detrended Q-Q plot of $V_{X,T}$ with Random Forest. (e) Detrended Q-Q plot of $V_{X,1}$ with XGBoost and (f) Detrended Q-Q plot of $V_{X,T}$ with XGBoost.

$\{0.01\%, 0.02\%, \ldots, 0.99\%\}$, $\{1\%, 2\%, \ldots, 99\%\}$, $\{99.01\%, 99.02\%, \ldots, 99.99\%\}$, and $\{99.991\%, 99.992\%, \ldots, 100\%\}$.[†] The detrended quantiles (estimated quantiles minus true quantiles) are then plotted against the true quantiles, the produced plot is called a detrended Q-Q plot. Figures 1(c), 2(c), 3(d), and 1(e), 2(e), 3(e) show the detrended Q-Q plots of $V_{X,1}$ with Random Forest and XGBoost, respectively. These figures show that overall the distribution of V_1 is better estimated with XGBoost than with Random Forest. Figures 1(d), 2(d), 3(d), and 1(f), 2(f), 3(f) show the detrended Q-Q plots of $V_{X,T}$ with Random Forest and XGBoost, respectively. Notably, the detrended Q-Q plots of $V_{X,T}$ in figure 2(d,f) reveal that for less than 3% of the training sample (that is, less than 600 points out of $n = 20\,000$), the embedded min-put options in the BRC are triggered and in the money. For the remaining sample points the payoff is equal to the face value, $F = 1$. And yet, as figure 2(c,e) show, this is enough for our ensemble learning method to learn the payoff function such that $V_{X,1}$ is remarkably close to the ground truth, with a normalized $L^2_{\mathbb{Q}}$-error less than 0.60%, as reported in table 3. In Boudabsa and Filipović (2022), they also compute the same detrended Q-Q plots as here. Overall, the detrended Q-Q plots drawn with the kernel-based method, see Boudabsa and Filipović (2022, Figures 1–3), and those drawn with the ensemble learning method are of comparable quality.

Third, as risk management application, we compute the value at risk and expected shortfall of long and short positions of the above portfolios. Thereto, we recall the definitions that can also be found in Föllmer and Schied (2004, Chapter 4). For a confidence level $\alpha \in (0, 1)$ and random loss L, the value at risk of L is defined as left α-quantile $\mathrm{VaR}_\alpha(\mathrm{L}) = \inf\{y \mid \mathbb{P}[\mathrm{L} \le y] \ge \alpha\}$, and the expected shortfall of L is given by $\mathrm{ES}_\alpha(\mathrm{L}) = \frac{1}{1-\alpha}\mathbb{E}_{\mathbb{P}}[(\mathrm{L} - \mathrm{VaR}_\alpha(\mathrm{L}))^+] + \mathrm{VaR}_\alpha(\mathrm{L})$. Both value at risk and expected shortfall are standard risk measures in practice. For instance, insurance companies have to compute the value at risk at level $\alpha = 99.5\%$ and the expected shortfall at level $\alpha = 99\%$, under Solvency II and the Swiss Solvency Test, respectively. For more discussion on these two risk measures we refer to McNeil *et al.* (2015). Henceforth, we assume the real-world measure $\mathbb{P} = \mathbb{Q}$, for simplicity. For the three payoff functions above, we compute value at risk and expected shortfall of the 1-period loss $\mathrm{L} = V_0 - V_1$ and its estimator $\mathrm{L}_X = V_{X,0} - V_{X,1}$ for a long position, namely $\mathrm{VaR}_{99.5\%}(\mathrm{L})$, $\mathrm{ES}_{99\%}(\mathrm{L})$, $\mathrm{VaR}_{99.5\%}(\mathrm{L}_X)$, and $\mathrm{ES}_{99\%}(\mathrm{L}_X)$. We compute the same risk measures for a short position, namely $\mathrm{VaR}_{99.5\%}(-\mathrm{L})$, $\mathrm{ES}_{99\%}(-\mathrm{L})$, $\mathrm{VaR}_{99.5\%}(-\mathrm{L}_X)$, and $\mathrm{ES}_{99\%}(-\mathrm{L}_X)$. And in tables 4 and 5, we report the relative errors of risk measures, namely estimated risk measure minus true risk measure)/true risk measure, which are expressed in %. From these two tables, we notice that all risk measures are more accurately estimated with XGBoost than with Random Forest. This echoes the detrended Q-Q plots of $V_{X,1}$, where we see in figures 1(c), 2(c), 3(d) and 1(e), 2(e), 3(e) that the left and right tails of the distribution of V_1 are better estimated with XGBoost than with Random Forest. With XGBoost the estimates of risk measures are satisfactory. In tables 4 and 5, we also compute the relative errors of risk measures with the kernel-based method using the risk measurements in Boudabsa and Filipović (2022, Tables 3–4). We observe that risk measures are better estimated with the kernel-based method than with the ensemble learning method. Nevertheless, one should keep in mind that these risk measures are a tough metric for our estimators, because they focus on the tails of the distribution beyond the 1%- and 99%-quantiles.

[†] Note that for the test sample of size $n_{\text{test}} = 10^5$, the left 0.001%-quantile (100%-quantile) corresponds to the smallest (largest) sample value. For the training sample of size $n_{\text{train}} = 2 \times 10^4$, the same holds, while the ten left- and right-most quantiles collapse to two values, respectively.

Table 4. Relative errors of value at risk $\mathrm{VaR}_{99.5\%}(\mathrm{L}_X)$ and $\mathrm{VaR}_{99.5\%}(-\mathrm{L}_X)$, computed as (estimated VaR minus true VaR)/true VaR using the test sample and expressed in %, using XGBoost and Random Forest, for the payoff functions min-put, BRC, and max-call.

Payoff	Estimator	$\mathrm{VaR}(\mathrm{L}_X)$	$\mathrm{VaR}(-\mathrm{L}_X)$
Min-put	XGBoost	− 9.658%	− 6.912%
	Random Forest	− 25.69%	− 20.57%
	Kernel-based method	**0.9695%**	**3.158%**
BRC	XGBoost	2.533%	− 42.85%
	Random Forest	− 3.510%	− 75.87%
	Kernel-based method	**0.1893%**	**− 13.91%**
Max-call	XGBoost	− 7.103%	− 4.140%
	Random Forest	− 23.87%	− 20.51%
	Kernel-based method	**0.07143%**	**− 3.582%**

Table 5. Relative errors of expected shortfall $\mathrm{ES}_{99\%}(\mathrm{L}_X)$ and $\mathrm{ES}_{99\%}(-\mathrm{L}_X)$, computed as (estimated ES minus true ES)/true ES using the test sample and expressed in %, using XGBoost and Random Forest, for the payoff functions min-put, BRC, and max-call.

Payoff	Estimator	$\mathrm{ES}(\mathrm{L}_X)$	$\mathrm{ES}(-\mathrm{L}_X)$
Min-put	XGBoost	− 10.23%	− 7.434%
	Random Forest	− 26.41%	− 21.07%
	Kernel-based method	**1.261%**	**4.769%**
BRC	XGBoost	3.940%	− 43.79%
	Random Forest	16.93%	− 76.33%
	Kernel-based method	**− 0.5269%**	**− 14.40%**
Max-call	XGBoost	− 7.808%	− 4.507%
	Random Forest	− 24.67%	− 21.31%
	Kernel-based method	**− 0.3460%**	**− 3.588%**

5. Outlook

In this section we discuss two future research directions. The first one is computational: how to deal with higher dimensional problems? The second one is theoretical: is the value process estimator V_X asymptotically consistent?

5.1. *Computational scalability*

To apply our ensemble learning method to high dimensional problems, where both the sample size n and the path space dimension $d \times T$ are very large, two conditions must be satisfied from a computational point of view. First, fast training of

Table 6. Training time, in seconds, of $f_{\boldsymbol{X}}$, using XGBoost and Random Forest, using the training sample $\boldsymbol{X}$ and the function values $\boldsymbol{f}$.

Payoff	Estimator	Training time (s)
Min-put	XGBoost	2
	Random Forest	1
BRC	XGBoost	8
	Random Forest	12
Max-call	XGBoost	2
	Random Forest	1

Notes: Computation is performed on 5 compute nodes, each has 2 Skylake processors running at 2.3 GHz, with 18 cores per processor. And we used 188 GB of RAM. We needed a large amount of RAM in order to evaluate V_1 and $V_{X,1}$ on a test sample of size $n_{\text{test}} = 100\,000$, see Section 4. However, for the training of $f_{\boldsymbol{X}}$ one just needs a sufficient amount of memory to store $\boldsymbol{X}$ and $\boldsymbol{f}$.

Table 7. Number of hyperrectangles N in the ensemble estimator $f_{\boldsymbol{X}}$ in (5), using XGBoost and Random Forest, and time in seconds to evaluate $V_{X,1}$ on a test sample $\boldsymbol{X}_{\text{test}}$ of size $n_{\text{test}} = 100\,000$.

Payoff	Estimator	Number of hyperrectangles N	Time to evaluate $V_{X,1}$ on $\boldsymbol{X}_{\text{test}}$ (s)
Min-put	XGBoost	88 127	413
	Random Forest	494 118	1629
BRC	XGBoost	189 864	10 947
	Random Forest	187710	13 472
Max-call	XGBoost	66 225	353
	Random Forest	489 747	1644

Notes: The number of hyperrectangles are copied from tables 1 and 2. Computation is performed on 5 compute nodes, each has 2 Skylake processors running at 2.3 GHz, with 18 cores per processor. And we used 188 GB of RAM. We needed a large amount of RAM in order to evaluate V_1 and $V_{X,1}$ on a test sample $\boldsymbol{X}_{\text{test}}$ of size $n_{\text{test}} = 100\,000$, see Section 4. However, for the training of $f_{\boldsymbol{X}}$ one just needs a sufficient amount of memory to store $\boldsymbol{X}$ and $\boldsymbol{f}$.

the ensemble estimator f_X in (5) subject to an accurate approximation. Second, fast evaluation of the value process estimator V_X in (6).

In the setting of our numerical experiments, the first condition is satisfied. As shown in table 6, the training of f_X is extremely fast. This speed comes from two sources: a relatively small training time complexity, and the exploitation of parallel processing. In fact, the training time complexity for building a Random Forest $f_{X,\Pi}$ in (12), under the bootstrapping sampling regime, is $\Theta(Mp\tilde{n}\log(\tilde{n}))$, where $\tilde{n} = 0.632n$, see Louppe (2014, Table 5.1). And if we consider the XGBoost implementation of Gradient Boosting $f_{X,t}$ in (13), this complexity becomes $\Theta(M\,\mathbf{max\_depth}\,dTn\log(n))$, see Chen and Guestrin (2016, Section Time Complexity Analysis). The notation $\Theta(g(n))$, for some function g, means that there exist constants $0 < c < C$, such that the training time complexity is bounded from below by $cg(n)$, and bounded from above by $Cg(n)$, as $n \to \infty$. Furthermore, the Random Forest implementation RandomForestRegressor in scikit-learn (Pedregosa *et al.* 2011), and the Gradient Boosting implementation XGBRegressor in XGBoost (Chen and Guestrin 2016) take advantage of parallel processing. A direction of future research could explore the scalability of Random Forest and Gradient Boosting to portfolios of larger dimensions than in our numerical experiments, while keeping the estimation error small. In fact, Buhlmann (2003, Section 2.1) give empirical evidence that Gradient Boosting is able to deal with high dimensional problems.

Now we shall discuss the second condition. The estimator f_X provided by a machine learning library, such as scikit-learn or XGBoost, should be rewritten into a suitable format. We recall the expression of f_X in (5). Thus f_X is determined by the hyperrectangles $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_N$, and the real coefficients $\beta_1, \ldots, \beta_N$. And from (8), we know that every hyperrectangle $\boldsymbol{A}_i$ is characterized by two matrices $a^{(i)}, b^{(i)} \in \mathbb{R}^{d\times T}$ such that $\boldsymbol{A}_i = (a^{(i)}, b^{(i)}]$. In summary, f_X is determined by the tuple $(\mathbf{cells}, \mathbf{values})$, where $\mathbf{cells} = ((a^{(1)}, b^{(1)}), \ldots, (a^{(N)}, b^{(N)}))$, and $\mathbf{values} = (\beta_1, \ldots, \beta_N)$. The rewriting of f_X as $(\mathbf{cells}, \mathbf{values})$ forms a processing step, and it requires a careful look at how f_X is encoded in the corresponding library.† After this processing step, the pseudo-code in Algorithm 1 shows how to evaluate $V_{X,t}$ at some point $(x_1, \ldots, x_t) \in \mathbb{R}^{d\times t}$. Thus fast evaluation of the value process V_X can be achieved by writing a fast code of Algorithm 1.

Recall that in Section 4, for each payoff function, we evaluate the function $V_{X,1}$ on a test sample of size $n_{\text{test}} = 100\,000$. To achieve this we parallelized Algorithm 1 using the packages MPI for Python Dalcin and Fang (2021) and joblib Team (2022). Specifically, MPI for Python allowed us to parallelize the 100 000 evaluations using several compute nodes. And joblib allowed us to parallelize the for loop in Algorithm 1. Table 7 shows the time to evaluate $V_{X,1}$. We observe that evaluations of $V_{X,1}$ are faster with XGBoost than with Random Forest, which is mainly due to the difference in the number of hyperrectangles N in these two estimators. The time to evaluate $V_{X,1}$ in table 7 can be shortened further using, e.g. a GPU code of Algorithm 1.

5.2. Consistency

In Section 4, we saw that our estimator V_X in (6) gives an accurate estimation of the value process V in (1). In order to theoretically characterize the goodness of the estimator V_X, from Doob's maximal inequality in (7), we see that it is enough to study the ensemble estimator f_X. In particular, we would be interested in consistency results of the form $\mathbb{E}_{\boldsymbol{Q}}[\|f - f_X\|^2_{2,\mathbb{Q}}] \to 0$, as $n \to \infty$, and finite sample guarantees of the form $\boldsymbol{Q}[\|f - f_X\|^2_{2,\mathbb{Q}} < c(\eta, n)] \geq 1 - \eta$, for all $n \geq n_0(\eta)$, for $\eta \in (0, 1]$. However, theoretical analysis of Random Forest, when f_X is $f_{X,\Pi}$ in (12), and Gradient Boosting, when f_X is $f_{X,t}$ in (13), is difficult in general, especially in our financial framework, where the function f typically is neither bounded nor compactly supported, see the max-call option

† Python codes corresponding to the processing steps of RandomForestRegressor of scikit-learn, and XGBRegressor of XGBoost are available from the authors upon request.

Algorithm 1: Evaluation of $V_{X,t}$

```
Input: cells, values, $t$, $(x_1, \dots, x_t)$;
value = 0;
ncells = length(values);
for $i = 1, \dots,$ ncells do
    $(a, b)$ = cells[$i$];
    if $a_{j,s} < x_{j,s} \le b_{j,s}$, $j = 1, \dots, d$, and $s = 1, \dots, t$ then
        value = value + values[$i$] $\times \prod_{s=t+1}^{T} \mathbb{Q}_s[(a_s, b_s]]$
    end
end
return value;
```

example in Section 4. Below we mention two recent consistency results available in the machine learning literature that drew our attention and that could be investigated further.

Despite the plethora of empirical works on Random Forest, see, e.g. Genuer *et al.* (2008), Archer and Kimes (2008), and Genuer *et al.* (2010), to name a few, little is known on the theoretical side. The sampling scheme, the split criterion (11), and the sampling of p coordinates for each hyperrectangle to split make the trees highly and non-trivially $(\boldsymbol{X},\boldsymbol{f})$-dependent. This renders the Random Forest difficult to analyse mathematically. To better understand theoretically the good performance of Random Forest in practice, less $(\boldsymbol{X},\boldsymbol{f})$-dependent versions of Random Forest have been studied, e.g. $(\boldsymbol{X},\boldsymbol{f})$-independent in Biau (2012), or $\boldsymbol{f}$-independent in Scornet (2016). A recent consistency result for the asymptotic Random Forest, $\lim_{M\to\infty} f_{X,\Pi}$ in (12), has been given in Scornet *et al.* (2015). They show that $\mathbb{E}_Q[\| \lim_{M\to\infty} f_{X,\Pi} - f\|_{2,\mathbb{Q}}^2] \to 0$, as $n \to \infty$, under the following assumptions: the path space is the unit cube $[0,1]^{d\times T}$ instead of $\mathbb{R}^{d\times T}$, X is uniformly distributed on $[0,1]^{d\times T}$, and f is continuous and additive. The last assumption reads in our case that $f(x) = \sum_{t=1}^{T}\sum_{j=1}^{d} f_{j,t}(x_{j,t})$, where each $f_{j,t}$ is continuous. These assumptions are too stringent in applications in finance. We recommend the survey (Biau and Scornet 2016) for an overview on the theoretical work on Random Forest.

As for Gradient Boosting, to the best of our knowledge, there is no consistency result for Gradient Boosting with CART in the context of regression problems in the literature. Nevertheless, we shall mention the recent paper Biau and Cadre (2021), where the authors study Gradient Boosting in both classification and regression problems. Their result (Biau and Cadre 2021, Theorem 4.1) holds in the case where the base estimator is a certain type of regression trees. However, it does not hold in the case where the base estimator is a CART regression tree.

6. Conclusion

We introduce a unified framework for quantitative portfolio risk management based on the dynamic value process of the portfolio. We use ensemble estimators with regression trees to learn the value process from a finite sample of the cumulative cash flow of the portfolio. Our portfolio value process estimator is fast to construct, given in closed form, and accurate. The last means that the normalized $L^2_{\mathbb{Q}}$-error $\| \max_{t=0,\dots,T} |V_t - V_{X,t}| \|_{2,\mathbb{Q}}/V_0$ is relatively small. In fact, numerical experiments for exotic and path-dependent options in the multivariate Black–Scholes model in moderate dimensions show good results for a moderate training sample size. In contrast to the kernel-based method in Boudabsa and Filipović (2022), our ensemble learning method can be scaled to deal with high dimensional problems.

Acknowledgments

We thank the editor and two anonymous referees for helpful comments.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

Archer, K.J. and Kimes, R.V., Empirical characterization of random forest variable importance measures. *Comput. Stat. Data Anal.*, 2008, **52**(4), 2249–2260.

Bartlett, P., Freund, Y., Lee, W.S. and Schapire, R.E., Boosting the margin: A new explanation for the effectiveness of voting methods. *Ann. Stat.*, 1998, **26**(5), 1651–1686.

Bauwens, L., Laurent, S. and Rombouts, J.V.K., Multivariate GARCH models: A survey. *J. Appl. Econom.*, 2006, **21**(1), 79–109.

BCV, Listing notice of a barrier reverse convertible, 2023. Available online at: https://www.bcv.ch/termsheet/35753694_en.pdf (accessed 16 March 2023).

Becker, S., Cheridito, P. and Jentzen, A., Deep optimal stopping. *J. Mach. Learn. Res.*, 2019, **20**(74), 1–25.

Biau, G., Analysis of a random forests model. *J. Mach. Learn. Res.*, 2012, **13**(38), 1063–1095.

Biau, G. and Cadre, B., Optimization by gradient boosting. In *Advances in Contemporary Statistics and Econometrics: Festschrift in Honor of Christine Thomas-Agnan*, edited by A. Daouia and A. Ruiz-Gazen, pp. 23–44, 2021 (Springer International Publishing: Cham).

Biau, G. and Scornet, E., A random forest guided tour. *Test*, 2016, **25**(2), 197–227.

Bollerslev, T., Generalized autoregressive conditional heteroskedasticity. *J. Econom.*, 1986, **31**(3), 307–327.

Boudabsa, L. and Filipović, D., Machine learning with kernels for portfolio valuation and risk management. *Finance Stoch.*, 2022, **26**(2), 131–172.

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

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

Breiman, L., Friedman, J., Stone, C. and Olshen, R., *Classification and Regression Trees*, The Wadsworth and Brooks-Cole statistics-probability series, 1984 (Taylor & Francis: New York).

Broadie, M., Du, Y. and Moallemi, C.C., Risk estimation via regression. *Oper. Res.*, 2015, **63**(5), 1077–1097.

Buhlmann, P, Boosting methods: Why they can be useful for high-dimensional data. In *Proceedings of the Third*

International Workshop on 'Distributed Statistical Computing' (DSC 2003), 2003 (Technische Universität Wien: Vienna). Available online at: https://www.r-project.org/conferences/DSC-2003/Proceedings/Buehlmann.pdf.

Cambou, M. and Filipović, D., Model uncertainty and scenario aggregation. *Math. Finance*, 2017, **27**(2), 534–567.

Cambou, M. and Filipović, D., Replicating portfolio approach to capital calculation. *Finance Stoch.*, 2018, **22**(1), 181–203.

Carriere, J.F., Valuation of the early-exercise price for options using simulations and nonparametric regression. *Insur.: Math. Econ.*, 1996, **19**(1), 19–30.

Chen, T. and Guestrin, C., XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, 2016 (ACM: New York, NY, USA).

Cheridito, P., Ery, J. and Wüthrich, M.V., Assessing asset-liability risk with neural networks. *Risks*, 2020, **8**(1), 16.

Dalcin, L. and Fang, Y.-L.L., mpi4py: Status update after 12 years of development. *Comput. Sci. Eng.*, 2021, **23**(4), 47–54.

Ech-Chafiq, Z.E.F., Henry-Labordere, P. and Lelong, J., Pricing Bermudan options using regression trees/random forests, 2021. arXiv:2201.02587.

Embrechts, P., Copulas: A personal view. *J. Risk. Insur.*, 2009, **76**(3), 639–650.

Fernandez-Arjona, L. and Filipović, D., A machine learning approach to portfolio pricing and risk management for high-dimensional problems. *Math. Finance*, 2022, **32**(4), 982–1019.

Föllmer, H. and Schied, A., *Stochastic finance*, Vol. 27 of *De Gruyter Studies in Mathematics*, 2004 (Walter de Gruyter & Co.: Berlin). Extended edition. An introduction in discrete time.

Freund, Y. and Schapire, R.E., Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning*, ICML'96, pp. 148–156, 1996 (Morgan Kaufmann Publishers Inc: San Francisco, CA, USA).

Friedman, J., Hastie, T. and Tibshirani, R., Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). *Ann. Statist.*, 2000, **28**(2), 337–407.

Friedman, J.H., Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 2001, **29**(5), 1189–1232.

Genuer, R. and Poggi, J.-M., Arbres CART et Forêts aléatoires, Importance et sélection de variables, 2017. Preprint.

Genuer, R., Poggi, J.-M. and Tuleau, C., Random forests: Some methodological insights, 2008. Preprint.

Genuer, R., Poggi, J.-M. and Tuleau-Malot, C., Variable selection using random forests. *Pattern Recognit. Lett.*, 2010, **31**(14), 2225–2236.

Genz, A., Numerical computation of multivariate normal probabilities. *J. Comput. Graph. Stat.*, 2000, **1**, 0–0.

Genz, A., Alan Genz website, software column, 2022. Available online at: http://www.math.wsu.edu/faculty/genz/homepage (accessed 29 March 2022).

Genz, A., Bretz, F., Miwa, T., Mi, X., Leisch, F., Scheipl, F. and Hothorn, T., *mvtnorm: Multivariate Normal and t Distributions*, 2021. R package version 1.1-3.

Glasserman, P. and Yu, B., Simulation for American options: Regression now or regression later? In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, edited by H. Niederreiter, pp. 213–226, 2004 (Springer: Berlin).

Gordy, M.B. and Juneja, S., Nested simulation in portfolio risk measurement. *Manage. Sci.*, 2010, **56**(10), 1833–1848.

Goudenège, L., Molent, A. and Zanette, A., Machine learning for pricing American options in high-dimensional Markovian and non-Markovian models. *Quant. Finance*, 2020, **20**(4), 573–591.

Ha, H. and Bauer, D., A least-squares Monte Carlo approach to the estimation of enterprise risk. *Finance Stoch.*, 2022, **26**(3), 417–459.

Hong, L., Juneja, S. and Liu, G., Kernel smoothing for nested estimation with application to portfolio risk measurement. *Oper. Res.*, 2017, **65**(3), 657–673.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. and Liu, T.-Y., LightGBM: A highly efficient gradient boosting decision tree. In *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, pp. 3146–3154, 2017 (Curran Associates, Inc).

Kearns, M., Thoughts on hypothesis boosting. Unpublished manuscript, 1988.

Liaw, A. and Wiener, M., Classification and regression by randomForest. *R News*, 2002, **2**(3), 18–22.

Liu, M. and Staum, J., Stochastic kriging for efficient nested simulation of expected shortfall. *J. Risk*, 2010, **12**(3), 3–27.

Loh, W.-Y., Fifty years of classification and regression trees. *Int. Stat. Rev.*, 2014, **82**(3), 329–348.

Longstaff, F. and Schwartz, E., Valuing American options by simulation: A simple least-squares approach. *Rev. Financ. Stud.*, 2001, **14**(1), 113–147.

Louppe, G., Understanding Random Forests: From Theory to Practice. PhD Thesis, University of Liège, 2014.

McNeil, A.J., Frey, R. and Embrechts, P., *Quantitative Risk Management: Concepts, Techniques and Tools*, 2015 (Princeton University Press: Princeton).

Morgan, J.N. and Sonquist, J.A., Problems in the analysis of survey data, and a proposal. *J. Am. Stat. Assoc.*, 1963, **58**(302), 415–434.

Natolski, J. and Werner, R., Mathematical analysis of different approaches for replicating portfolios. *Eur. Actuar. J.*, 2014, **4**(2), 411–435.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E., Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 2011, **12**, 2825–2830.

Pelsser, A. and Schweizer, J., The difference between LSMC and replicating portfolio in insurance liability modeling. *Eur. Actuar. J.*, 2016, **6**(2), 441–494.

Peskir, G. and Shiryaev, A., *Optimal Stopping and Free-Boundary Problems*, 2006 (Birkhäuser: Basel, Boston).

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V. and Gulin, A., Catboost: Unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems 31*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, pp. 6638–6648, 2018 (Curran Associates, Inc: Montréal).

Quinlan, J.R., *C4.5: Programs for Machine Learning*, 1993 (Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA).

Revuz, D. and Yor, M., *Continuous martingales and Brownian motion*, Vol. 293 of *Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]*. 2nd ed., 1994 (Springer: Berlin).

Risk, J. and Ludkovski, M., Sequential design and spatial modeling for portfolio tail risk measurement. *SIAM J. Financ. Math.*, 2018, **9**(4), 1137–1174.

Schapire, R.E., The strength of weak learnability. *Mach. Learn*, 1990, **5**, 197–227.

Scornet, E., On the asymptotics of random forests. *J. Multivar. Anal.*, 2016, **146**, 72–83. Special Issue on Statistical Models and Methods for High or Infinite Dimensional Spaces.

Scornet, E., Biau, G. and Vert, J.-P., Consistency of random forests. *Ann. Statist.*, 2015, **43**(4), 1716–1741.

Sklar, M., Fonctions de répartition à *n* dimensions et leurs marges. *Publ. Inst. Statist. Univ. Paris*, 1959, **8**, 229–231.

Team, J.D., Joblib: Running python functions as pipeline jobs, 2022. Available online at: https://joblib.readthedocs.io/en/latest/ (accessed 29 March 2022).

Tsitsiklis, J.N. and van Roy, B., Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Trans. Automat. Contr.*, 1999, **44**(10), 1840–1851.

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C., Polat, I., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F. and van Mulbregt, P., SciPy 1.0 Contributors, SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nat. Methods*, 2020, **17**(3), 261–272.

Appendix 1. Comparison with regress-now

The method we develop in this paper gives an estimation of the entire value process V. In practice, one could be interested in the estimation of the portfolio value V_t only at some fixed time t, e.g. $t = 1$. In Glasserman and Yu (2004) two least squares Monte Carlo methods are described to deal with this problem in the context of American options pricing. Their first method, termed 'regress-later', consists in estimating the payoff function f by means of a projection on a finite number of basis functions. The basis functions are chosen such that their conditional expectation at time $t = 1$ is in closed form. Our method can be seen as a double extension of this, because it covers the case where both the basis functions and their number are not known a priori, and it gives closed-form estimation of the portfolio value V_t at any time t. Their second method, termed 'regress-now', consists in estimating V_1 by means of a projection on a finite number of basis functions that depend solely on the variable of interest $x_1 \in \mathbb{R}^d$.

We compare our method, which corresponds to 'regress-later', and which gives the estimator $V_{\boldsymbol{X},t}$ in (14) for $t = 1$, to its regress-now variant, whose estimator we denote by $V^{\text{now}}_{\boldsymbol{X},1}$. Thereto we briefly discuss how to construct $V^{\text{now}}_{\boldsymbol{X},1}$ in the context of the three payoff functions studied in Section 4.

The construction of $V^{\text{now}}_{\boldsymbol{X},1}$ is simpler than that of $V_{\boldsymbol{X},1}$. First, instead of the whole sample $\boldsymbol{X}$, one only needs the $(t = 1)$-cross-section $\boldsymbol{X}_1 = (X^{(1)}_1, \ldots, X^{(n)}_1)$. Second, with the input $\boldsymbol{X}_1$ and $\boldsymbol{f}$, the estimators Random Forest in Section 2.2, and Gradient Boosting in Section 2.3 give directly $V^{\text{now}}_{\boldsymbol{X},1}$. As we did in Section 4, we use the validation sample $\boldsymbol{X}_{\text{valid}}$, along with its corresponding function values $\boldsymbol{f}_{\text{valid}}$, to find the optimal hyperparameter values for Random Forest and Gradient Boosting by validation on the sets of hyperparameter values $\mathcal{P}_{\text{RF}}$ and $\mathcal{P}_{\text{XGB}}$, respectively. Table A1 shows the normalized $L^2_{\mathbb{Q}}$-error $\|V^{\text{now}}_{\boldsymbol{X},1} - f\|_{2,\mathbb{Q}}/V_0$, computed using the validation sample $\boldsymbol{X}_{\text{valid}}$, and the number of hyperrectangles N in the Random Forest and Gradient Boosting $f_{\boldsymbol{X}}$ in (5) for the optimal hyperparameter values in $\mathcal{P}_{\text{RF}}$ and $\mathcal{P}_{\text{XGB}}$. Unlike in Section 4, here we choose the optimal hyperparameter value for Random Forest, although the number of hyperrectangles N induced is very large ($N > 500\,000$). This is because the evaluation of $V^{\text{now}}_{\boldsymbol{X},1}$ is very fast irrespectively of N, thanks to the highly optimized implementations of Random Forest and Gradient Boosting, RandomForestRegressor in scikit-learn (Pedregosa *et al.* 2011) and XGBRegressor in XGBoost (Chen and Guestrin 2016), respectively.

Table A2 shows the normalized $L^2_{\mathbb{Q}}$-errors of $V_{\boldsymbol{X},1}$ and $V^{\text{now}}_{\boldsymbol{X},1}$. The former values are copied from table 3 for convenience. We observe that our regress-later estimators always perform better than their regress-now variants. This finding is confirmed by the normalized $L^2_{\mathbb{Q}}$-errors of $V_{\boldsymbol{X},1}$ and $V^{\text{now}}_{\boldsymbol{X},1}$ as function of the training sample in figures A1(a,c,e), A2(a,c,e). In Boudabsa and Filipović (2022), they also compare regress-later and regress-now with the kernel-based method. In table A2 we report, from Boudabsa and Filipović (2022, Table 5), the $L^2_{\mathbb{Q}}$-errors corresponding to the kernel-based method. We see that also with the kernel-based method, regress-later outperforms regress-now.

Now we discuss the detrended Q-Q plots in figure A1(b,d,f) for Random Forest, and in figure A2(b,d,f) for XGBoost. Their construction is detailed in the second-to-last paragraph of Section 4. In figure A2(b,d,f), we see that for XGBoost, the detrended Q-Q plots with regress-later are of much better quality, i.e. they are more aligned with the horizontal black line, than the detrended Q-Q plots with regress-now. As for Random Forest, the outperformance of regress-later over regress-now in terms of normalized $L^2_{\mathbb{Q}}$-error, seen in figure A1(a,c,e), does not clearly appear in the detrended Q-Q plots in figure A1(b,d,f). By comparing figures A1(b) and A2(b); figures A1(d) and A2(d); figures A1(f) and A2(f), we see that regress-later XGBoost gives the best detrended Q-Q plots, i.e. the detrended Q-Q plots that are the most aligned with the horizontal black line.

We finish this section by discussing the risk measure estimates in tables A3 and A4. Their construction is detailed in the last paragraph of Section 4. Consistently with the last comment in the above paragraph, it is regress-later XGBoost that gives best estimates of risk measures in most cases, 10 cases out of 12. In the 2 left cases, which correspond to the estimation of risk measures of the short position of the BRC, it is regress-now Random Forest that is the most accurate. The last is consistent with the detrended Q-Q plots in figures A1(d) and A2(d), where we observe that regress-now Random Forest gives the best estimation of the right tail distribution of V_1 among all estimators.

Table A1. Regress-now Random Forest and regress-now XGBoost validation steps: normalized $L^2_{\mathbb{Q}}$-error $\|V^{\text{now}}_{\boldsymbol{X},1} - f\|_{2,\mathbb{Q}}/V_0$, computed using the validation sample $\boldsymbol{X}_{\text{valid}}$ and expressed in %, and number of hyperrectangles N in the Random Forest and Gradient Boosting $f_{\boldsymbol{X}}$ in (5), for the optimal hyperparameter values in $\mathcal{P}_{\text{RF}}$ and $\mathcal{P}_{\text{XGB}}$, for the payoff functions min-put, BRC, and max-call.

	Min-put	BRC	Max-call
Optimal hyperparameter value for Random Forest	(500, 5, 2)	(500, 5, 1)	(500, 5, 2)
Normalized $L^2_{\mathbb{Q}}$-error	31.98%	7.195%	46.58%
Number of hyperrectangles	2 619 900	323 896	2 620 048
Optimal hyperparameter value for Gradient Boosting	(44, 40, 45)	(52, 40, 45)	(41, 50, 45)
Normalized $L^2_{\mathbb{Q}}$-error	32.43%	6.886%	46.62%
Number of hyperrectangles	13 401	9606	13 165

Table A2. Normalized $L^2_{\mathbb{Q}}$-error $\|V_1 - \widehat{V}_1\|_{2,\mathbb{Q}}/V_0$, computed using the test sample and expressed in %, for $\widehat{V}_1 \in \{V_{\boldsymbol{X},1}, V^{\text{now}}_{\boldsymbol{X},1}\}$, for the payoff functions min-put, BRC, and max-call.

Payoff	Estimator	$V_{\boldsymbol{X},1}$	$V^{\text{now}}_{\boldsymbol{X},1}$
Min-put	XGBoost	***1.525%***	9.539%
	Random Forest	**2.300%**	6.487%
	Kernel-based method	**1.827%**	1.946%
BRC	XGBoost	**0.3530%**	1.492%
	Random Forest	**0.5276%**	1.499%
	Kernel-based method	***0.2506%***	0.2806%
Max-call	XGBoost	***2.217%***	14.24%
	Random Forest	**3.155%**	9.863%
	Kernel-based method	**2.315%**	2.606%

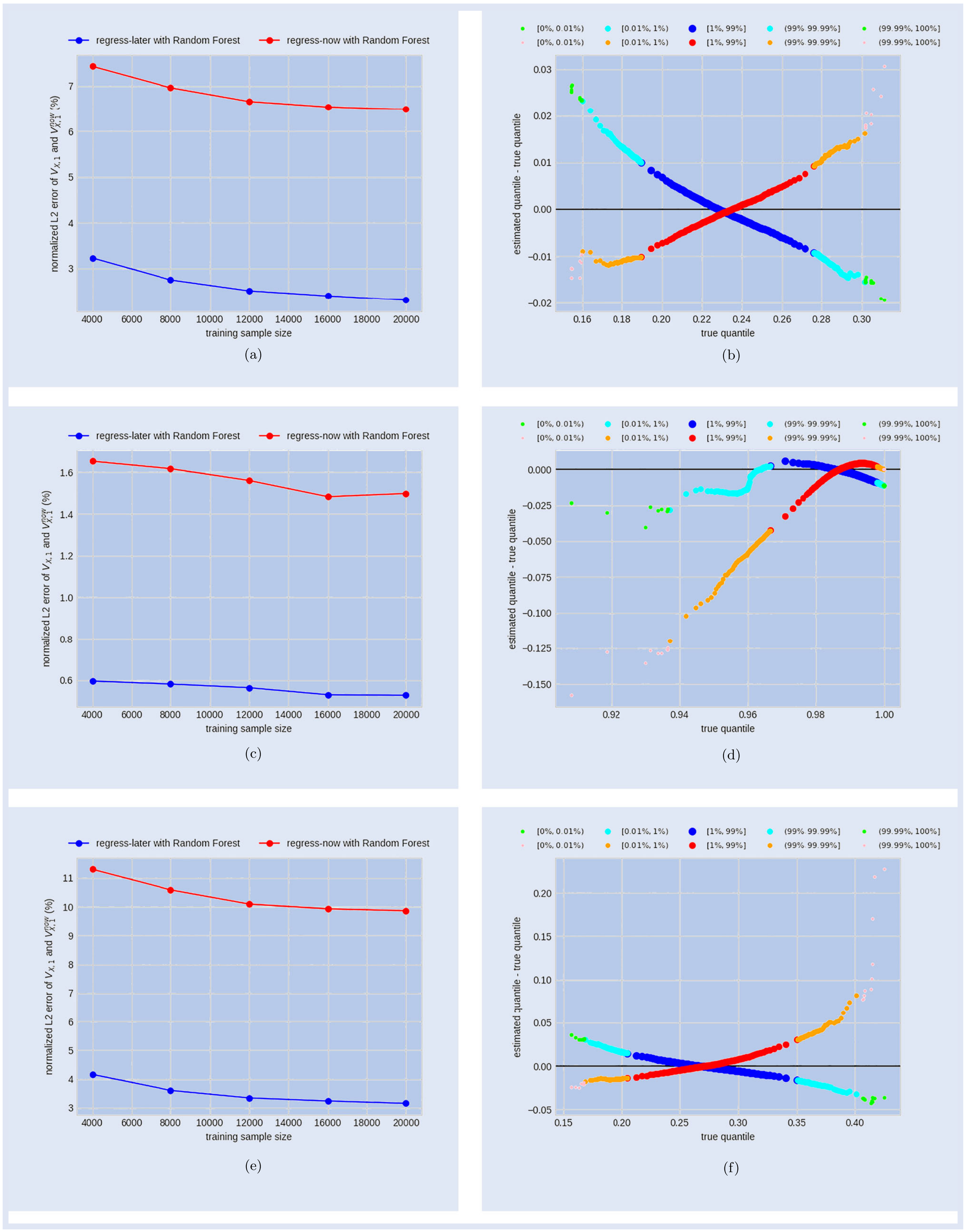

Figure A1. Results for the min-put, BRC, and max-call with Random Forest. The normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ and $V^{\text{now}}_{X,1}$, $\|V_1 - V_{X,1}\|_{2,\mathbb{Q}}/V_0$ and $\|V_1 - V^{\text{now}}_{X,1}\|_{2,\mathbb{Q}}/V_0$, are computed using the test sample and expressed in %. In the detrended Q-Q plots, the blue, cyan, and lawngreen (red, orange, and pink) dots are built using regress-later Random Forest (regress-now Random Forest) and the test sample. $[0\%, 0.01\%)$ refers to the quantiles of levels $\{0.001\%, 0.002\%, \ldots, 0.009\%\}$, $[0.01\%, 1\%)$ refers to the quantiles of levels $\{0.01\%, 0.02\%, \ldots, 0.99\%\}$, $[1\%, 99\%]$ refers to the quantiles of levels $\{1\%, 2\%, \ldots, 99\%\}$, $(99\%, 99.99\%]$ refers to the quantiles of levels $\{99.01\%, 99.02\%, \ldots, 99.99\%\}$, and $(99.99\%, 100\%]$ refers to the quantiles of levels $\{99.991\%, 99.992\%, \ldots, 100\%\}$. (a) Min-put with Random Forest: normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ and $V^{\text{now}}_{X,1}$ in %. (b) Min-put with Random Forest: detrended Q-Q plots of $V_{X,1}$ and $V^{\text{now}}_{X,1}$. (c) BRC with Random Forest: normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ and $V^{\text{now}}_{X,1}$ in %. (d) BRC with Random Forest: detrended Q-Q plots of $V_{X,1}$ and $V^{\text{now}}_{X,1}$. (e) Max-call with Random Forest: normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ and $V^{\text{now}}_{X,1}$ in % and (f) Max-call with Random Forest: detrended Q-Q plots of $V_{X,1}$ and $V^{\text{now}}_{X,1}$.

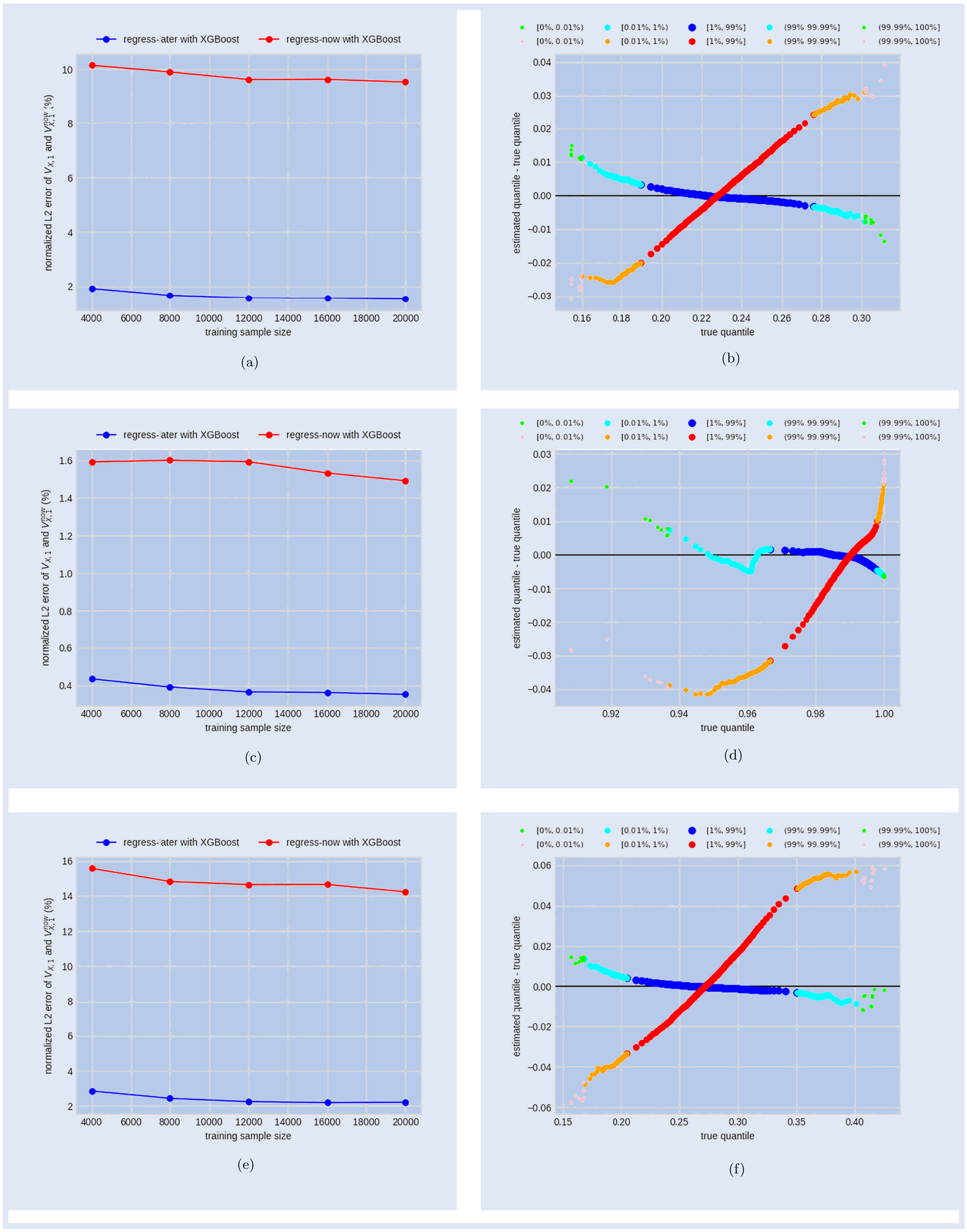

Figure A2. Results for the min-put, BRC, and max-call with XGBoost. The normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ and $V^{\text{now}}_{X,1}$, $\|V_1 - V_{X,1}\|_{2,\mathbb{Q}}/V_0$ and $\|V_1 - V^{\text{now}}_{X,1}\|_{2,\mathbb{Q}}/V_0$, are computed using the test sample and expressed in %. In the detrended Q-Q plots, the blue, cyan, and lawngreen (red, orange, and pink) dots are built using regress-later XGBoost (regress-now XGBoost) and the test sample. $[0\%, 0.01\%)$ refers to the quantiles of levels $\{0.001\%, 0.002\%, \ldots, 0.009\%\}$, $[0.01\%, 1\%)$ refers to the quantiles of levels $\{0.01\%, 0.02\%, \ldots, 0.99\%\}$, $[1\%, 99\%]$ refers to the quantiles of levels $\{1\%, 2\%, \ldots, 99\%\}$, $(99\%, 99.99\%]$ refers to the quantiles of levels $\{99.01\%, 99.02\%, \ldots, 99.99\%\}$, and $(99.99\%, 100\%]$ refers to the quantiles of levels $\{99.991\%, 99.992\%, \ldots, 100\%\}$. (a) Min-put with XGBoost: normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ and $V^{\text{now}}_{X,1}$ in %. (b) Min-put with XGBoost: detrended Q-Q plots of $V_{X,1}$ and $V^{\text{now}}_{X,1}$. (c) BRC with XGBoost: normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ and $V^{\text{now}}_{X,1}$ in %. (d) BRC with XGBoost: detrended Q-Q plots of $V_{X,1}$ and $V^{\text{now}}_{X,1}$. (e) Max-call with XGBoost: normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ and $V^{\text{now}}_{X,1}$ in % and (f) Max-call with XGBoost: detrended Q-Q plots of $V_{X,1}$ and $V^{\text{now}}_{X,1}$.

Table A3. Relative errors of value at risk $\mathrm{VaR}_{99.5\%}(\mathrm{L}_X)$, $\mathrm{VaR}_{99.5\%}(\mathrm{L}_X^{\mathrm{now}})$, $\mathrm{VaR}_{99.5\%}(-\mathrm{L}_X)$, $\mathrm{VaR}_{99.5\%}(-\mathrm{L}_X^{\mathrm{now}})$, computed as (estimated VaR minus true VaR)/true VaR using the test sample and expressed in %, using XGBoost and Random Forest, for the payoff functions min-put, BRC, and max-call.

Payoff	Estimator	$\mathrm{VaR}(\mathrm{L}_X)$	$\mathrm{VaR}(\mathrm{L}_X^{\mathrm{now}})$	$\mathrm{VaR}(-\mathrm{L}_X)$	$\mathrm{VaR}(-\mathrm{L}_X^{\mathrm{now}})$
Min-put	XGBoost	**– 9.658%**	50.94%	**– 6.912%**	48.21%
	Random Forest	– 25.69%	**21.38%**	**– 20.57%**	23.30%
	Kernel-based method	***0.9695%***	1.697%	***3.158%***	– 9.184%
BRC	XGBoost	**2.533%**	123.3%	**– 42.85%**	141.8%
	Random Forest	**– 3.510%**	197.6%	– 75.87%	**24.91%**
	Kernel-based method	***0.1893%***	– 16.81%	– 13.91%	***– 7.342%***
Max-call	XGBoost	**– 7.103%**	50.97%	**– 4.140%**	57.89%
	Random Forest	– 23.87%	**21.88%**	**– 20.51%**	40.00%
	Kernel-based method	***0.07143%***	5.357%	– 3.582%	***1.237%***

Table A4. Relative errors of value at risk $\mathrm{ES}_{99\%}(\mathrm{L}_X)$, $\mathrm{ES}_{99\%}(\mathrm{L}_X^{\mathrm{now}})$, $\mathrm{ES}_{99\%}(-\mathrm{L}_X)$, $\mathrm{ES}_{99\%}(-\mathrm{L}_X^{\mathrm{now}})$, computed as (estimated ES minus true ES)/true ES using the test sample and expressed in %, using XGBoost and Random Forest, for the payoff functions min-put, BRC, and max-call.

Payoff	Estimator	$\mathrm{ES}(\mathrm{L}_X)$	$\mathrm{ES}(\mathrm{L}_X^{\mathrm{now}})$	$\mathrm{ES}(-\mathrm{L}_X)$	$\mathrm{ES}(-\mathrm{L}_X^{\mathrm{now}})$
Min-put	XGBoost	**– 10.23%**	49.75%	**– 7.434%**	47.22%
	Random Forest	– 26.41%	**20.46%**	**– 21.07%**	23.68%
	Kernel-based method	***1.261%***	1.775%	***4.769%***	– 8.546%
BRC	XGBoost	**3.940%**	113.5%	**– 43.79%**	151.1%
	Random Forest	**16.93%**	203.1%	– 76.33%	**22.27%**
	Kernel-based method	***– 0.5269%***	– 18.58%	– 14.40%	***– 8.271%***
Max-call	XGBoost	**– 7.808%**	50.12%	**– 4.507%**	55.32%
	Random Forest	– 24.67%	**20.90%**	**– 21.31%**	42.63%
	Kernel-based method	***– 0.3460%***	5.329%	– 3.588%	***0.8112%***

Appendix 2. Bermudan options pricing

So far we showed that ensemble estimators with regression trees can be employed to learn a value process of the form (1). In this section, we sketch how these estimators can also be applied to deal with another important and difficult problem in finance, the pricing of Bermudan options under discrete-time local volatility models.[†] A standard solution to this problem is to recursively estimate the continuation value, see, e.g. Tsitsiklis and van Roy (1999), and Longstaff and Schwartz (2001). Recent solutions based on machine learning techniques include Becker *et al.* (2019), who apply deep neural networks to learn the optimal stopping rule, and Goudenège *et al.* (2020), who apply Gaussian Process Regression to learn the value function at every time step. We add to this literature by showing that ensemble estimators with regression trees give closed-form estimators of the entire value process of Bermudan options.

Similarly to what we discussed in Appendix 1 for European style options pricing, we first apply regress-later and then regress-now.

A.1. *Regress-later*

Assume the stochastic driver $X = (X_1, \dots, X_T)$ has standard normal distribution on $\mathbb{R}^{d\times T}$, i.e. $\mathbb{Q}_t = \mathcal{N}(0, I_d)$ for every $t = 1, \dots, T$. Let $m \in \mathbb{N}$, and $\alpha_t : \mathbb{R}^m \to \mathbb{R}^m$ and $\beta_t : \mathbb{R}^m \to \mathbb{R}^{m\times d}$ be measurable functions, for $t = 0, \dots, T-1$. Let $(Z_t)_{0\le t\le T}$ be an m-dimensional stochastic process that represents the evolution of the log-price of m underlying assets. We assume that it follows the following discrete-time local volatility model,

$$\begin{cases} Z_t = \alpha_{t-1}(Z_{t-1}) + \beta_{t-1}(Z_{t-1})X_t, \quad t = 1, \dots, T, \\ Z_0 = z_0, \end{cases} \tag{A1}$$

where $z_0 \in \mathbb{R}^m$ is the initial log-price vector. We denote by $\mathbb{G} = (\mathcal{G}_t)_{0\le t\le T}$ the natural filtration of Z, $\mathcal{G}_t = \sigma(Z_s \mid 0 \le s \le t)$. Let $g_t : \mathbb{R}^m \to \mathbb{R}$ be measurable functions such that $\mathbb{E}_{\mathbb{Q}}[g_t(Z_t)^2] < \infty$, for $t = 0, \dots, T$. Let $\mathcal{T}_t$ be the set of $\mathbb{G}$-stopping times τ taking values in $\{t, t+1, \dots, T\}$, for $t = 0, \dots, T$. We are interested in the following optimal stopping problem,

$$V_t = \sup\nolimits_{\tau\in\mathcal{T}_t} \mathbb{E}_{\mathbb{Q}}[g_\tau(Z_\tau) \mid \mathcal{G}_t], \quad t = 0, \dots, T. \tag{A2}$$

It is well known, see, e.g. Peskir and Shiryaev (2006, Section 1), problem (A2) can be solved by backward induction as follows.[‡] For time $t = T$, set $V_T = g_T(Z_T)$. By induction, for any time $t = T-1, \dots, 0$, define the continuation value $C_t = \mathbb{E}_{\mathbb{Q}}[V_{t+1}(Z_{t+1}) \mid Z_t]$, so that $V_t = \max(g_t(Z_t), C_t)$. Then, an optimal stopping time $\tau_t^\star \in \mathcal{T}_t$, for which $V_t = \mathbb{E}_{\mathbb{Q}}[g_{\tau_t^\star}(Z_{\tau_t^\star}) \mid \mathcal{G}_t]$, is given by $\tau_t^\star = \inf\{t \le s \le T \mid V_s = g_s(Z_s)\}$.

Our method to solve (A2) is based on a backward induction, where for each time $t = T-1, \dots, 0$ we estimate the value function V_t by an ensemble estimator $V_{X,t}$. Specifically, assume available a finite i.i.d. sample $\boldsymbol{Z} = (Z^{(1)}, \dots, Z^{(n)})$ drawn from (A2). And let $\boldsymbol{Z}_t = (Z_t^{(1)}, \dots, Z_t^{(n)})$ be the t-cross-section sample of $\boldsymbol{Z}$, for $t = 1, \dots, T$. Then proceed backward as follows:

(i) For time $t = T$: set $V_{X,T} = g_T(Z_T)$.
(ii) For any time $t = T-1, \dots, 0$: let $\widehat{V}_{X,t+1}$ be an ensemble estimator of $V_{X,t+1}$, obtained using the sample $\boldsymbol{Z}_{t+1}$, along with the function values $\boldsymbol{V}_{X,t+1} = (V_{X,t+1}(Z_{t+1}^{(1)}), \dots, V_{X,t+1}(Z_{t+1}^{(n)}))$. Then, we claim that

$$C_{X,t} = \mathbb{E}_{\mathbb{Q}}[\widehat{V}_{X,t+1}(Z_{t+1}) \mid Z_t] \text{ is in } \textbf{closed form}. \tag{A3}$$

Then set $V_{X,t} = \max(g_t(Z_t), C_{X,t})$, which is therefore also in closed form.

† For the sake of presentation, we only consider discrete-time local volatility models. However, the method we detail below can be adapted to also deal with discrete-time stochastic volatility models.

‡ We look at Markovian problems, in the sense that g_t depends only on the state variable Z_t. However, our method can be adapted to also deal with non Markovian problems, where g_t could depend on the whole past trajectory $Z_0, \dots, Z_t$.

Now let us explain why $C_{X,t}$ in (A3) is in closed form. The function $\widehat{V}_{X,t+1}$ is an ensemble estimator, whose expression can be brought into the form (5), i.e. $\widehat{V}_{X,t+1} = \sum_{i=1}^{N} \beta_i \mathbb{1}_{A_i}$, for some real coefficients β_i and some hyperrectangles $\boldsymbol{A}_i$ of $\mathbb{R}^m$. Subsequently, by independence of Z_t and X_{t+1}, we readily obtain that

$$C_{X,t}(z_t) = \sum_{i=1}^{N} \beta_i \mathbb{Q}_{t+1}[\alpha_t(z_t) + \beta_t(z_t)X_{t+1} \in \boldsymbol{A}_i], \quad z_t \in \mathbb{R}^m.$$

Now observe that $\alpha_t(z_t) + \beta_t(z_t)X_{t+1}$ is normally distributed under $\mathbb{Q}_{t+1}$, with mean $\alpha_t(z_t)$ and covariance matrix $\beta_t(z_t)\beta_t(z_t)^\top$. Thanks to Genz (2000), $\mathbb{Q}_{t+1}[\alpha_t(z_t) + \beta_t(z_t)X_{t+1} \in \boldsymbol{A}_i]$ is in closed form for any hyperrectangle $\boldsymbol{A}_i$. In fact, functions to integrate a multivariate normal distribution on a hyperrectangle are readily accessible on scientific languages, such as Python (see the mvn function in the subpackage stats of the library SciPy (Virtanen *et al.* 2020)), and R (see the function pmvnorm in the mvtnorm package (Genz *et al.* 2021)). This became possible thanks to the Fortran code of Genz (2022). Matlab code to integrate a multivariate normal distribution on a hyperrectangle can also be found in Genz (2022).

After the construction of the value process estimator V_X, we define the optimal stopping time estimator

$$\tau^\star_{X,t} = \inf\{t \le s \le T \mid V_{X,s} = g_s(Z_s)\}. \tag{A4}$$

Now as a simple numerical example, let S_t represent the nominal price of $m = 1$ underlying asset, whose dynamics is given in (2) with $d = 1$. We denote by $Z_t = \log(S_t)$ its log-price. Then Z_t follows the dynamics

$$\begin{cases} Z_t = Z_{t-1} + (r - \sigma^2/2)\Delta_t + \sigma\sqrt{\Delta_t}X_t, & t = 1, \ldots, T, \\ Z_0 = z_0, \end{cases}$$

which is of the form (A1). We are interested in estimating the value process V in (A2), where the payoff function g_t is

- Put $g_t(Z_t) = \mathrm{e}^{-r\sum_{s=1}^{t}\Delta_s}(K - \mathrm{e}^{Z_t})^+$.

Here we set the following parameter values, $z_0 = 0$, $r = 0$, $\sigma = 0.2$, $T = 7$, $(\Delta_1, \ldots, \Delta_T) = (1/T, \ldots, 1/T)$, and $K = 1$. Under this parameter specification, we generate a training sample $\boldsymbol{X}$ of size $n = 5000$, and a test sample $\boldsymbol{X}_{\text{test}}$ of size $n_{\text{test}} = 100\,000$. When V_X is the Random Forest estimator of V, we use the RandomForestRegressor class of the library scikit-learn (Pedregosa *et al.* 2011), with the following hyperparameter values: M=10, **nodesize** = 2, $p = 1$, **sampling regime**=bootstrapping.† When V_X is the Gradient Boosting estimator of V, we use the XGBRegressor class of XGBoost (Chen and Guestrin 2016), with the default hyperparameter values: $t = 100$, **nodesize** = 1, **max_depth** = 6.

With the Bermudan put example and the parameter specification, $r = 0$, one can readily show that early stopping is not optimal, so that $V_t = \mathbb{E}[(K - \mathrm{e}^{Z_T})^+ \mid Z_t]$ equals the European put option price, and V_t is given in closed form thanks to Black's formula. In fact, for $t = 0, \ldots, T-1$, $V_t = -\mathrm{e}^{Z_t}\Phi(-d_1) + K\Phi(-d_2)$, where $d_1 = \frac{1}{\sigma\sqrt{\Delta_{t+1}+\cdots+\Delta_T}}(\ln(\mathrm{e}^{Z_t}/K) + (r+\sigma^2/2)(\Delta_{t+1} + \cdots + \Delta_T))$ and $d_2 = d_1 - \sigma\sqrt{\Delta_{t+1} + \cdots + \Delta_T}$. We thus have exact ground truth benchmark for our estimator V_X.

After the construction of the estimated value process V_X, we evaluate $V_{X,t}$ and V_t on the test sample $\boldsymbol{X}_{\text{test}}$, for $t = 0, \ldots, T-1$. Then we carry out the following four evaluation tasks.

First, we compute the normalized $L^2_{\mathbb{Q}}$-error $\|V_{X,t} - V_t\|_{2,\mathbb{Q}}/V_0$, for $t = 0, \ldots, T-1$. Figure A3(a,b) show the evolution of the normalized $L^2_{\mathbb{Q}}$-error of $V_{X,t}$ as function of $t = 0, \ldots, T-1$. First, we notice that all normalized $L^2_{\mathbb{Q}}$-errors are below 0.2% and 0.6% with Random Forest and XGBoost, respectively. Second, with the exception of $t = 0$, the normalized $L^2_{\mathbb{Q}}$-errors have a tendency to increase with time to maturity $T - t$. There seems to be an accumulation of errors, due to the estimation of V_t at each induction step $t + 1 \to t$. Whereas at $t = 0$ the errors seem to cancel out across the sample, as $V_{X,0} = C_{X,0}$ is given by the unconditional expectation (A3).

Second, we compute the detrended Q-Q plots of V_X. Figure A3(c,d) show detrended Q-Q plots of $V_{X,t}$, for $t = 1, \ldots, T-1$, using Random Forest and XGBoost, respectively. They are constructed as the detrended Q-Q plots in Section 4. Specifically, here we draw the detrended Q-Q plots of $V_{X,t}$ using the test sample, for every $t = 1, \ldots, T-1$. Thereto, for $t \in \{1, \ldots, T-1\}$, we compute the empirical left quantiles of $V_{X,t}$ and V_t at levels $\{0.001\%, 1\%, 2\%, \ldots, 100\%\}$. The detrended quantiles (estimated quantiles minus true quantiles) are then plotted against the true quantiles. We notice that, as function of t, the decrease of the normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,t}$ translates into a flattening of the detrended Q-Q plots. In fact, for illustration, the almost zero normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,4}$, $V_{X,5}$, and $V_{X,6}$ in figure A3(a) correspond to almost perfect detrended Q-Q plots of $V_{X,4}$, $V_{X,5}$, and $V_{X,6}$ in figure A3(c), i.e. they correspond to detrended Q-Q plots that are almost perfectly aligned with the horizontal black line. Overall, both Random Forest and XGBoost give excellent detrended Q-Q plots of $V_{X,t}$, for every $t = 1, \ldots, T-1$.

Third, we use our value process estimator V_X to compute the value at risk at level $\alpha = 99.5\%$, and the expected shortfall at level $\alpha = 99\%$ of $V_{X,t} - V_{X,t+1}$ and $V_{X,t+1} - V_{X,t}$, for every $t = 0, \ldots, T-1$. $V_{X,t} - V_{X,t+1}$ and $V_{X,t+1} - V_{X,t}$ are the 1-period losses at time t of long position and short position, respectively. We perform the same risk measure computations with our benchmark V. Then, we compute the relative errors of risk measures, computed as (estimated risk measure minus true risk measure)/true risk measure and expressed in %. Figure A3(e) shows the evolution of relative errors of risk measures $\text{ES}_{99\%}$ and $\text{VaR}_{99.5\%}$ for both long and short positions with Random Forest. Figure A3(f) shows the same computations with XGBoost. Let's focus on figure A3(e). Relative errors of risk measures of long and short positions are all in the intervals $[-0.3\%, 0\%]$ and $[-1.2\%, 0.1\%]$, respectively. Furthermore, we highlight that the relative errors for $t \in \{3, 4, 5, 6\}$ are equal to 0%. The last is in line with the detrended Q-Q plots at $t \in \{3, 4, 5, 6\}$ in figure A3(c), which are almost perfectly aligned with the horizontal black line.

Fourth, we compute the optimal stopping rule estimator $\tau^\star_{X,0}$ in (A4) for the n_{test} simulations in $\boldsymbol{X}_{\text{test}}$. Table A5 shows the distribution of $\tau^\star_{X,0}$ using the test sample $\boldsymbol{X}_{\text{test}}$. The distribution of the true optimal stopping rule $\tau^\star_0$ is the Dirac distribution $\delta_7(dx)$. We observe that estimation of the distribution of $\tau^\star_0$ is accurate with both Random Forest and XGBoost, and it is XGBoost that outperforms Random Forest.

A.2. Regress-now

We now compare the above regress-later method for Bermudan options pricing to its regress-now variant, as discussed in Appendix 1 for European style options.

Let us start by introducing the regress-now estimator of the value process of Bermudan options. We shall denote this estimator by V^{now}_X. Its construction is as follows. To solve (A2), proceed backward, where at each time $t = T-1, \ldots, 0$ estimate the continuation value function C_t by an ensemble estimator $C^{\text{now}}_{X,t}$. Specifically, assume available a finite i.i.d. training sample $\boldsymbol{Z} = (Z^{(1)}, \ldots, Z^{(n)})$ drawn from (A2). And let $\boldsymbol{Z}_t = (Z^{(1)}_t, \ldots, Z^{(n)}_t)$ denote the t-cross-section samples of $\boldsymbol{Z}$. Then proceed backward as follows:

(i) For time $t = T$: set $V^{\text{now}}_{X,T} = g_T(Z_Z)$.
(ii) For any time $t = T-1, \ldots, 0$: let $C^{\text{now}}_{X,t}$ be an ensemble estimator of C_t, obtained using the sample $\boldsymbol{Z}_t$, along with the function values $\boldsymbol{V}^{\text{now}}_{X,t+1} = (V^{\text{now}}_{X,t+1}(Z^{(1)}_{t+1}), \ldots, V^{\text{now}}_{X,t+1}(Z^{(n)}_{t+1}))$. Then, we set $V^{\text{now}}_{X,t} = \max(g_t(Z_t), C^{\text{now}}_{X,t})$.

When V^{now}_X is the Random Forest estimator of V, we use the RandomForestRegressor class of the library scikit-learn (Pedregosa *et al.* 2011), with the hyperparameter values: M=500, **nodesize** =

† Except for the number of trees M, the other hyperparameter values are the default values in RandomForestRegressor. We picked a small value for M for computational reasons.

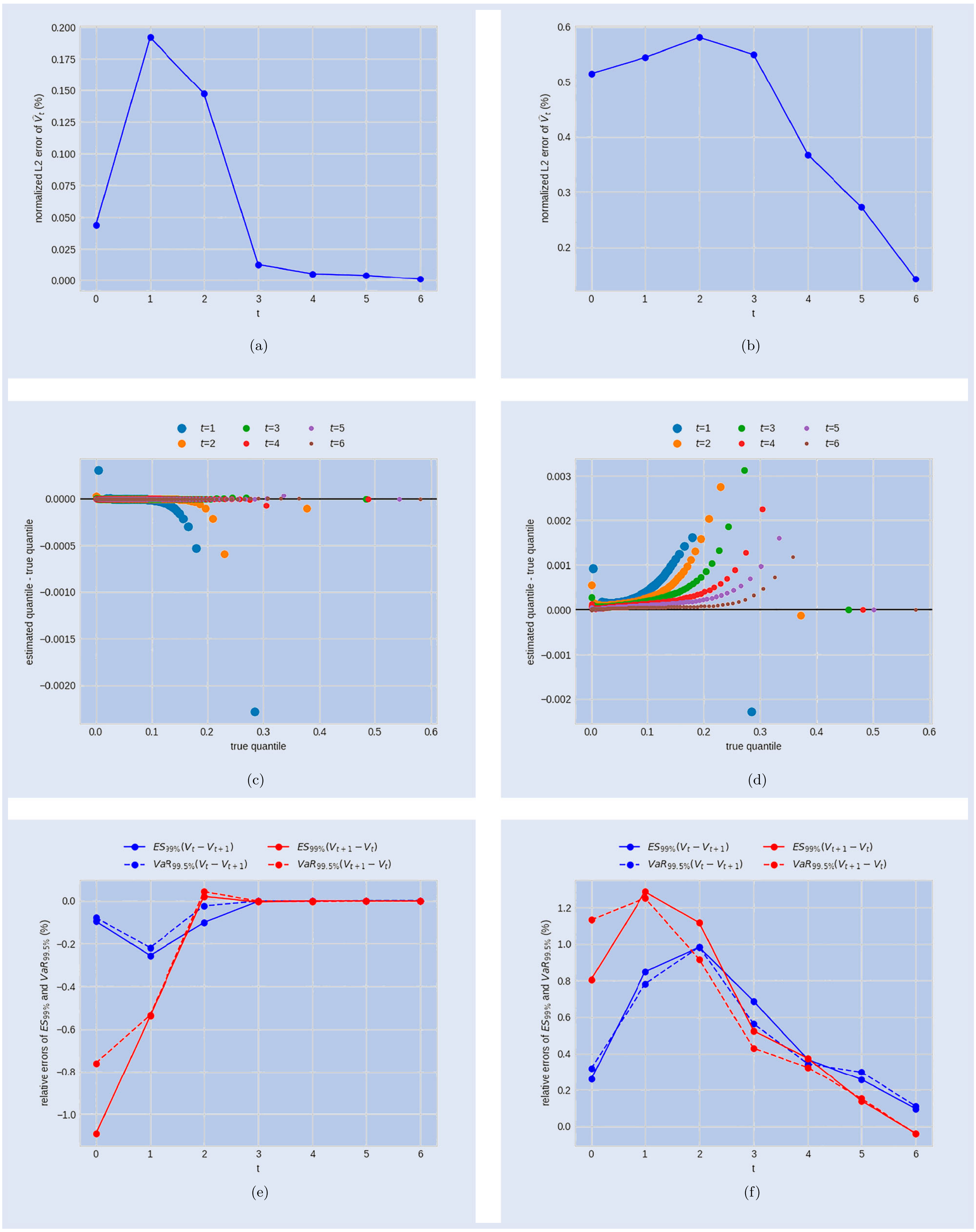

Figure A3. Results for the Bermudan put with regress-later Random Forest and regress-later XGBoost. The normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,t}$, $\|V_t - V_{X,t}\|_{2,\mathbb{Q}}/V_0$, for $t = 0, \ldots, T-1$, are computed using the test sample and expressed in %. The detrended Q-Q plots are built using the test sample. They show the detrended quantiles of levels $\{0.001\%, 1\%, 2\%, \ldots, 100\%\}$ of $V_{X,t}$, for $t = 1\ldots, T-1$. The relative errors of value at risk and expected shortfall, computed as (estimated risk measure minus true risk measure)/true risk measure using the test sample, are expressed in %. (a) With Random Forest: normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,t}$ as function of $t = 0, \ldots, T-1$. Values are expressed in %. (b) With XGBoost: normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,t}$ as function of $t = 0, \ldots, T-1$. Values are expressed in %. (c) With Random Forest: detrended Q-Q plot of $V_{X,t}$ for $t = 1, \ldots, T-1$. (d) With XGBoost: detrended Q-Q plot of $V_{X,t}$ for $t = 1, \ldots, T-1$. (e) With Random Forest: relative errors of $\mathrm{ES}_{99\%}(V_t - V_{t+1})$, $\mathrm{VaR}_{99.5\%}(V_t - V_{t+1})$, $\mathrm{ES}_{99\%}(V_{t+1} - V_t)$, and $\mathrm{VaR}_{99.5\%}(V_{t+1} - V_t)$ for the estimator V_X. Values are expressed in % and (f) With XGBoost: relative errors of $\mathrm{ES}_{99\%}(V_t - V_{t+1})$, $\mathrm{VaR}_{99.5\%}(V_t - V_{t+1})$, $\mathrm{ES}_{99\%}(V_{t+1} - V_t)$, and $\mathrm{VaR}_{99.5\%}(V_{t+1} - V_t)$ for the estimator V_X. Values are expressed in %.

Table A5. Distribution of $\tau^{\star}_{X,0}$, constructed with V_X using the test sample $\boldsymbol{X}_{\text{test}}$, using XGBoost and Random Forest.

Estimator | time t	0	1	2	3	4	5	6	7
XGBoost	0	0.00009	0.00037	0.00036	0.00064	0.00030	0.00104	***0.99720***
Random Forest	0	0.00026	0.00308	0.00197	0.00693	0.00097	0.00534	**0.98145**

Note: The true distribution of $\tau^{\star}_0$ is the Dirac distribution $\delta_7(dx)$.

Table A6. Distribution of $\tau^{\star,\text{now}}_{X,0}$, constructed with V^{now}_X using the test sample $\boldsymbol{X}_{\text{test}}$, using XGBoost and Random Forest.

Estimator | time t	0	1	2	3	4	5	6	7
XGBoost	0	0.00235	0.00803	0.004043	0.05611	0.07934	0.12134	***0.69240***
Random Forest	0	0.00320	0.0077	0.03035	0.05756	0.06859	0.27567	**0.56386**

Note: The true distribution of $\tau^{\star}_0$ is the Dirac distribution $\delta_7(dx)$.

20, $p = 1$, **sampling regime**=bootstrapping. When V_X is the Gradient Boosting estimator of V, we use the XGBRegressor class of XGBoost (Chen and Guestrin 2016), with the hyperparameter values: $t = 300$, **nodesize** $= 20$, **max_depth** $= 50$.†

Just as in Section A.1 above, we use the test sample $\boldsymbol{X}_{\text{test}}$ to compute the normalized $L^2_{\mathbb{Q}}$-error $\|V^{\text{now}}_{X,t} - V_t\|_{2,\mathbb{Q}}/V_0$, for $t = 0, \ldots, T-1$. Figure A4(a,b) show the evolution of the normalized $L^2_{\mathbb{Q}}$-error of $V^{\text{now}}_{X,t}$ as function of $t = 0, \ldots, T-1$. First, we notice that all normalized $L^2_{\mathbb{Q}}$-errors are below 10% and 11% with Random Forest and XGBoost, respectively. Second, with the exception of $t = 0$, the normalized $L^2_{\mathbb{Q}}$-errors have a tendency to increase with time to maturity $T - t$. Again, there seems to be an accumulation of errors, due to the estimation of C_t at each induction step $t + 1 \to t$. Whereas at $t = 0$ the errors seem to cancel out across the sample, as $V_{X,0} = C_{X,0}$ is given by the unconditional expectation (A3). Third, by comparing figures A4(a,b) to A3(a,b), we highlight the outperformance of our regress-later method over its regress-now variant in terms of normalized $L^2_{\mathbb{Q}}$-errors.

Then we compute the detrended Q-Q plots of V^{now}_X. Figure A4(c,d) show detrended Q-Q plots of $V^{\text{now}}_{X,t}$, for $t = 1, \ldots, T-1$. They are the counterpart of the detrended Q-Q plots of V_X in figure A3(c,d). By comparing these four figures, we see that the detrended Q-Q plots are of much better quality with regress-later than with regress-now.

Next, we use our value process estimator V^{now}_X to compute the value at risk at level $\alpha = 99.5\%$, and expected shortfall at level $\alpha = 99\%$ of $V^{\text{now}}_{X,t} - V^{\text{now}}_{X,t+1}$ and $V^{\text{now}}_{X,t+1} - V^{\text{now}}_{X,t}$, for $t = 0, \ldots, T-1$. Figure A4(e,f) show relative errors of risk measures with V^{now}_X. They are the counterpart of figure A3(e,f). By comparing these four figures, we see the outperformance of regress-later over regress-now in terms of relative errors of risk measures.

We finish this section by computing the optimal stopping rule estimator $\tau^{\star,\text{now}}_{X,0} = \inf\{0 \le s \le T \mid V^{\text{now}}_{X,s} = g_t(Z_s)\}$ for the n_{test} simulations in $\boldsymbol{X}_{\text{test}}$. Table A6 is the counterpart of table A5. Here again we see the outperformance of regress-later over regress-now in terms of accuracy in the estimation of the optimal stopping rule distribution.

Appendix 3. Correlated stock returns

In Section 4, we assumed independent stock returns for our numerical examples. As a robustness check, we have performed some computations under the assumption of correlated stock returns, which are similar to those in Section 4.

Concretely, from (3), we see that the conditional correlation between the log returns $\log \frac{S_{i,t}}{S_{i,t-1}}$ and $\log \frac{S_{j,t}}{S_{j,t-1}}$ is given by $\rho_{i,j,t} = \frac{\sigma_i^{\top}\sigma_j}{\|\sigma_i\|\|\sigma_j\|}$. For the min-put example we chose a constant negative correlation of $\rho_{i,j,t} = -0.2$. For the barrier reverse convertible example we chose a constant positive correlation of $\rho_{i,j,t} = 0.25$. For the max-call example we tested both choices.

The results are as follows. For the min-put with correlation $\rho_{i,j,t} = -0.2$, with Random Forest and XGBoost the normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ are 2.723% and 1.077%, respectively. For the barrier reverse convertible with correlation $\rho_{i,j,t} = 0.25$, with Random Forest and XGBoost the normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ are 0.441% and 0.243%, respectively. For the max-call with correlation $\rho_{i,j,t} = -0.2$, with Random Forest and XGBoost the normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ are 4.196% and 1.733%, respectively. For the max-call with correlation $\rho_{i,j,t} = 0.25$, with Random Forest and XGBoost the normalized $L^2_{\mathbb{Q}}$-errors of $V_{X,1}$ are 8.851% and 2.765%, respectively. These values correspond to those in table 3, where the correlation is equal to 0. We see that the values are comparable. Our approach also delivers satisfactory results in terms of $L^2_{\mathbb{Q}}$-errors for correlated stock returns.

† These hyperparameter values give much better numerical results than the default hyperparameter values we used in Section A.1.

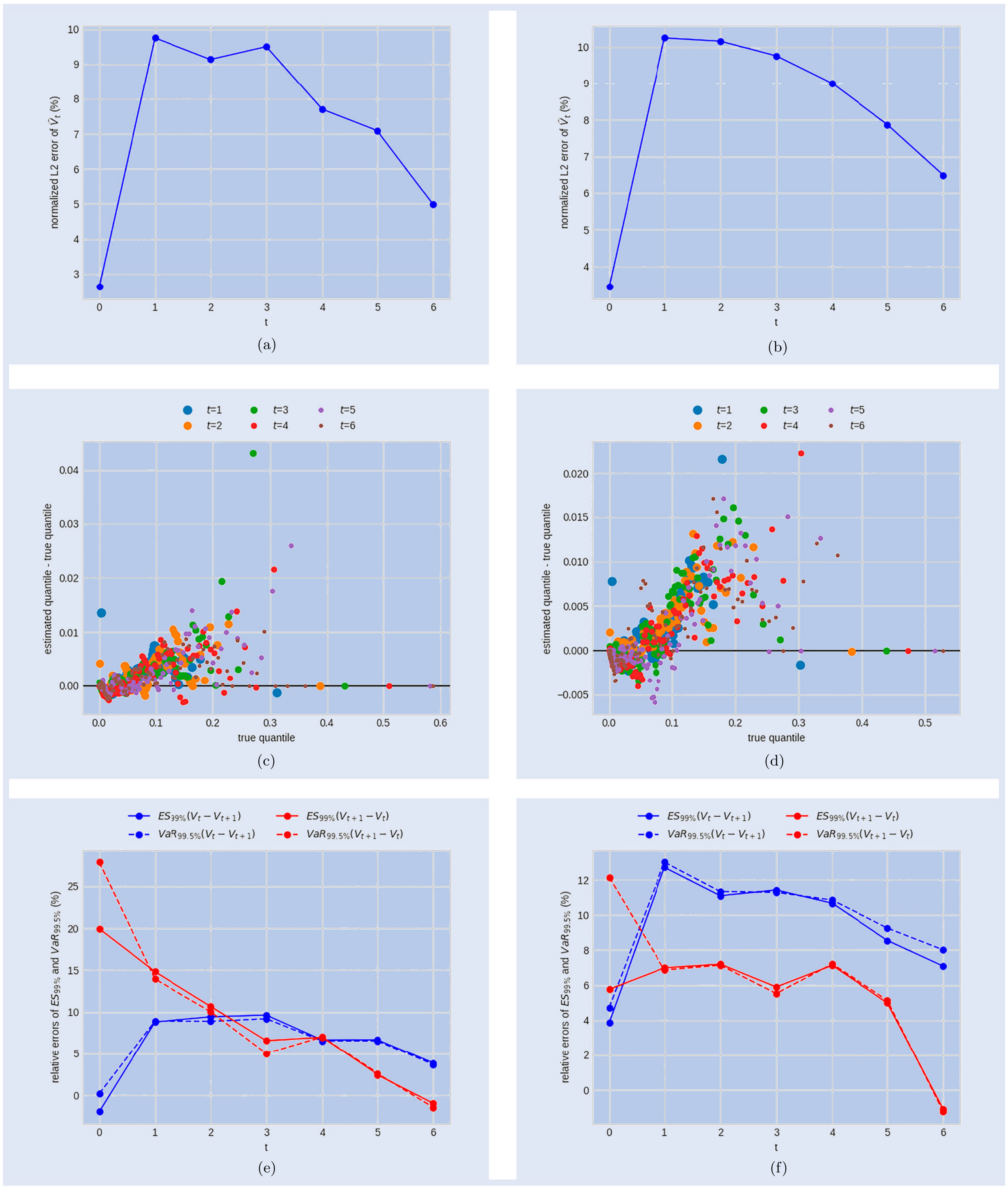

Figure A4. Results for the Bermudan put with regress-now Random Forest and regress-now XGBoost. The normalized $L^2_{\mathbb{Q}}$-errors of $V^{\text{now}}_{X,t}$, $\|V_t - V^{\text{now}}_{X,t}\|_{2,\mathbb{Q}}/V_0$, for $t = 0,\dots,T-1$, are computed using the test sample and expressed in %. The detrended Q-Q plots are built using the test sample. They show the detrended quantiles of levels $\{0.001\%, 1\%, 2\%, \dots, 100\%\}$ of $V^{\text{now}}_{X,t}$, for $t = 1,\dots,T-1$. The relative errors of value at risk and expected shortfall, computed as (estimated risk measure minus true risk measure)/true risk measure using the test sample, are expressed in %. (a) With Random Forest: normalized $L^2_{\mathbb{Q}}$-errors of $V^{\text{now}}_{X,t}$ as function of $t = 0,\dots,T-1$. Values are expressed in %. (b) With XGBoost: normalized $L^2_{\mathbb{Q}}$-errors of $V^{\text{now}}_{X,t}$ as function of $t = 0,\dots,T-1$. Values are expressed in %. (c) With Random Forest: detrended Q-Q plot of $V^{\text{now}}_{X,t}$ for $t = 1,\dots,T-1$. (d) With XGBoost: detrended Q-Q plot of $V^{\text{now}}_{X,t}$ for $t = 1,\dots,T-1$. (e) With Random Forest: relative errors of $\text{ES}_{99\%}(V_t - V_{t+1})$, $\text{VaR}_{99.5\%}(V_t - V_{t+1})$, $\text{ES}_{99\%}(V_{t+1} - V_t)$, and $\text{VaR}_{99.5\%}(V_{t+1} - V_t)$ for the estimator V^{now}_{X}. Values are expressed in % and (f) With XGBoost: relative errors of $\text{ES}_{99\%}(V_t - V_{t+1})$, $\text{VaR}_{99.5\%}(V_t - V_{t+1})$, $\text{ES}_{99\%}(V_{t+1} - V_t)$, and $\text{VaR}_{99.5\%}(V_{t+1} - V_t)$ for the estimator V^{now}_{X}. Values are expressed in %.