Received: 31 October 2019 | Revised: 29 December 2020 | Accepted: 30 December 2020
DOI: 10.1002/asmb.2607

SPECIAL ISSUE PAPER

WILEY

Machine learning techniques in nested stochastic simulations for life insurance

Gilberto Castellani[1] | **Ugo Fiore**[2] | **Zelda Marino**[2] | **Luca Passalacqua**[1] | **Francesca Perla**[2] | **Salvatore Scognamiglio**[2] | **Paolo Zanetti**[2]

[1]Department of Statistical Sciences, Sapienza University, Rome, Italy

[2]Department of Management Studies and Quantitative Methods, Parthenope University, Naples, Italy

Correspondence
Francesca Perla, Department of Management Studies and Quantitative Methods Parthenope University, Naples, Italy.
Email: francesca.perla@uniparthenope.it

Abstract
The insurance regulatory regime introduced in the European Union by the "Solvency II" Directive 2009/138, that has become applicable on 1 January 2016, is aimed to safeguard policyholders and beneficiaries by requiring insurance undertakings to hold own funds able to cover losses, in excess to the expected ones, at the 99.5% confidence level, over a 1-year period. In order to assess risks and evaluate the regulatory Solvency Capital Requirement, undertakings should compute the probability distribution of the Net Asset Value over a 1-year period, with a financially inspired *market consistent* approach. In life insurance, given the peculiarities of the contracts, the valuation of the Net Asset Value distribution requires a *nested Monte Carlo* simulation, which is extremely time-consuming. Machine learning techniques are considered a promising candidate to reduce the computational burden of nested simulations. This work investigates the potential of well-established methods, such as *deep learning networks* and *support vector regressors*, when applied to the valuation of the Solvency Capital Requirement of *participating life insurance policies*, by empirically assessing their effectiveness and by comparing their efficiency and accuracy, also w.r.t. the "traditional" *least squares Monte Carlo* technique. The work aims also to contribute to the global process of renewal of the European insurance industry, where Solvency II has made the board of directors fully responsible of the choice of evaluation techniques and algorithmic processes, under the periodic monitoring of national supervisory authorities.

KEYWORDS
deep learning, least squares Monte Carlo, Solvency II nested Monte Carlo, support vector regression, with-profit insurance policies

1 | INTRODUCTION

This work originates from the deep transformation that the insurance industry is presently undergoing, following the adoption of European Union Directive 2009/138/EC—better known as "Solvency II"[1]—that has become applicable on 1 January 2016. Solvency II is a complex system of regulation, aimed at defining a harmonised and risk-based framework, common to all Member States, establishing the conditions and the procedures for the granting of authorisation (as well as for any refusal) for the "taking-up and pursuit of the business of Insurance and Reinsurance". The insurance sector is peculiar under many points of view: it covers an extremely broad range of risks (accidents, natural and man-made catastrophes, human lifetime, financial risks, …); it is an activity where the production cycle is inverted: premiums precede the pay-outs, thus requiring financial control; life insurance has a typical time horizon of many decades; it is historically one of the major institutional buyers of governments debt. The main objective of the Directive is, then, the adequate protection of policy holders and beneficiaries, and this requires insurance and reinsurance undertakings to be subject to effective solvency requirements.

Solvency is the state of having assets that are worth more than the liabilities; the question of "how solvent" an insurance company is at present, or will be on a specific time horizon, can be answered only in a probabilistic framework, since solvency-insolvency is a two-state space. In this aspect, Solvency II is a major revolution in the insurance regulatory framework, since probabilities (and stochastic models) enter officially in the legal and in the balance sheet frameworks. To be more precise, one of the most important requirements of Solvency II is that insurance and reinsurance undertakings should hold own funds able to cover losses, in excess to expected ones, at the 99.5% confidence level over 1-year time horizon. This means that, to determine their *Solvency Capital Requirement* (SCR), insurance and reinsurance undertakings should compute the probability distribution of the Net Asset Value (NAV), that is value of assets minus value of liabilities, over 1-year time horizon. Solvency II is based on a principle of *market consistency*, in that assets and liabilities should "be valued at the amount for which they could be exchanged between knowledgeable willing parties in an arm's length transaction" (article 75), and "the calculation of technical provisions shall make use of and be consistent with information provided by the financial markets and generally available data on underwriting risks" (article 76). The *market consistent* approach cannot however be applied directly to illiquid contracts such as insurance policies; in other words while insurance assets can be (mostly) valued following this approach, insurance liabilities should be valued *mark-to-model*, that is according to sound valuation principles, similar to those applied in the financial markets.

Nowadays, the standard approach for the valuation of financial contracts is the so-called *risk-neutral* (opposed to *real-world*) probability measure approach, which is a consequence of the request of the absence of arbitrage opportunities on the market (heuristically, an arbitrage is an investment strategy able to produce a strictly positive gain starting from a zero initial capital). Technically speaking, this means that the price of a contract can be computed as an expected value, conditional to the available information, of a cash flow which in turn depends on a (typically) multidimensional random process defined in a proper filtered probability space. By definition, the expectation under the risk-neutral measures accounts for the market price of risk that investors assign to (risky) contingent claims. Being computationally very efficient, the Monte Carlo technique is used for the evaluation of the expectation (typically, an integral) whenever closed form solutions are not available, or the dimension of the random process is not sufficiently small[2] to use quadrature techniques.

In the Solvency II framework, the use of the *market consistent* approach in the valuation of the NAV over 1-year time horizon implies that future prices (both of assets and liabilities) should be computed conditional to the information that *will be available* in the future (in 1 year), that is each Monte Carlo prices valuation, according to the *risk-neutral* probability measure, should be repeated for each possible scenario that can be realised over 1-year time-horizon, according to the *real-world* probability measure. All in one, this means that Solvency II capital requirements on insurance and reinsurance undertakings need—in principle—a *nested Monte Carlo* valuation,[3] which is obviously very time-consuming. To cope with such a complexity, Solvency II allows insurance and reinsurance undertakings to compute capital requirements either using a *standard formula* setting, which simplifies the calculation of the NAV distribution according to a set of hypotheses specified in the Directive, or using an *internal model* specified by the undertaking, subject to the approval of the national supervisory authority. In fact, within the internal model approach, the freedom of the methodological and procedural choices, with regard to the structure and the calculation processes to be used, is extremely high, specifically "no particular method for the calculation of the probability distribution forecast shall be prescribed" (article 121). However, the undertaking—more precisely the board of directors—has full responsibility for evaluation techniques, algorithmic processes, probability distributions and correlations[1]. In this framework, the Directive assigns to the regulator only the task of validating the model used by the insurance company and of verifying that it has full control over the methods employed.

The present work investigates technical solutions relevant for the evaluation of the solvency requirement by using an internal model in the life sector. It is worth mentioning that participating life policies—on which this work is

[1]Salvatore Rossi—President of the Italian supervisory authority (IVASS)—in the opening lecture of the first conference on Solvency II in 2016—noted that "in a risk-dominated environment, supervisors are required to systematically check companies, verify their risk attitude and control how risks are identified, measured and managed; [...]. In our experience, not all board members are always ready and able to support constructive dialogue with the supervisory authority, particularly in smaller companies. This, however, is an essential element in the new regulatory framework. We have a *cultural gap* to fill and we have to do it together".

focusing—are very complex derivatives, not only because their values are affected by a large number of risk drivers, but also because the value also depends on the (dynamical) strategies used by insurance companies to invest their assets, which remain undisclosed for self-evident reasons. From this point of view, the investigation benefits from the large experience of some of the authors in the field of valuation and modelling of life policies, and from their knowledge of the *best practices* used in the insurance sector.

At present, given the number of life insurance contracts to be evaluated, the complexity of their payoffs, the large number of risk drivers and the availability of computing power, the nested Monte Carlo approach cannot be pursued by *brute force*,[2] at least in computing times suited for the management of the insurance companies (even with high-level commercial cloud solutions of several hundreds of processors). Therefore, different strategies have been proposed in the literature in order to reduce the computational burden. Among these, the *least squares Monte Carlo* (LSMC)[4,5] (discussed in detail in the following) seems to be the most promising one. In short, the least squares Monte Carlo techniques is a two-step procedure that requires, first to create a set of future prices using the nested Monte Carlo approach, albeit with a reduced number of Monte Carlo simulations. This set of prices is then used to estimate the parameters of a closed form expression approximating the dependence of future prices on the (future) values of the set of risk drivers. Note that this relationship is ex ante unknown, given the complexity of the stochastic processes involved and is numerically inferred by regression. The closed form expression is then used in a second plain (no nested) Monte Carlo simulation where the evolution of the risk drivers is simulated, while future prices are computed analytically using the relationship derived in the first step of the procedure. As such, the least squares Monte Carlo approach is based on the solution of a typical machine learning regression problem. At present, to the best of our knowledge, insurance companies, using the LSMC method, choose the closed form approximation by truncating an orthogonal polynomials series expansion. However, it is not clear that such approach remains advantageous with respect to other machine learning techniques when the dimension of the space of variables grows, that is as the number of risk driver increases.

Very recently, following applications in other disciplines, machine learning techniques have been advocated to solve insurance problems. Despite their recent application in the finance and insurance industry,[6-8] they are quickly becoming popular[9-11] in addressing the new paradigm of the "high-tech business world".[12] Concerning the problem addressed in this article, the available literature is relatively poor.

A recent paper has investigated the potential of various machine learning regression approaches (including generalised linear model, generalised additive model, kernel regression and others) to replace the traditional polynomial regression in a least squares Monte Carlo setting.[13] Another interesting work has compared a more sophisticated algorithm, artificial neural network, against the standard least square Monte Carlo to estimate the portfolio value and the corresponding SCR. Numerical experiments, carried out on financial derivative portfolios (containing European-style call and put options and compound ratchet equity indexed annuity), highlighted that a feed-forward neural network with multiple hidden layers can produce more accurate results.[14] In another recent paper, working on variable annuities portfolios, a neural network implementation of the spatial interpolation technique has been proposed to promote the efficiency valuations of these products in a large portfolio setting.[15] In this scheme, a small set of representative variable annuities contracts is selected and valued via Monte Carlo simulations. In a second stage, the value of all contracts in the input portfolio was obtained as a linear combination of the values of the representative contracts. In a more recent paper, the same authors apply the spatial interpolation approach within the nested simulation for the liabilities value calculation and SCR estimation.[16]

These first experiments, presently still taking place, confirm the necessity and convenience of addressing specific problems in a tailored way (since only by working on the details of the "case" it is possible to gain awareness and decide on the adequacy and feasibility); further, they demonstrate the potential of new techniques to improve on traditional settings, although requiring new discipline in data collection and analysis; finally, they also show that the identification of the "optimal' technique is still at a very preliminary stage.

In this framework, this work contributes to the recent literature investigating the potential of machine learning in insurance by developing a data-driven procedure for the SCR calculation in life insurance in a realistic context. In particular, two alternative techniques have been investigated: the first one uses a regression version of the support vector machines, known as *support vector regression* (SVR), the second is based on *deep learning networks* (DLN), a type of neural networks with a large number of hidden layers. Numerical experiments were carried out on some realistic insurance portfolios to validate the proposed procedure and a comparative study against the LSMC method, discussing pros and cons of each approach, was performed.

This article is structured as follows. In Section 2 a rigorous yet accessible summary on the problem being targeted and on nested Monte Carlo simulation is provided. In Section 3 the contractual structure of participating life insurance policies is described, while Section 4 outlines the methods applied in the work. In Section 5 details of the experimental methodology and the judgemental metrics are given, and the corresponding results are presented and discussed. Finally, Section 6 highlights the main contributions and directions for further study.

2 | BACKGROUND

Solvency II requires insurance and reinsurance undertakings to hold own funds covering the sum of (a) the value of the technical provision, and (b) the value of the *SCR*. Loosely speaking the value of the technical provisions is the value of liabilities generated by the insurance activities, while the *SCR* is a buffer that should ensure that the undertaking remains solvent in a 1-year period at the 99.5% confidence level.

Let t be the valuation time, so that $t+1$ is the 1-year ahead time, let A_{t+1} and L_{t+1} the value of assets and liabilities at time $t+1$, respectively; at time t the *NAV* in $t+1$, $N_{t+1}=A_{t+1}-L_{t+1}$, is (obviously) a random variable. Given some confidence level $\alpha \in (0,1)$ the "worst case loss" at the confidence level α is the *Value-at-Risk*—VaR_α—given by the smallest number ℓ such that the probability that the loss $\Lambda_{t+1} := -N_{t+1}$ exceeds ℓ is no larger than $(1-\alpha)$. Formally,

$$\text{VaR}_\alpha = \inf\{\ell \in \mathbb{R} : \mathbf{P}[\Lambda_{t+1} > \ell] \leq 1-\alpha\} = \inf\{\ell \in \mathbb{R} : F_{\Lambda_{t+1}}(\ell) \geq \alpha\}, \tag{1}$$

where $F_{\Lambda_{t+1}}$ is the distribution function of Λ_{t+1}. The difference between the worst case loss and the expected loss

$$\text{UL}_\alpha = \text{VaR}_\alpha - \mathbf{E}[\Lambda_{t+1}], \tag{2}$$

is the buffer that the undertaking shall have available in $t+1$ to face *unexpected losses*, and thus to ensure solvability over the 1-year period at the $\alpha = 99.5\%$ confidence level; the *SCR* is its present value, that is the amount of money to be held in t, in order to have the proper buffer in $t+1$:

$$\text{SCR}(t, t+1) = \text{UL}_\alpha \; v(t, t+1), \tag{3}$$

where $v(t,t+1)$ is the risk-free appropriate discount factor prevailing on the financial market at time t for the maturity $t+1$.

The distribution function of the loss Λ_{t+1}, or—equivalently—of the net asset value N_{t+1}, receives special attention in the Directive, since it is the prerequisite for the valuation of the *SCR*. In Solvency II jargon the probability density of the *NAV* is called the *Probability Distribution Forecast* (*PDF*), and is defined by art. 13: "probability distribution forecast means a mathematical function that assigns to an exhaustive set of mutually exclusive future events a probability of realisation"; while art. 122 states that "where practicable" the *SCR* should be evaluated "directly from the probability distribution forecast generated by the internal model" using the Value-at-Risk approach at a confidence level of 99.5%, over a 1-year period".

The probability distribution that is used to calculate the *SCR* represents the undertaking expectations about the future; in this sense it is said that the probabilities of future events are "real-world" (or "natural") probabilities. Technically speaking, one must refer to a proper probabilistic framework defined by a filtered probability space $(\Omega, \mathcal{F}, \{\mathcal{F}\}_{t\geq 0}, \mathbb{P})$, where Ω is the set of events, $\mathcal{F}$ is the sigma-algebra defining the measurable events, $\{\mathcal{F}_t\}_{t\geq 0}$ is the filtration defining the time flow of information, and $\mathbb{P} : \mathcal{F} \to [0,1]$ is the real-world probability measure. Expectations are computed conditional on the available information, that is conditional on the current value of the filtration.

The determination of N_{t+1} requires the joint valuation of assets and liabilities. According to financial theory, the price of a contract can be determined by the knowledge of (a) the cash flows it generates, (b) the knowledge of the *risk-neutral* probability measure $\mathbb{Q}$ that allows to determine prices as expectations of the cash flows. However, the cash flow generated by assets and liabilities depends in turn on the value of other random variables, the so-called "risk drivers", that are of financial nature, as interest rates or stock prices, or of actuarial nature, such as human survival or the decision to lapse or surrender a contract.

The risk drivers are the elementary sources of uncertainty. Since the determination of future cash flows demands the description of the time evolution of the risk drivers, the valuation of assets and liabilities requires to define a set of joint stochastic differential equations for the risk drivers. In most internal models the vast majority of risk drivers are assumed to follow Itô diffusion processes, with the others belonging to the enlarged class of Lévy processes, either of the jump-diffusion or of the pure jump type. The structure of dependence between risk drivers is another important aspect of the theoretical framework, since it can produce either a shrinking of the *NAV* distribution (the so called "diversification effect") or enhance the tail of the *NAV* distribution so that "catastrophic" events become more probable.

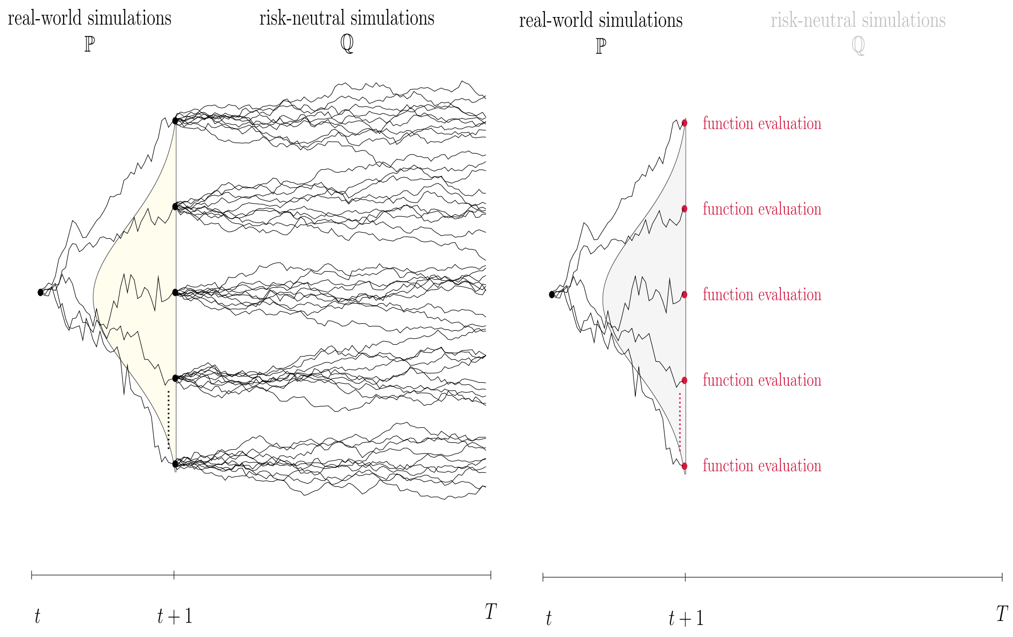


FIGURE 1 Representation of the nested Monte Carlo (left) and least squares Monte Carlo (right) simulations [Colour figure can be viewed at wileyonlinelibrary.com]

From the algorithmic point of view, the probability distribution forecast of asset and liability at time $t+1$ is obtained by defining a time grid from present to run-off time (that is to the time of the payment of the last amount due). The stochastic processes of the risk drivers are simulated over the time grid according to the real-world probability measure $\mathbb{P}$ up to $t+1$ and then, conditional on the values in $t+1$, to run-off, according to the risk-neutral probability measure $\mathbb{Q}$. Successively, cash flows are computed, and finally prices can be determined. Notice however that the determination of prices as conditional expectations requires to generate many risk-neutral trajectories for each real-world trajectory, as represented in Figure 1. This two-phase procedure results in a nested Monte Carlo simulation.

The nested Monte Carlo approach may be avoided if prices in $t+1$, that is expectations, can be computed in closed form. Indeed, this is often possible for assets, that generally are contracts with simple cash flows (e.g. government bonds), while it is extremely rare for liabilities. Notice that, from the financial point of view, contracts that foresee the right to surrender at any time before maturity—in the financial jargon called *American style* derivatives—cannot have closed form solutions to the expectation since it can be shown that their price is the solution of a free boundary ordinary differential equation.

The theoretical approach and the computational issues of nested Monte Carlo procedure require fine precautions.[17] It has been proved that, in the evaluation of the quantile, the asymptotic variance of the Monte Carlo estimator depends on number of real-world trajectories, N, while the asymptotic bias is influenced by risk-neutral simulations, K.[3,17] Since the computational cost to produce a dataset is proportional to $N \times K$, the choice of the couple (N, K) should be carefully considered in real-world applications.[4]

In order to have a good estimation of the SCR, both the number of risk-neutral and real-world simulations have to be very large, thus resulting in a "compute-intensive" process.[2,18,19] Nevertheless, nested Monte Carlo simulation is the standard approach for insurance undertakings[3,4,17] even if it can present prohibitive computational challenges.

This background excursion should clarify the emerging technical responsibilities of the insurance company governance: ensure consistency of the evaluation between present and future values; rightly manage the risk drivers' probabilities, the correlations and the aggregation process. Then, the quantitative risk management required by Solvency II involves methodological and algorithmic complexity: stochastic models, pricing techniques, econometric processes, optimisation processes, Monte Carlo simulation.[20]

Further, a timely response is often crucial for the effectiveness of decisions and for providing answers to be given in compliance with regulations (market disclosure, information to the supervisory authority); a (complex) trade-off between processes efficiency and regulatory constraints has to be found.

In the described framework, the present work is devoted to investigate how well-established machine learning techniques might reduce the computational burden of nested Monte Carlo simulation, empirically assessing their effectiveness and comparing their computing efficiency and accuracy. The recourse to consolidated methodologies of machine learning is motivated by the aim of attracting the interest of insurance industry, taking into account, at the same time, the necessary trade-off between accuracy, efficiency and the regulatory constraints for the approval of the model.

At present, to achieve this trade-off, the insurance undertakings mainly use high performance computing technology (parallel and distributed systems, grid computing, cloud computing)[2] and LSMC method.[21]

Briefly, LSMC, a technique originally designed to price American style derivatives,[22] is used to strongly reduce the number of risk-neutral simulations by finding a closed form approximation of the value of assets and liabilities as a function of the risk drivers in $t+1$. Practically, the full Monte Carlo determination of the value of asset and liability is replaced by a truncated series expansion in orthonormal polynomials. The coefficients of the series expansion are calibrated on a reduced-size nested Monte Carlo, and the resulting polynomial is applied to a large dataset produced by the real-world loop, thus substituting the risk-neutral loop with the analytical expression obtained by the expansion. The main drawbacks with this technique are the choice of the polynomials, the way the (multidimensional) expansion is truncated to prevent a combinatorial explosion in the number of coefficients, and the estimation of the error introduced by truncation. This approach can be viewed as a particular case of supervised learning, that is an application of machine learning where a dataset is available to train the model.

This work aims to investigate alternative approaches in computing the SCR of typical *with-profit life insurance policies*, based on (SVR) and (DLN), and to compare them with LSMC method used as benchmark. Here, DLN and SVR are used in the risk-neutral loop in place of LSMC polynomial regression to overcame the previous shortcomings and to improve accuracy of the evaluation. In Section 4 all three methods are described more in detail.

3 | VALUATION IN THE SOLVENCY II FRAMEWORK

The internal models for the determination of the SCR in the Solvency II framework are typically composed of (a) an Economic Scenario Generator that creates via Monte Carlo simulation *real-world* scenarios of financially relevant risk drivers (inter alia nominal and real interest rates, both riskless and risky, currency exchange rates, equity and property values) at the end of the prescribed 1-year time horizon, (b) an Actuarial Scenario Generator that provides *real-world* scenarios of actuarially relevant risk drivers (inter alia mortality and lapse rates), (c) a procedure for the determination of the value (price) of asset and liabilities, conditional on the values of the risk drivers, also taking into account the management actions possibly reshaping the composition of assets and liabilities portfolios. From the evaluation point of view, life insurance contracts are extremely peculiar, since for life policies financial risks affects simultaneously the value of assets and liabilities in a very complex fashion.

The case examined in this work concerns a life insurance portfolio composed of the so called *with-profit* or *participating policies*, characterised by "profit-sharing rules", which provide the possibility for the policyholder to participate in the earnings the insurance company derives from investing the premia in an assets fund (in the Italian market the "segregated fund"[2]). In addition, the policy offers the right to a minimum interest rate guarantee; hence the policy embeds a financial option, typically of the "cliquet" type[3].[23] Using a standard decomposition, it is possible to express the value of the policy as the sum of a guaranteed investment and a call option,[23] whose value, in the Solvency II jargon, is called the value of *future discretionary benefits* (FDBs). While the valuation of the guaranteed component is relatively simple, at least as difficult as the valuation of a government bond, the valuation of FDBs is the most difficult part of the liability valuation procedure.

To further illustrate the valuation framework, one may refer to the case of a single-premium pure endowment insurance contract, written at time t for a life (individual) with age x, with term T years and a premium π_t. In this type of contract, the benefits are calculated at the end of the year T according to the *revaluation rule*

$$C_T = \pi_t \Phi_T, \tag{4}$$

where the readjustment factor Φ_T is defined as

$$\Phi_T = \prod_{k=1}^{T} [1 + \max(\beta_k I_k, \gamma_k)], \tag{5}$$

[2]Given the typically very long time horizon of life policies, this type of funds is accounted in a different balance sheet from that of the insurance company, thus offering the policyholder protection against the default of the company itself.
[3]"Cliquet" is the French word for the pawl mechanism; in this context it refers to the fact that benefits acquired in 1 year cannot be cancelled by later negative performances of the fund.

where γ_k is the effective minimum guaranteed annual cliquet rate for the year k, I_k is the annual rate of return of the asset fund in the year $[k-1, k]$, $\beta_k \in (0, 1]$ is the participation coefficient. In this case, the cash flow is composed by a payment of π_t unit of currency at time t from the policyholder to the insurance company, followed by either a payment of C_T units of currency at time T from the company to the beneficiary if the holder is alive and has not policy surrendered before time T, or a payment at a random time $\tau \in (t, T]$ of an amount, computed according to the clauses established in the contract, that depends on whether the policyholder has either died or surrendered.

Notice that the rate of return I_k in (5) is a random variable; however, differently from the return of typical financial instruments, it is computed using "accounting rules" and it depends on the management strategy of the fund. This is in line with the market consistency approach, since I_k is used only to determine the benefits of the policy. Insurance company prefer this mechanism of indexing since it provides a way to smooth the behaviour of financial markets, hence providing a rate of return to the policyholder that varies slowly in time. In the language of financial options, this means that the volatility of I_k is smaller than the value that would have been computed *mark-to-market*, which in turn decreases the value of the *FDB*s. From the valuation point of view, the "accounting rules" imply the need to simulate the management strategy, in addition to the simulation of financial and actuarial risk drivers. Details of the management actions are usually not disclosed to the public. If we denote by $\mathcal{E}(x, T)$ the event that outcomes to payment (e.g. "the aged x policyholder is alive at time T"), then the benefit for the policyholder—the liability of the company—in T is given by

$$Y_T = C_t \Phi_T \mathbb{I}_{\mathcal{E}(x,T)}, \tag{6}$$

where $\mathbb{I}_{\mathcal{E}(x,T)}$ is the indicator function of $\mathcal{E}(x, T)$ expressing "technical" (actuarial) uncertainty. For this kind of policy uncertainty arises from financial risk drivers that determine the value of the segregated fund and then of I_k, from the details of the management strategy of the fund, and from the actuarial risk drivers that determine the survival (or surrender) of the life insured. The value of the liability at time t is the value at time t of Y_T, while the value of the assets is the value of the segregated fund in which π_t is invested. According to the fundamental theorem of asset pricing, the value of liabilities, that is the value at time t of Y_T, $V(t, Y_T, \mathbf{r}(t))$ can be computed as[4]:

$$Y_t := V(t, Y_T, \mathbf{r}(t)) = \mathbf{E}_t^{\mathbb{Q}} \left[e^{-\int_t^T r_u du} \, Y_T \right], \tag{7}$$

where $\mathbf{r}(t) = (r_t, S_t, p_t, \dots)$ denotes the d-dimensional vector of risk driver values at time t, r_t is the *risk-free* instantaneous spot rate at time t and $\mathbf{E}_t^{\mathbb{Q}}[\cdot] = \mathbf{E}^{\mathbb{Q}}[\cdot|\mathcal{F}_t]$ is the expected value, conditional to the information available at time t, according to the risk-neutral probability measure $\mathbb{Q}$.

The complexity of profit sharing rule and the management actions involved entails to use numerical simulation for valuating $V(t; Y_T, \mathbf{r}(t))$ and in fact the risk-neutral expectation in equation (7) is often computed by Monte Carlo simulations on fine grained grid time.

In order to obtain an empirical probability distribution function of contracts values in future time H[5] (with $t \le H \le T$) and then the corresponding moments and percentiles, the value of the liabilities must be calculated for each possible future realisation of the risk drivers $\mathbf{r}^{(i)}(H)$ simulated according to the real-world probability measure $\mathbb{P}$.

The need to generate a realistic set of future economic scenarios $\mathbf{r}^{(i)}(H)$ for $i = 1, \dots, N$ (under $\mathbb{P}$) and simultaneously perform the expected values $V^{(i)}(H, Y_T, \mathbf{r}(H))$ in Equation (7) (under $\mathbb{Q}$) in all scenario involves nested Monte Carlo simulations. The use of this technique requires:

1. the simulation of N sample paths $(\mathbf{r}^{(i)}(H))$, $i = 1, \dots, N$, from $t = 0$ to $t = H$ under the real-world measure $\mathbb{P}$, conditional to $\mathcal{F}_0$ (in particular conditional to $\mathbf{r}(0)$ if $\mathbf{r}(t)$ is a Markov process);
2. for each of the N sample paths $(\mathbf{r}^{(i)}(H))$ from $t = 0$ to $t = H$, the simulation of K sample paths $(\mathbf{r}^{(i,j)}(H))$, $j = 1, \dots, K$, from $t = H$ to $t = T$ under the equivalent martingale measure $\mathbb{M}$ (for example the risk-neutral probability $\mathbb{Q}$), conditional to $\mathcal{F}_H$ (in particular conditional to $\mathbf{r}(H)$ if $\mathbf{r}(t)$ is a Markov process).

The number of "inner" simulations N determines the accuracy of the $V(H, Y_T, \mathbf{r}(H))$ estimation, while the number of "outer" simulations K affects the numerical precision in the valuation of the percentile of $V(H, Y_T, \mathbf{r}(H))$. The number of both inner and outer simulations must be sufficient to ensure that the selected risk measure can be calculated with a

[4]Notice that $Y_t \neq \pi_t$.
[5]In the Solvency II framework the future time is set $H = t + 1$.

prefixed approximation error and in pre-established computing time.[2] It is important to note that from the computational point of view the risk neutral evaluation of the liabilities is the most time-consuming part of the procedure.

For the present work, the values of the empirical probability distribution of assets and liabilities have been determined using *DISAR*® (Dynamic Investment Strategy with Accounting Rules), a commercial computational system that tackles market-consistent valuation of complex cash flows using numerical techniques in a stochastic framework,[2,19] designed to be used in the Italian insurance sector.[23] DISAR is composed by a database management system and of a set of computing engines. The simulation of the evolution of market risk drivers is based on a stochastic model and Monte Carlo simulation is used.

A brief description of the Economic Scenario Generator used by DISAR is provided in Appendix A.

4 | METHODS

In this section, a brief description of the methods under investigations is provided. The idea behind each method is to fit the values of *FDB* using a function $f(\mathbf{r})$ (for notational convenience, $\mathbf{r}$ will be used to indicate $\mathbf{r}(H)$ in the rest of the paper) that includes risk factors as inputs.

Let $V(H, Y_T, \mathbf{r})$ be the market value of the future liabilities of insurance company at time H; a model that links liabilities to risk drivers is sought:

$$V(H, Y_T, \mathbf{r}) = f(\mathbf{r}) + \varepsilon, \tag{8}$$

where ε is a "small" error term. LSMC, SVR and DLN have been employed to fit the value of the liabilities avoiding the time-consuming risk-neutral evaluation.

4.1 | Least squares Monte Carlo

The LSMC approach represents an important starting point in Solvency II context and provides a benchmark to evaluate alternative methods for function approximation.[21] The use of orthogonal polynomials can be formally justified when the sought function $f : \Omega_{\mathbf{r}} \to \mathbb{R}$ is an element of L^2, the space of square-integrable functions relative to some weighting function $\omega(\mathbf{r})$, that is:

$$\|f(\mathbf{x})\|^2 = \int_{\Omega_{\mathbf{x}}} f^2(\mathbf{x})\ \omega(\mathbf{x})\ d\mathbf{x} < +\infty. \tag{9}$$

Moreover, for any couple of L^2 functions, their product $g \cdot h$, is defined as:

$$g(\mathbf{x}) \cdot h(\mathbf{x}) = \int_{\Omega_{\mathbf{x}}} g(\mathbf{x})\ h(\mathbf{x})\ \omega(\mathbf{x})\ d\mathbf{x}. \tag{10}$$

In this case any L^2 function $f(\mathbf{r})$ can be expressed as a linear combination of a countable set of orthogonal basis functions measurable in L^2:

$$f(\mathbf{r}) = \sum_{k=0}^{\infty} \alpha_k\ p_k(\mathbf{r}), \quad \alpha_k = \frac{f(\mathbf{r}) \cdot p(\mathbf{r})}{\|L_k(\mathbf{r})\|^2}, \tag{11}$$

where the coefficients α_k are real numbers and the p_k $(k = 0, 1, 2, \ldots)$ constitute an orthogonal (possibly orthonormal) basis of polynomials of order k. In practical applications, given a training set $\{\mathbf{r}_i, v_i\}_{i=1}^N$, where the response variable v_i corresponds to realisations of the quantity under observation $V(t, Y_T, \mathbf{r})$ and the covariates $\mathbf{r}_i$ refer to the realisations of the risk factors, the series in (11) is truncated at order m:

$$f_m(\mathbf{r}) = \sum_{k=0}^{m} \alpha_k\ p_k(\mathbf{r}), \tag{12}$$

and the coefficients α_k are determined by solving the least-squares problem:

$$\min_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{N} (v_i - f_m(\mathbf{r}_i))^2. \tag{13}$$

TABLE 1 Number of terms for a polynomial given m and d

	d							
m	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1	2	3	4	5	6	7	8
2	2	5	9	14	20	27	35	44
3	3	9	19	34	55	83	119	164
4	4	14	34	69	125	209	329	494
5	5	20	55	125	251	461	791	1286
6	6	27	83	209	461	923	1715	3002
7	7	35	119	329	791	1715	3431	6434
8	8	44	164	494	1286	3002	6434	12 869
9	9	54	219	714	2001	5004	11 439	24 309

The use of orthogonal polynomials as approximators involves some problems:

1. the choice of the weighting function $\omega(\mathbf{r})$, that typically depends on $\Omega_{\mathbf{r}}$ being open or limited;
2. the choice of the polynomial degree m;
3. the selection of significant terms amidst all possible ones; in fact, given the number of risk drivers $d > 1$, the number of terms in an d-dimensional orthogonal polynomial increases very rapidly when the degree of the polynomial raises, being equal to

$$\binom{m+d}{m}. \tag{14}$$

Table 1 shows the number of terms for polynomials of degree m in d number of variables.

To keep the number of regression terms low when d and m grow, it is possible to use recursive techniques for variable selection, such as the forward, the backward and the stepwise method.[21] These techniques add, delete, or alternate between adding and deleting variables on the basis of their significance.

4.2 | Support vector regression

A different approach to approximate the unknown function in (8) involves the use of SVR. While based on common underpinnings, different flavours of SVR exist. Here, ϵ-insensitive regression is discussed. Consider a training set $\{\mathbf{r}_i, v_i\}_{i=1}^N$ and a linear model

$$f(\mathbf{r}) = \mathbf{w}^T\mathbf{r} + w_0. \tag{15}$$

The broad objective is minimisation of the empirical risk

$$R_e(f) = \frac{1}{N}\sum_{i=1}^{N}\Theta_\epsilon(v_i - f(\mathbf{r}_i)),$$

where $\Theta_\epsilon(\cdot)$ is the Vapnik ϵ-insensitive loss function

$$\Theta_\epsilon(x) = \begin{cases} |x| - \epsilon & \text{if } |x| \geq \epsilon \\ 0 & \text{otherwise} \end{cases}$$

that causes small residuals (of size less than ϵ) to be ignored. Moreover, not being squared as in classical regression, large residuals have a limited impact. Introducing slack variables $\xi_1^+, \dots, \xi_N^+$ and $\xi_1^-, \dots, \xi_N^-$ (respectively for positive and negative deviations) to express the amount by which a constraint is violated in a data point, SVR minimises the overall error

$$\min_{\mathbf{w},w_0} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i^+ + \xi_i^-) \tag{16}$$

subject to

$$\begin{aligned} &v_i - (\mathbf{w}^T\mathbf{r}_i + w_0) \le \epsilon + \xi_i^+ \quad && i = 1, 2, \dots, N \\ &(\mathbf{w}^T\mathbf{r}_i + w_0) - v_i \le \epsilon + \xi_i^- \quad && i = 1, 2, \dots, N \\ &\xi_i^+, \xi_i^- \ge 0 && i = 1, 2, \dots, N. \end{aligned}$$

The first term in (16) is a regularisation term that penalises large weights, the second term penalises training errors, and C is a tuning parameter that controls the relative influence of these two terms. The quadratic problem is solved by means of Lagrange multipliers α_i^+, α_i^-. Because of the shape of the ϵ-insensitive loss function, only some of the Lagrange multipliers are nonzero. Those are termed *support values*, and the corresponding training data points are called *support vectors*. The solution can be written as

$$\mathbf{w}^T = \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)\,\mathbf{r}_i^T. \tag{17}$$

Using the property that the fitted line depends on the input values only through internal products, as it can be seen combining (15) and (17), the linear method can been extended to a nonlinear technique. A nonlinear mapping $\phi(\cdot)$ puts the space where input data reside in correspondence with a new space, termed feature space, where a linear regression model expresses the relationship between the response variable and the covariates. Computing the transformation $\phi(\mathbf{r})$ explicitly is not necessary: internal products can be obtained by means of a kernel function $\mathcal{K}(\cdot,\cdot)$

$$\mathcal{K}(\mathbf{u},\mathbf{v}) = \langle \phi(\mathbf{u}), \phi(\mathbf{v}) \rangle.$$

Any function satisfying Mercer's conditions[24] can be used as a kernel function. The most commonly used kernels are

a) the Gaussian Radial Basis Function (RBF): $\mathcal{K}(\mathbf{u},\mathbf{v}) = \exp(-\|\mathbf{u}-\mathbf{v}\|^2/\sigma^2)$;
b) the polynomial kernel (of degree η): $\mathcal{K}(\mathbf{u},\mathbf{v}) = (a_0 + a_1\langle \mathbf{u},\mathbf{v}\rangle)^\eta$;
c) the hyperbolic tangent kernel: $\mathcal{K}(\mathbf{u},\mathbf{v}) = \tanh(a_0 + a_1\langle \mathbf{u},\mathbf{v}\rangle)$.

Note that σ, a_0 and a_1 are kernel-specific hyperparameters to be tuned. The forecasting accuracy of a SVR model indeed depends on a good setting of hyperparameters. Such hyperparameters to be tuned in SVR include the tolerance ϵ, the cost C as well as hyperparameters specific to the kernel, such as the σ that controls the scale for the RBF kernel. C regulates the trade-off between margin maximisation and error minimisation. Large values create a high penalty for large residuals, and overfitting may ensue as a consequence. The choice of ϵ also has an effect. In practice, however, the cost parameter provides more flexibility.[25] On the other hand, small values pose a risk of underfitting.

4.3 | Deep learning networks

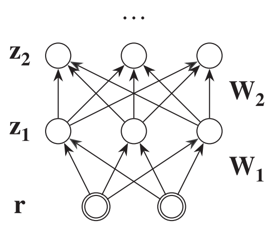
Deep learning has recently achieved excellent performance on a wide array of problems across several fields.[26]

The structure of a (feed-forward) DLN is depicted in Figure 2. A number of interconnected *units* are arranged in layers. Units in the bottom layer are "clamped" to the input, denoted by $\mathbf{r}$. Considering a DLN architecture for the regression problem in Equation (8), let $g \in \mathbb{N}$ the number inner layers—each layer with several units—the output $f(\mathbf{r})$ is obtained by:

$$\begin{aligned} \mathbf{z}_1 &= \phi_1(W_1^T\mathbf{r} + \mathbf{b}_1) \\ \mathbf{z}_2 &= \phi_2(W_2^T\mathbf{z}_1 + \mathbf{b}_2) \\ &\dots \\ f(\mathbf{r}) &= \phi_g(\mathbf{W}_g^T\mathbf{z}_{g-1} + b_g), \end{aligned} \tag{18}$$

where $W_1, W_2, \dots, W_g$ are weight matrices, $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_g$ are bias vectors, and $\phi_1, \phi_2, \dots, \phi_g$ are activation functions not necessarily different from each other. Weights and biases (collectively, the network parameters) are adjusted to reflect the

FIGURE 2 A deep neural network



characteristic of training data through a learning algorithm. A cost function (in regression a measure of the prediction error, such as the mean squared error) is minimised by (backward) gradient descent on the parameters. Commonly used nonlinear functions include the sigmoid $\phi(z) = 1/(1 + e^{-z})$, the hyperbolic tangent $\phi(z) = \tanh z$, and the rectified linear[6] $\phi(z) = \max(0, z)$. The output of the top layer constitutes the network output. Inner layers are called hidden layers. The layered structure of DLNs allows to learn multiple levels of representation.[27] Instead of transforming raw input into a carefully crafted internal representation where patterns could show up, raw input is directly fed into a model, letting it learn multiple representations at an increasing level of "abstraction" in the inner layers.

Feed-forward networks are well known for their universal approximation properties. It has been mathematically proven that they can approximate any continuous function arbitrarily.[28] Known issues with deep neural networks include their opacity (the inability to explain the reasons behind decisions taken) and their limited robustness to perturbations of their input.[29]

In practice, various hyperparameters influence the performance of a DLN, including:

(a) Architecture. A first relevant question is the choice of number of hidden layers. Once the number of hidden layers is determined, another important configuration choice is the number of units in each hidden layer. That is a complex question in neural networks, because an increase in the number of units tends to reduce training error but increase testing error (overfitting).
(b) Activation function. Shape and steepness of the activation function have a substantial impact on performance.
(c) Learning algorithm. The modality and rate with which network parameters are adjusted influence both training quality and training time. Stochastic gradient descent provides the basis for the learning algorithms currently in widespread use. A high learning rate can cause divergence, while choosing this rate too low slows learning down.
(d) Regularisation. To constrain the weights, a regularisation method can be used. L_1-regularisation acts on the absolute value of the weights and has the effect of sparsifying weights to reduce model complexity and avoid overfitting. L_2-regularisation limits the sum of squares of the weights, reducing variance at the price of introducing some bias in the parameter estimates.
(e) Dropout. The input dropout ratio represents the fraction of the features for each training example that are randomly omitted from training in order to improve the generalisation capabilities of the network.[30]

4.4 | Comparison

ML-based methods and traditional LSMC present common points and distinctive features. For sure, the calibration of all three methods is a delicate process that must be handled with care. ML literature shows that the performance of these techniques depends on the values of the hyperparameters and unfortunately this issue also affects LSMC. In fact, there is no unanimous literature on the kind of orthogonal polynomials to use and an inappropriate maximum degree of the polynomials regression could involve underfitting or overfitting problems. The calibration process highlights other pros and cons of each method. The hyperparameters tuning of machine learning-based methods can be conveniently carried out automatically using tools integrated into several computational software packages, but it often is time-consuming. On the contrary, the calibration of the LSMC is sometimes computationally less demanding but it requires the modeler to make more decisions.

Another element that points out different patterns among the methods is the number of risk drivers that characterises the insurance portfolio under investigation. As described in the previous section, in the LSMC setting, the number of terms of the polynomial regression rapidly grows when the dimension of input data increases. This suggests that adopting

[6]Units with such an activation function are termed rectified linear units (ReLU).

a LSMC approach with a sufficiently high degree on an insurance portfolio affected by a large number of risk drivers could be very difficult due to the unmanageable number of terms. Differently, one of the most important merits of the SVR and DLN techniques is the overcoming of the curse of dimensionality. Indeed the complexity of their calibration does not depend on the dimension of the input space and then it makes them promising tools to assess insurance portfolios with a large number of risk drivers.

5 | NUMERICAL EXPERIMENTS

All the experiments have been carried out using data produced by DISAR system that represent the starting point for supervised learning procedures. Then, using DISAR, the investigation is performed in a realistic framework, considering some benchmark portfolios of policies—mimicking those hold "on average" on the Italian market—and a set of best practice models and techniques.

For the goals of this work, other relevant inputs to DISAR are the number N of real-world simulations and the number K of risk-neutral simulations, that determine the ultimate precision achievable on the evaluation of the *SCR*. DISAR's output typically includes: (a) the values of the risk drivers in $t+1$, (b) the market-consistent values of assets and liabilities, (c) the decomposition of liabilities into a guaranteed component and *FDB*s, whose valuation requires nested Monte Carlo simulations. The Directive specifies that the value of *FDB*s should be calculated separately.[1] For this reason, the goodness of fit of the *FDB*s needs to be checked. Sensitivity of the results to the values of financial variables is out of the scope of this work. However, it is important to stress that the complexity of the functions to be approximated depends mildly on market conditions (therefore on financial parameters) and strongly on the type of the highly nonlinear payoff of the insurance contracts considered.

All experiments have been run on a 8-core Linux server with R version 3.4.3, using the keras, mlr, parallel, parallelMap, orthopolynom, e1071 packages for the selected models.

5.1 | Valuation of accuracy

The first set of experiments is targeted to determine the accuracy with which each of the methods under study can approximate the value of the *FDB*s. A comparison of the performance of a SVR and a DLN regressor has been carried out. Results are also compared with those obtained through approximation with the traditional LSMC. Some global accuracy measures have been considered to this purpose such as normalised root mean squared error (NRMSE), Kolmogorov–Smirnov (KS) distance and Kullback–Leibler (KL) divergence. The first one is the RMSE normalised by the range of the response variable (to improve the readability) and it measures the error produced by the model in predicting the quantities of interest. KS distance is a metric that measures the distance between two distributions; it is defined as the largest absolute difference between the two empirical cumulative distribution functions. Using this measure it is possible to quantify the distance between the reference distribution and the approximation produced by each model. Finally, KL divergence measures how much information is lost when a reference distribution is replaced by an approximation. Monitoring this value it is possible to determine the loss of information due to using one of these models instead of nested simulations. The joint use of these three measures allows to compare the results produced by investigated methods in detail. Moreover, in order to assess the accuracy of the evaluation of SCR$(0,1)$, the relative error on the ratio between the *SCR* and $V(0)$—that provides guidance on the riskiness of the policy—has been used as a tail-based measure.

DISAR has been used to produce two empirical probability distributions of the *FDB*s (v_i, $i=1, \dots, N$) with respectively $N=10{,}000$, $K=1000$ and $N=100{,}000$, $K=10{,}000$. In addition, the output of DISAR includes the values of the d-dimensional vectors of risk drivers $\mathbf{r}_i$, $i=1, \dots, N$ at $t=1$ year. The comparison among methods requires establishing a "ground truth". Since the lack of closed-form formulas for the *FDB* values, the accuracy of each method is measured as the distance from the empirical distribution of the *FDB* obtained by a very large simulation which represents the best reference distribution available. This approach represents the standard in the mainstream literature.[2,14] In our setting, the full nested simulation performed with $N=100{,}000$ and $K=10{,}000$ is considered as benchmark.

The evaluation process consists of two phases:

1. *Calibration phase*. In order to identify the best 'configuration' of each model under investigation, the tuning procedure is carried out on DISAR output with $N=10{,}000$ and $K=1000$, using $(\mathbf{r}_i, v_i)$, $i=1, \dots, N$. Following standard practice,

the dataset has been partitioned in two subsets, the training set and the validation set, respectively accounting for 2/3 and 1/3 of the data. The output of calibration is the set of parameters giving the best performance on the validation set.

2. *Testing phase*. The best configuration of parameters of the calibration phase has been tested using the values of risk drivers obtained with $N = 100{,}000$. The probability distribution of the *FDB*s, produced by each model has been compared with the benchmark.

In the calibration phase, each model requires the setting of specific characteristics. An initial round of experiments has been performed to roughly establish the ranges to be explored for hyperparameters. Subsequently, a detailed calibration has been carried out, as described hereafter.

1. *LSMC*. A linear regression of *FDB*s against orthogonal polynomials of degree 2, 3 and 4 in the seven risk drivers has been performed on the training and validation sets. Ordinary, Legendre, Hermite and Laguerre polynomials have been compared to select the most appropriate kind of polynomials to approximate the data. The number of terms grows rapidly as the degree of the polynomial increases. Degrees higher than 4 have not been taken into account. To reduce the number of terms to handle, a stepwise procedure based on the Akaike information criterion (AIC) has been applied. As shown in Reference 21, this procedure turns out to be the one that picks the lowest number of terms while not sacrificing regression quality.
2. *SVR*. Hyperparameter tuning for SVR has been performed in a two-stage procedure. The first stage is composed of coarse-grained experiments where different kernels have been compared. For each kernel, a set of 2^6 preliminary experiments with different parameters was done. Therefore, the second stage was limited to the RBF kernel which is resulted in being the most promising one. Values in a grid from 0.014 to 0.018 with a step of $2 \cdot 10^{-4}$ have been tested for ϵ in Equation (8). For the parameter C, five equally spaced values from 50 to 300 have been considered. The RBF hyperparameter σ was allowed to vary between 0.004 and 0.007 with a step of $1 \cdot 10^{-4}$.
3. *DLN*. The number of analysed hidden layers ranged from 1 to 4; each layer has been tested with a number of units equal to $\{2^4, 2^5, 2^6\}$. The ReLu, tanh, Softmax activation functions, as implemented in the keras package, have been examined. The Adam algorithm[31] has been used to adjust the weights. Regularisation methods tested were L_1-regularisation, L_2-regularisation and dropout. Moreover, in order to fight the vanishing gradient problem that affects deeper learning networks, skip connections (additional connections that directly connect the input layer to the output layer concatenate the input data with the output of the last hidden layer[32]) were provided for neural networks with more than three layers.

Now, findings relative to the three different approaches are firstly individually reported. A comparison and final conclusions follow.

First, Table 2 shows the results of the LSMC calibration reporting NRMSE obtained with ordinary, Legendre, Hermite and Laguerre orthogonal polynomials for, respectively, the training and validation sets.

One can note that ordinary, Hermite and Laguerre polynomials overperformed Legendre polynomials in the validation sample. This is not surprising since Legendre polynomials are unsuited for functions defined over open intervals. The similarity between Laguerre and Hermite polynomials has been ascribed to the fact that the involved financial risk drivers have distributions limited to $\mathbb{R}^+$ or with small tails in $\mathbb{R}^-$. The similarity between ordinary and Laguerre/Hermite polynomials is ascribed to the fact that, by simple resummation, truncated Hermite/Laguerre polynomials have the same analytic expression of ordinary polynomials. Laguerre polynomials overperformed the other families. The accuracy improvement obtained by increasing the degree of polynomials from 2 to 3 is modest, while from 3 to 4 is nearly negligible. Degree 3 seems the best choice because increasing the degree to 4 does not carry an improvement in accuracy commensurate to the explosion in the number of terms. The AIC-based stepwise procedure selected 51 terms over the 119 possible ones for the Laguerre polynomial of degree 3.

Second, the results obtained from the two-stage calibration of the SVR are described below. In the first stage, we compare the best performance of each kernel to find the most suitable one for our experiments. In this phase, we observe that RBF (0.005836) seems to be more promising than linear (0.006635) and polynomial (0.006637) kernel. This evidence can be probably justified remarking that, differently from the other kernels, the RBF kernel involves that the scalar product between two training examples is performed in an infinite dimensional space.[33]

In the second stage, the tuning procedure is focused on the RBF kernel. Interestingly, this kernel has a very low number of hyperparameters, so a full Cartesian search has been run, obtaining a cost $C = 250$, a tolerance $\epsilon = 0.0158$, and a value of the σ hyperparameter for the RBF kernel of 0.0065.

TABLE 2 Performance of LSMC for ordinary and orthogonal polynomials

		NRMSE	
Type	**Degree**	**Training**	**Validation**
Ordinary	2	0.00541	0.00615
Ordinary	3	0.00519	0.00591
Ordinary	4	0.00502	0.00592
Legendre	2	0.00541	0.00615
Legendre	3	0.00519	0.00590
Legendre	4	0.00504	0.00588
Hermite	2	0.00541	0.00616
Hermite	3	0.00519	0.00590
Hermite	4	0.00504	0.00587
Laguerre	2	0.00541	0.00616
Laguerre	3	0.00521	0.00588
Laguerre	4	0.00507	0.00583

TABLE 3 Performance of neural networks for different number of layers, units and different activation functions

		N. of layers			
N. units	**Activation f.**	**1**	**2**	**3**	**4**
2^4	tanh	0.006209	0.006456	0.005858	0.005806
	relu	0.007303	0.009520	0.005791	0.005842
	sigmoid	0.006255	0.006116	0.005854	0.005812
2^5	tanh	0.006224	0.006524	0.005930	0.006323
	relu	0.007358	0.006614	0.005807	0.005818
	sigmoid	0.006216	0.006109	0.005826	0.006099
2^6	tanh	0.006202	0.006628	0.006032	0.006653
	relu	0.006097	0.007718	0.005849	0.005937
	sigmoid	0.006253	0.006199	0.006013	0.006212
	average	0.006457	0.006876	0.005884	0.006056

Finally, tuning results for DLN are reported in Table 3 that provides the performances in term of NRMSE for different number of units, number of layers and activation functions. The best performance is obtained with a 3-layered neural network with 16 units in each layer and relu activation function. In addition, other interesting comments can be done:

a) regarding the activation function, there is not a clear ranking among the tested functions for shallow neural networks but the ReLU function outperformed the other functions tested when more than three hidden layers were considered;
b) as shown by the average performance by number of layers (the last row of Table 3), single-layer and two-layers networks generally underperformed networks with three or four layers and the best performance was obtained using a three-layered architecture. This evidence probably suggests that shallow networks have a too low number of weights while the four-layered networks include redundant weights;
c) performance usually improved with a not very high number of units on each layer;
d) no clear evidence emerged for the influence of other hyperparameters.

The comparison among the methods is discussed below. The hyperparameters tuning of ML-based methods requires substantial time, 4 h for SVR and 10 h for DLN, while LSMC is faster in this phase. It should be noted that this process can be done only once for each insurance portfolios and, although it has not been studied in this work, the running times

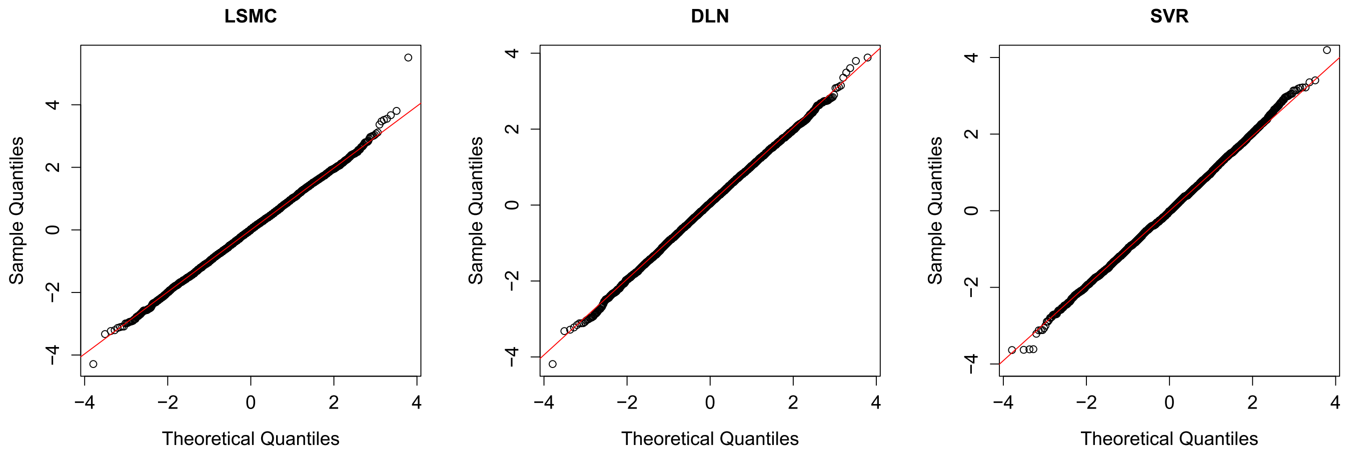


FIGURE 3 Normal quantile–quantile plots for the standardised residuals of the three models [Colour figure can be viewed at wileyonlinelibrary.com]

TABLE 4 Results for Kolmogorov–Smirnov test

	Model		
	LSMC	SVR	DLN
p-value	.9203	.2692	.8659

TABLE 5 Performance on testing set of LSMC, DLN and SVR in *FDB*s evaluation

Model	NRMSE	KS distance	KL divergence
LSMC	0.002443	0.00441	0.04054
SVR	0.002394	0.00346	0.00773
DLN	0.002262	0.00151	0.00517

required by the hyperparameters tuning of the machine learning methods can be reduced by using parallel computing systems.

After having identified the configuration and best values of the hyperparameters for, respectively, the polynomial, DLN and SVM models, a comparison between the three approaches has been performed with respect to (*a*) internal consistency of the model, as measured by the distribution of the residuals; (*b*) precision attained in the determination of *FDB*'s distribution, measured by the NRMSE, (KS) distance and (KL) divergence; (*c*) the quality of the *SCR* estimate.

The quantile-quantile plots in Figure 3 depict the standardised empirical quantiles for the residuals of the three models against the quantiles of a standardised normal distribution. The chart shows a distribution of residuals close to normal for all three models. This is further confirmed by the results, reported in Table 4, of the Kolmogorov–Smirnov test.

After the diagnostics on residuals, the models are used to fit the *FDB*s value in all real-world scenarios. The accuracy of fitted distributions is evaluated using as a benchmark the *FDB*s estimated with full nested MC with 10,000 risk-neutral simulations.

Table 5 reports the performance of LSMC, DLN and SVR on the testing set in terms of NRMSE, KS distance and KL divergence.

A first interesting result concerns the ranking among the models which does not seem to vary for all three measures considered. SVR and DLN exhibit a higher precision in approximating the *FDB*'s distribution. More specifically DLN seems to produce the best performance among the tested models.

Looking to the NRMSE values, performances of all three models are adequate; interestingly, the *FDB*'s values predicted by SVR and DLN seem to be more accurate in the least squares sense. The lowest NRMSE coincides with the smallest KS distance between the reference distribution and the approximation. This result suggests that when the FDB values are predicted with a higher level of precision, the approximation of the empirical distribution tends to have a lower distance from the reference one. In addition, investigating the results in terms of KL divergence, it is possible to note that the

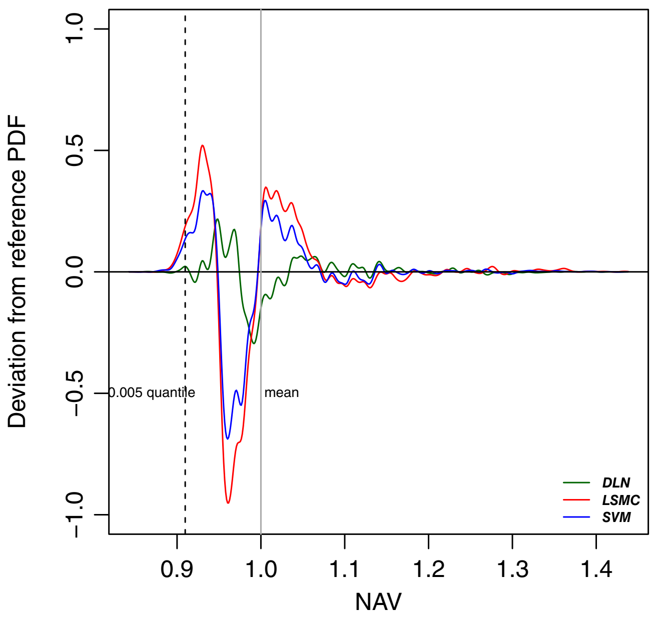


FIGURE 4 Deviance of between *NAV* estimations obtained by LSMC, SVR and DLN against the nested Monte Carlo benchmark [Colour figure can be viewed at wileyonlinelibrary.com]

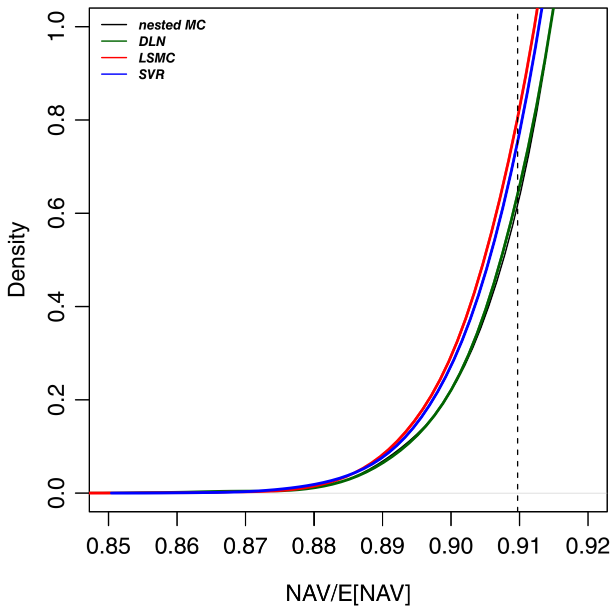


FIGURE 5 Kernel density estimates of the *NAV*: left tail [Colour figure can be viewed at wileyonlinelibrary.com]

approximation provided via SVR involves an information loss which is about 1/5 compared with the loss information produced by LSMC. DLN further improves performance, reducing the ratio of information loss to 1/8.

Finally, the *NAV* at $t = 1$ year for each model under investigation has been computed using the *FDB*s distribution obtained by the corresponding fitting procedure, together with other *NAV* components produced directly by DISAR. Differences between the approximated densities and the benchmark one are plotted in Figure 4. Values close to 0 correspond to cases in which the fitted distribution roughly coincides with the reference (the model produces an almost perfect fitting). Positive values indicate that the approximated *PDF* is higher than the benchmark one (the model produces an overestimation in probability density) and, symmetrically, negative values reflect an underestimation in probability density. The figure suggests that DLN seems to perform better in the region of interest for the computation of *SCR*. Figure 5 shows the empirical *NAV* density computed with nested Monte Carlo and the three approximations, restricted to the left tail, which is the relevant one for *SCR* estimation. The results confirm that the empirical density obtained by DLN is closest to the nested Monte Carlo one, indicating that the "worst case value" estimate will be more precise for DLN than it will be for other models. This is further stressed in Table 6, showing the relative precision in the determination of the *SCR* achieved by the three models with respect to the one obtained with the nested Monte Carlo.

TABLE 6 Relative error in the SCR/$V(0)$ estimation obtained with LSMC, SVR and DLN with respect to the nested Monte Carlo benchmark

	Model		
	LSMC	**SVR**	**DLN**
Relative error	2.34%	1.88%	0.01%

TABLE 7 Summary of the datasets used in the experiments

Name	Outer simulations	Inner simulations
10Mx10	10,000,000	10
10Mx50	10,000,000	50
10Mx100	10,000,000	100

5.2 | Robustness of quantile estimation

The second set of tests is aimed to analyse the robustness of quantile estimation for all of the models under investigation. Despite the results obtained for a single quantile are not representative of the model performance at all quantiles, the analysis focuses on the 0.5% quantile since it is the one prescribed by the Directive.[1] In particular, an experimental goal is the assessment of the relative importance of the number of risk-neutral simulations on the variability of the quantile estimation.

To this end, simulations with a very large number of real-world scenarios have been carried out. However, the computational cost of such large simulations would have been prohibitive if the previous realistic portfolio was used. For this reason, two less complex insurance portfolios were considered: the first including only single-premium policies, while the second one containing only recurring-premium policies. In the single-premium policy, the insured makes a single payment to obtain the benefits established in the insurance contract. The recurring-premium policy provides for the payment of a sequence of premiums, each of which activates a new insurance policy and increases the benefits for the insured. In this sense, a recurring-premium policy can be considered as a portfolio of single-premium policies with different time-to-maturity. Obviously, the first kind of contracts is simpler than the second one, and the joint investigation of both portfolios allows us to evaluate the performance of these methods on different levels of complexity. The (relatively) simple structure of both portfolios allows a large number of real-world scenarios to be simulated with a reasonable computational cost.

For each portfolio, DISAR has been used to produce three datasets using nested Monte Carlo, varying the number of risk-neutral simulations as reported in Table 7.

The values of the quantile at 0.5% confidence level of the NAV used as a benchmark in all comparisons are obtained through a large size full nested Monte Carlo with $N = 100,000$ and $K = 1000$; the values are respectively -944.2871 for the first portfolio and -352.9184 for the second one.

Each dataset was partitioned into 1000 segments with 10,000 real-world samples each and the quantile was estimated separately on each segment. Considering each segment as an independent dataset, the distributions of the estimates for the models under investigation has been compared.

As discussed in Section 4, the pricing functions produced by LSMC, SVR and DLN need to be calibrated using a subset of data. The first of the 1000 segments is employed to calibrate the coefficients (weights) and the quantile is estimated in each one of the remaining splits. In this way, a sample of 999 different quantile estimates is obtained for all three methods. Differently, the nested MC approach performs the pricing via MC simulations and it does not require the estimation of the coefficients. In this case, the quantile is estimated in all splits and a sample of 1000 quantile estimations is obtained.

Tables 8 and 9 show, for all the methods considered, the number of splits, the split size, the mean quantile estimated, the relative error and standard deviation and Figure 6 depicts the empirical distributions obtained.

Some interesting evidence emerges. First, when few risk-neutral simulations are used, the nested MC estimator is biased. This result is visible in Figure 6 where can be noticed that the empirical density representing the nested MC (drawn in black) is far from the reference value for both portfolios. Furthermore, in Tables 8 and 9 it can be observed that the relative error produced by the nested MC estimator on the portfolio of recurring-premium policies is greater than the bias obtained on the portfolio of single-premium policies for the same number of risk neutral simulations. This issue appears to be consistent with the different degree of complexity of these two insurance portfolios. Second, the bias of full nested MC decreases, as expected, when more risk-neutral iterations are used. This result is visible in Figure 6, where

TABLE 8 Mean quantile estimations for portfolio with a single-premium policy

	No. of splits	Split size	Mean quantile	Relative error	Std. dev.
Nested MC					
10Mx10	1000	10,000	−983.2410	4.13%	8.7899
10Mx50	1000	10,000	−950.0097	0.61%	8.0738
10Mx100	1000	10,000	−945.7810	0.16%	7.8763
LSMC					
10Mx10	1000	10,000	−945.5961	0.14%	8.0690
10Mx50	1000	10,000	−941.4122	0.30%	7.8715
10Mx100	1000	10,000	−942.9744	0.14%	8.1700
SVR					
10Mx10	999	10,000	−941.5488	0.29%	7.9466
10Mx50	999	10,000	−942.6156	0.18%	8.0816
10Mx100	999	10,000	−942.3543	0.20%	8.0534
DLN					
10Mx10	999	10,000	−942.4480	0.19%	7.9192
10Mx50	999	10,000	−944.5563	0.03%	8.0798
10Mx100	999	10,000	−944.7542	0.05%	8.0913

TABLE 9 Mean quantile estimations for portfolio with a single recurring-premium policy

	No. of splits	Split size	Mean quantile	Relative error	Std. dev.
Nested MC					
10Mx10	1000	10,000	−378.5657	7.27%	4.4265
10Mx50	1000	10,000	−357.1506	1.20%	3.8907
10Mx100	1000	10,000	−354.3523	0.41%	3.7658
LSMC					
10Mx10	1000	10,000	−353.4090	0.14%	3.9809
10Mx50	1000	10,000	−351.4305	0.42%	3.7933
10Mx100	1000	10,000	−352.2006	0.20%	3.8314
SVR					
10Mx10	999	10,000	−353.0890	0.05%	3.8907
10Mx50	999	10,000	−352.0734	0.24%	3.8510
10Mx100	999	10,000	−352.2680	0.18%	3.8642
DLN					
10Mx10	999	10,000	−352.3674	0.16%	3.9660
10Mx50	999	10,000	−352.6624	0.07%	3.9250
10Mx100	999	10,000	−352.4008	0.15%	3.8407

it is observed that, when the number of risk-neutral scenarios K increases, the distributions of the quantile of the *NAV* obtained by nested MC get close to the benchmark value.

In contrast, LSMC, SVR and DLN yield better results: the performances, in terms of relative error, are satisfactory (lower than 0.5%) for all values of risk-neutral simulations and the resulting empirical densities are centred on the reference values even when few risk neutral are used. This evidence seems to suggest that, similarly to the LSMC, it is possible to obtain a precise quantile estimation using SVR and DLN even when few risk neutral simulations have been employed for their calibration and this results works in both portfolios.

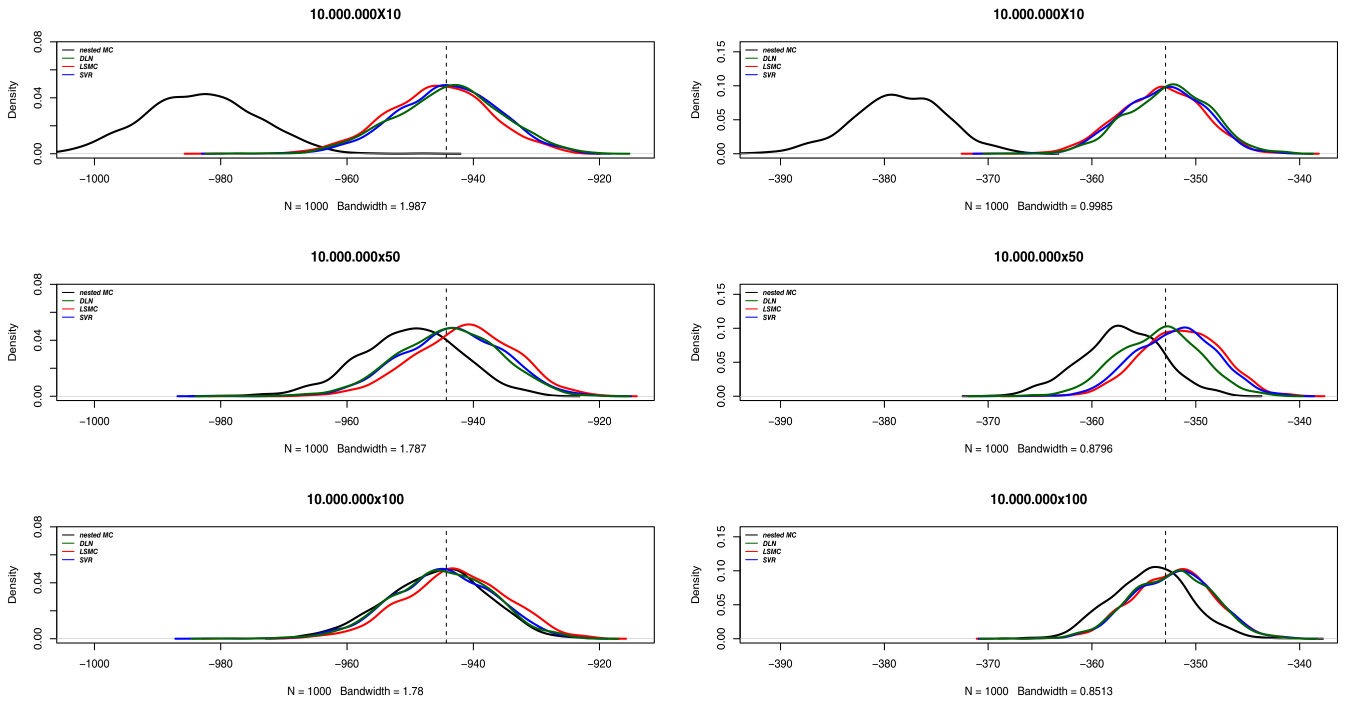


FIGURE 6 Quantile empirical probability distribution for nested MC, LSMC, SVR and DLN for portfolio with a single-premium policy (left) and a single recurring-premium policy (right) [Colour figure can be viewed at wileyonlinelibrary.com]

6 | CONCLUSIONS AND FURTHER WORK

Nested Monte Carlo in a Solvency II context is hardly viable due to its huge computational costs. The LSMC methodology, at present used by many practitioners, involves a difficult calibration process where complexity diverges when the number of risk drivers increases. In order to overcome these issues, the present article develops a procedure that makes use of machine learning techniques. Two variants are tested respectively based on the SVR and DLN algorithms and a comparative study against the LSMC is carried out.

Numerical experiments on some realistic insurance portfolios show that SVR and DLN are more accurate than traditional LSMC in describing the functional relation between the value of liabilities and the risk drivers. In addition, similarly to the LSMC, these techniques show strong robustness in 0.005 quantile estimation which is one of the most critical steps of the SCR calculation. More in details, DLNs seem to produce quite larger gain in terms of precision but they are more difficult to interpret. This evidence points out a large potential of deep learning techniques in insurance and probably suggests that more scientific effort should focus on a deep validation process of these methods and the interpretability of results. On the other hand, SVR shows a precision level between DLN and LSMC but seems to be much easier to use, having a very small number of hyperparameters to tune, and much simpler to understand for nonspecialists.

In conclusion, the findings obtained in this article suggest that ML techniques could represent promising tools for life insurance actuaries. In fact, an extensive use of these techniques is already done in the fields where a higher level of accuracy in predictive tasks can lead to a competitive advantage on the competitors.[34] A similar scenario may occur in the life insurance sector in the future since it is a highly competitive sector and the use of new technology by one of the market participants can represent an incentive for competitors.

In order to achieve a full and safe introduction of these techniques in real-world insurance practice, more scientific interest should be directed to the investigation of these techniques.

The literature in this field is far from being complete and some issues remains an open questions and deserve more attention.

Future studies, as well as exploring other simulation schemes,[35,36] should also involve the use of other machine learning techniques such as autoencoders and gradient boosting. Besides, the performance of these approaches should be tested on other insurance portfolios with different features, especially studying their potential in a large-dimensional risk setting. Finally, high-performance computing systems should be considered in order to reduce the computational times of training and tuning of these methods.

In addition, the scientific advances in machine learning for life insurance should also be accompanied by the training of highly qualified professional figures able to handle these powerful but delicate tools and to explain the decisions taken and the results obtained.

ACKNOWLEDGMENT

The authors thank the editor and anonymous referees whose comments helped to improve the manuscript.

DATA AVAILABILITY STATEMENT

Research data are not shared due to commercial restrictions.

ORCID

Francesca Perla https://orcid.org/0000-0002-4671-3917

REFERENCES

1. Directive 2009/138/EC of the European parliament and of the Council of 25 November 2009 on the taking-up and pursuit of the business of insurance and reinsurance. *Offic J Europ Union*; 2009. L 335.
2. Casarano G, Castellani G, Passalacqua L, Perla F, Zanetti P. Relevant applications of Monte Carlo simulation in solvency II. *Soft Comput*. 2017;21(5):1181-1192.
3. Gordy MB, Juneja S. Nested simulation in portfolio risk measurement. *Manag Sci*. 2010;56(10):1833-1848.
4. Bauer D, Reuss A, Singer D. On the calculation of the solvency capital requirement based on nested simulations. *ASTIN Bull J IAA*. 2012;42(2):453-499.
5. Floryszczak A, Le Courtois O, Majri M. Inside the solvency 2 black box: net asset values and solvency capital requirements with a least-squares Monte-Carlo approach. *Insur Math Econom*. 2016;71:15-26.
6. Richman R. AI in actuarial science. MPDI; 2018. https://ssrn.com/abstract=3218082.
7. Xu Q, Liu X, Jiang C, Yu K. Nonparametric conditional autoregressive expectile model via neural network with applications to estimating financial risk. *Appl Stoch Models Bus Ind*. 2016;32(6):882-908.
8. Heaton J, Polson N, Witte JH. Deep learning for finance: deep portfolios. *Appl Stoch Models Bus Ind*. 2017;33(1):3-12.
9. Gabrielli A, Wüthrich MV. An individual claims history simulation machine. *Risks*. 2018;6:29.
10. Banks D. Discussion of "machine learning application in non-life insurance". *Appl Stoch Models Bus Ind*. 2020;36(4):538-540. https://dx.doi.org/10.1002/asmb.2537.
11. Baudry M, Robert CY. A machine learning approach for individual claims reserving in insurance. *Appl Stoch Models Bus Ind*. 2019;35(5):1127-1155.
12. Wüthrich MV, Buser C. Data analytics for non-life insurance pricing. Swiss Finance Institute Research Paper No. 16-68, 2019. https://ssrn.com/abstract=2870308.
13. Krah AS, Nikolić Z, Korn R. Machine learning in least-squares Monte Carlo proxy modeling of life insurance companies. *Risks*. 2020;8(1):21.
14. Nilsson M, Sandberg E. Application and evaluation of artificial neural networks in solvency capital requirement estimations for insurance products; 2018.
15. Hejazi SA, Jackson KR. A neural network approach to efficient valuation of large portfolios of variable annuities. *Insur Math Econom*. 2016;70:169-181.
16. Hejazi SA, Jackson KR. Efficient valuation of SCR via a neural network approach. *J Comput Appl Math*. 2017;313:427-439.
17. Broadie M, Du Y, Moallemi CC. Efficient risk estimation via nested sequential simulation. *Manag Sci*. 2011;57(6):1172-1194.
18. Bauer D, Bergmann D, Kiesel R. On the risk-neutral valuation of life insurance contracts with numerical methods in view. *ASTIN Bull J IAA*. 2010;40(1):65-95.
19. Euro-Par Workshops. 6586 of Lecture Notes Computer ScienceCastellani G, Passalacqua L. Applications of distributed and parallel computing in the solvency ii framework: the DISAR system. In: Guarracino M, ed. New York, NY: Springer; 2010:413-421.
20. McNeil AJ, Frey R, Embrechts P. *Quantitative Risk Management: Concepts, Techniques and Tools*. Princeton, NJ: Princeton University Press; 2015.
21. Teuguia ON, Ren J, Planchet F. *Internal Model in Life Insurance: Application of Least Squares Monte Carlo in Risk Assessment. Technical Report 12*. Lyon, France: Laboratoire de Sciences Actuarielle et Financière; 2014.
22. Longstaff FA, Schwartz ES. Valuing American options by simulation: a simple least-squares approach. *Rev Financ Stud*. 2001;14(1):113-147.
23. De Felice M, Moriconi F. Market based tools for managing the life insurance company. *ASTIN Bull J IAA*. 2005;35(1):79-111.
24. Mercer J. Functions of positive and negative type and their connection with the theory of integral equations. *Philos Trans Royal Soc Lond*. 1909;209:415-446.
25. Kuhn M, Johnson K. *Applied Predictive Modeling*. New York, NY: Springer; 2013.
26. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436-444.

27. Bengio Y, Courville A, Vincent P. Representation learning: a review and new perspectives. *IEEE Trans Pattern Anal Mach Intell*. 2013;35(8):1798-1828.
28. Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal approximators. *Neural Netw*. 1989;2(5):359-366.
29. Goodfellow I, Bengio Y, Courville A. *Deep Learning*. Cambridge, MA: MIT Press; 2016.
30. Schmidhuber J. Deep learning in neural networks: an overview. *Neural Netw*. 2015;61:85-117.
31. Kingma DP, Ba J. Adam: a method for stochastic optimization; 2014. arXiv preprint arXiv:1412.6980.
32. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. Paper presented at: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016. Las Vegas, NV; 2016:770-778.
33. Suykens JA, Van Gestel T, De Brabanter J. *Least Squares Support Vector Machines*. Singapore: World Scientific; 2002.
34. Kraus M, Feuerriegel S, Oztekin A. Deep learning in business analytics and operations research: models, applications and managerial implications. *Europ J Operat Res*. 2020;281(3):628-641.
35. Bielza C, Müller P, Insua DR. Decision analysis by augmented probability simulation. *Manag Sci*. 1999;45(7):995-1007.
36. Müller P, Berry DA, Grieve AP, Smith M, Krams M. Simulation-based sequential Bayesian design. *J Stat Plann Infer*. 2007;137(10):3140-3150.
37. Brigo D, Mercurio F. *Interest Rate Models Theory and Practice*. New York, NY: Springer; 2004.
38. Cox JC, Ingersoll JE, Ross SA. A theory of the term structure of interest rates. *Econometrica*. 1985;53(2):401-402.

How to cite this article: Castellani G, Fiore U, Marino Z, et al. Machine learning techniques in nested stochastic simulations for life insurance. *Appl Stochastic Models Bus Ind*. 2021;37:159–181. https://doi.org/10.1002/asmb.2607

APPENDIX A

For this work the economic scenarios have been simulated using the commercial computational system *DISAR*® (Dynamic Investment Strategy with Accounting Rules). The Economic Scenario Generator (ESG) embedded in DISAR allows to choose among a set of alternative stochastic processes, one for each relevant risk. In this work the segregated fund that has been considered is composed of 46 bonds, including fixed rate and inflation-indexed bonds, issued in three currencies, namely Euro, U.S. Dollars and British Pounds, and 12 stocks, all belonging to the Euro-zone market. Accordingly, the set of risks is composed of interest rate risk, inflation risk, exchange risk and equity risk. The corresponding models are listed in Table A1. Nominal interest rates have been described with a modified version of the Cox, Ingersoll and Ross (CIR) model, known as CIR++ model,[37] which allows perfect fitting of the term structure of interest rates observed on the market at the valuation date. In this model the spot rate $r(t)$ is the sum of a deterministic component $\phi(t)$ and a stochastic component $x(t)$ which follows a mean reverting square-root diffusion processes. The model is calibrated using spot interest rates derived, via the standard *bootstrap* procedure, from interest rate swaps (IRS's) market values. In the framework proposed by Cox, Ingersoll and Ross,[38] real interest rates are related to nominal interest rates via the stochastic version of the Fisher equation, which requires the modelling of inflation. In this work inflation is simulated using a lognormal diffusion process for the consumer prices, calibrated to the Euro-zone Harmonised Index of Consumer Prices (HICP). For bonds issued in currencies different from the Euro, exchange rate risk is described by a lognormal diffusion process. Furthermore, in order to match market prices, a deterministic idiosyncratic spread is added to describe the price of each bond. Finally, equity risk is described using a benchmark approach similar to the Capital Asset Pricing Model (CAPM), where the 12 stocks price evolution is driven by a single benchmark, identified as the segregated fund equity component itself (the weighted average of the 12 stocks). Following the Black and Scholes model the benchmark value is also described by a lognormal diffusion process.

The set of initial values of the risk drivers, *real-world* and *risk-neutral* parameters used in this work are listed in Table A2. All constraints imposed by the no-arbitrage requirement over the values of the risk-neutral parameters have been implemented (in this case Table A2 reports an equation rather than a numerical value).

The overall number of risk drivers is seven, namely the Euro, U.S. Dollar, British Pound spot rates, the Euro-Dollar and Euro-Pound exchange rates, the consumer price index and the equity benchmark. Since the dynamics of each risk driver is described by a diffusion process, a seven-dimensional Brownian motion $\boldsymbol{Z}(t) = (Z_{x_1}(t), Z_{x_2}(t), Z_{x_3}(t), Z_P(t), Z_{C_1}(t), Z_{C_2}(t), Z_{\mathrm{B}}(t))$ is used to describe the joint dynamics of the risk drivers. The correlation matrix of the multidimensional Brownian motion is reported in Table A3.

TABLE A1 Stochastic models for financial risks

Risk driver	Evolution	Model
Nominal risk-free spot rates	$r_i(t) = \phi_i(t) + x_i(t)$	
$(i=1, \dots, 3)$	$dx_i(t) = \alpha_i\ (\gamma_i - x_i(t))\ dt + \rho_i\ \sqrt{x_i(t)}\ dZ_{x_i}(t)$	CIR++ model
Nominal risk-free disc. factor	$v(t,s) = v_\phi(t,s) \cdot v_x(t,s)$	
Risky nominal disc. factors	$v_j(t,s) = v(t,s) \cdot e^{-\eta_j(s-t)}$	
Credit spread $(j=1, \dots, 46)$	Deterministic *idiosyncratic* spread η_j	
Real risk-free rates	$\bar{r}_1(t) = r_1(t) - \left(y(t) - \sigma_p^2\right)$	Stoch. Fisher eq.
Consumer prices	$dp(t) = y(t)\ p(t)\ dt + \sigma_p\ p(t)\ dZ_p(t)$	Lognormal model
Expected inflation	$y(t) = y_\infty + (y_0 - y_\infty)\ e^{-\alpha_y\ t}$	Deterministic
Equity benchmark	$dB(t) = B(t)\ \mu_B\ dt + B(t)\ \sigma_B\ dZ_B(t)$	Black–Scholes model
Equity prices $(i=1, \dots, 12)$	$dS_i(t) = S_i(t)\ \mu_S^i\ dt + S_i(t)\ (\beta_i\ \sigma_B)\ dZ_B(t)$	CAPM
Exchange rates $(i=1,2)$	$dC_i(t) = C_i(t)\ \mu_C^{(i)}\ dt + C_i(t)\ \sigma_C^{(i)}\ dZ_{C_i}(t)$	Lognormal model

TABLE A2 Set of initial values of the risk drivers, *real-world* and *risk-neutral* parameters used in this work

		Parameters				
Risk driver	**Initial value**	**Real word**			**Risk neutral**	
Interest rate model						
	$r(0)$	α	γ	ρ	$\hat{\alpha}$	$\hat{\gamma}$
$r_1(t)$—EUR	0.00119	0.12203	0.02266	0.06107	0.11402	0.02425
$r_2(t)$—GBP	0.00467	0.28128	0.02527	0.08670	0.28128	0.02527
$r_3(t)$—USD	0.00089	0.26315	0.03299	0.11040	0.26315	0.03299
Exchange rate model						
	$C(0)$	μ_C	σ_C		$\hat{\mu}_C$	
$C_1(t)$—GBP	0.77890	0	0.05829		$r_1(t) - r_2(t)$	
$C_2(t)$—USD	1.21000	0	0.05956		$r_1(t) - r_3(t)$	
Inflation model						
Risk-driver	$p(0)$	$y_\infty = y_0$	σ_p		$\hat{y}_t$	
$p(t)$—HICP	117.01	0.0198	0.01592		$y_\infty - \sigma_p^2$	
Equity model						
	$B(0)$	μ_B	σ_B		$\hat{\mu}_B$	
$B(t)$—equity portfolio	1	0.04	0.19000		$r_1(t)$	
	$S(0)$	μ_S	β		$\hat{\mu}_S$	
$S_1(t)$	27.645	0.02919	0.49047		$r_1(t)$	
$S_2(t)$	36.554	0.04850	1.25966		$r_1(t)$	
$S_3(t)$	19.330	0.04147	0.81070		$r_1(t)$	
$S_4(t)$	103.500	0.01574	0.44279		$r_1(t)$	
$S_5(t)$	52.704	0.04602	1.25966		$r_1(t)$	
$S_6(t)$	56.331	0.04703	1.25966		$r_1(t)$	
$S_7(t)$	42.515	0.03394	0.57949		$r_1(t)$	
$S_8(t)$	14.510	0.03605	0.66362		$r_1(t)$	
$S_9(t)$	4.098	0.03174	0.63840		$r_1(t)$	
$S_{10}(t)$	3.760	0.03179	0.61710		$r_1(t)$	
$S_{11}(t)$	19.430	0.03246	0.57949		$r_1(t)$	
$S_{12}(t)$	22.825	0.02753	0.49869		$r_1(t)$	

Note: Risk neutral parameters are denoted with the circumflex symbol "^".

TABLE A3 Correlation matrix between the risk drivers used in this work

	r_1	r_2	r_3	C_1	C_2	p	B
r_1	1	−0.05186	0.02021	0.05837	−0.01125	0.05483	0
r_2	−0.05186	1	−0.00618	0.20062	0.04160	0.13604	0
r_3	0.02021	−0.00618	1	0.07128	0.01737	−0.05338	0
C_1	0.05837	0.20062	0.07128	1	0.58425	0.01923	0
C_2	−0.01125	0.04160	0.01737	0.58425	1	−0.07146	0
p	0.05483	0.13604	−0.05338	0.01923	−0.07146	1	0
B	0	0	0	0	0	0	1