Annals of Actuarial Science (2022), **16**, pp. 95–118
doi:10.1017/S1748499521000075

PAPER

Tree-based models for variable annuity valuation: parameter tuning and empirical analysis

Zhiyu Quan[1], Guojun Gan[2*] and Emiliano Valdez[2]

[1]Department of Mathematics, University of Illinois at Urbana-Champaign, Champaign, IL 61801, USA and [2]Department of Mathematics, University of Connecticut, Storrs, CT 06269-1009, USA
*Corresponding author. E-mail: guojun.gan@uconn.edu

(Received 23 June 2020; revised 07 February 2021; accepted 10 February 2021; first published online 16 March 2021)

Abstract

Variable annuities have become popular retirement and investment vehicles due to their attractive guarantee features. Nonetheless, managing the financial risks associated with the guarantees poses great challenges for insurers. One challenge is risk quantification, which involves frequent valuation of the guarantees. Insurers rely on the use of Monte Carlo simulation for valuation as the guarantees are too complicated to be valued by closed-form formulas. However, Monte Carlo simulation is computationally intensive. In this paper, we empirically explore the use of tree-based models for constructing metamodels for the valuation of the guarantees. In particular, we consider traditional regression trees, tree ensembles, and trees based on unbiased recursive partitioning. We compare the performance of tree-based models to that of existing models such as ordinary kriging and generalised beta of the second kind (GB2) regression. Our results show that tree-based models are efficient in producing accurate predictions and the gradient boosting method is considered the most superior in terms of prediction accuracy.

Keywords: Tree-based model; Variable annuity; Portfolio valuation; Metamodelling

1. Introduction

A variable annuity (VA) is a tax-deferred retirement vehicle that is created by insurance companies to address concerns many people have about outliving their assets. VA policies typically contain guarantees, which include guaranteed minimum death benefit (GMDB), guaranteed minimum accumulation benefit (GMAB), guaranteed minimum income benefit (GMIB), and guaranteed minimum withdrawal benefit (GMWB) (Hardy, 2003). These guarantees provide policyholders downside protection during significant declines in the financial market. As a result, these are financial guarantees that cannot be adequately addressed by traditional actuarial approaches. For example, during a bear market when stock prices are falling, insurance companies can expect to lose large sums of money on their portfolios of VA policies.

Dynamic hedging is adopted by many insurance companies to mitigate the financial risks associated with VA guarantees. An important step of dynamic hedging is to quantify the risks, which involves calculating the fair market values of the guarantees. Since the guarantees are complex, their fair market values cannot be determined explicitly or in closed form. In practice, insurance companies resort to Monte Carlo simulation to calculate the fair market values of guarantees. While Monte Carlo simulation is flexible and can handle any types of guarantees, it is computationally intensive. Using Monte Carlo simulation to calculate the fair market values of a large portfolio of VAs can take days or weeks.

To speed up the valuation of VA portfolios based on Monte Carlo simulation, metamodelling techniques have been proposed in the past few years. See Gan & Lin (2015), Gan & Valdez (2017a), Hejazi *et al.* (2017), Gan (2018), Xu *et al.* (2018), Dang *et al.* (2019), Liu & Tan (2020), Gweon *et al.* (2020), Lin & Yang (2020), and Feng *et al.* (2020). Metamodelling techniques involve building a predictive model based on a small number of representative VA policies in order to reduce the number of policies that are valued by Monte Carlo simulation. Specifically, a metamodelling technique consists of the following four components:

1. Select a small number of representative VA policies;
2. Use Monte Carlo simulation to calculate the fair market values of the representative policies;
3. Build a predictive model, called a metamodel, based on the representative policies and their fair market values; and
4. Use the predictive model to estimate the fair market value for every VA policy in the portfolio.

Since only a small number of VA policies are valued by Monte Carlo simulation and the predictive model is much faster than Monte Carlo simulation, metamodelling techniques have the potential to reduce the valuation time significantly.

In the past, ordinary kriging (Gan, 2013), universal kriging (Gan & Lin, 2017), generalised beta of the second kind (GB2) regression model (Gan & Valdez, 2018), and neural networks (Hejazi & Jackson, 2016; Xu *et al.*, 2018) have been used to build various types of metamodels. Kriging is a family of estimators used to interpolate spatial data. Advantages of kriging methods include producing accurate aggregate results at the portfolio level and requiring only a few parameters to estimate. However, kriging methods have the disadvantages that they require a large number of distance calculations and assume normal distribution of the response variable. The GB2 regression model has the advantage that it can handle highly skewed data, but estimating parameters poses quite a challenge. Neural networks have several advantages: they can approximate any compactly supported continuous function arbitrarily well; they can model data that has non-linear relationships between variables; and they can handle interactions between variables. From a metamodelling perspective, however, we intend to select a small number of representative samples from the entire portfolio. Neural networks perform better by learning from a larger training dataset, which can be challenging for our purposes.

In this paper, we study the use of tree-based models to predict the fair market value of the guarantees embedded in a VA policy. Originating in early 1960s, tree-based models are data mining algorithms that repeatedly partition the space of the explanatory variables to create a tree structure in predicting the response variable. Using survey data, Morgan & Sonquist (1963) developed the very first naive regression tree algorithm called the Automatic Interaction Detection (AID). Nowadays, tree-based models have become an attractive alternative predictive tool for building classification and regression models. Tree-based models possess the following properties:

- Tree-based models are considered as non-parametric models, and thus do not require to specify the form of the explanatory variables to the response variable.
- Tree-based models can handle missing data automatically.
- Tree-based models can handle categorical variables naturally.
- Tree-based models can capture non-linear effects and handle interactions between variables.
- Tree-based models can perform and assess variable importance (Breiman *et al.*, 1984; Ishwaran, 2007).
- Tree-based models, especially single tree models, can be interpreted straightforwardly by visualising the tree structure.

These advantages have encouraged us to explore and demonstrate the promising use of tree-based models as an alternative to various metamodels for the efficient valuation of large portfolios of VA products. This paper provides additional details to enhance predictive performance of tree-based models, for example, hyperparameter tuning.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

This paper has been structured as follows. In Section 2, we discuss the concept of tree-based models and their extensions. Here, we also describe another framework of binary splitting generally called unbiased recursive partitioning; conditional inference trees and their extension fall in this category. Section 3 provides details of parameter tuning, which is an important process in building tree-based models that helps significantly improve prediction accuracy. In particular, methods used in this paper include cross-validation and optimisation. In Section 4, we summarise the synthetic dataset used in our empirical investigation and perform preliminary data exploration. Furthermore, we compare the performance of various tree-based models, as well as other metamodels, in terms of predictive accuracy and computational efficiency. A menu of tree-based models is presented for purposes of being comprehensive. Section 5 concludes the paper with some remarks.

2. Tree-based Models

In this section, we present several tree-based models. To that end, we let the response variable be denoted as $\mathbf{Y}$, the sample space as $\mathcal{Y}$ (which can be multivariate as well), and let n denote the number of observations. The ith sample with p-dimensional explanatory variables is denoted as $\mathbf{X}_i = (x_{i1}, x_{i2} \ldots, x_{ip})$, which is sampled from the space $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_p$. Each observation in the dataset can form a learning sample denoted by $\mathcal{L}_n$:

$$\mathcal{L}_n = \{\mathbf{Y}_i, x_{i1}, x_{i2} \ldots, x_{ip}\}_{i=1}^{n}$$

2.1. Classification and Regression Tree (CART)

The Classification and Regression Tree (CART) algorithm (Breiman *et al.*, 1984) uses greedy search called recursive binary partition to create a tree structure. The algorithm includes growing and pruning steps. It can easily handle missing data by utilising "surrogate" splitting, which finds an alternative explanatory variable that mimics the explanatory variable that contains missing values. The surrogate splitting process also provides variable importance at each node by measuring the decrease of the error function. Breiman *et al.* (1984) obtained conditions for all recursive partitioning techniques to be Bayes risk consistent. In conventional terms, the trees obtained by the algorithm are called classification trees when the response variable is categorical and are called regression trees when the response variable is continuous.

Here, we briefly formulate the process in the regression framework. For details, refer to Quan & Valdez (2018). The algorithm divides the explanatory variable space $\mathcal{X}$ into disjoint M regions, $R_1, R_2, \ldots, R_M$, and assigns a constant c_m as the predicted value for region R_m:

$$f(\mathbf{X}_i|\Theta) = T(\mathbf{X}_i; \Theta) = \sum_{m=1}^{M} c_m \mathbf{1}_{R_m}(\mathbf{X}_i)$$

where $\Theta = \{R_m, c_m\}_{m=1}^{M}$.

Under the sum of squared errors (SSE) loss function, the best or optimal $\hat{c}_m$ is the average of y_i in the region R_m. In other words, the algorithm optimises the following objective function $L(T)$ with penalty:

$$L(T) = \sum_{m=1}^{M} \sum_{\mathbf{x}_i \in R_m} (y_i - \hat{c}_m)^2 + \alpha M$$

The tuning parameter $\alpha \geq 0$ governs the trade-off between the size of the tree and its goodness of fit to the data. Estimation of α can be achieved by cross-validation, which is explained in a later section.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

2.2. Ensemble methods

Ensemble methods combine several models to help improve prediction accuracy. Two popular ensemble methods for regression trees are: (a) bagging and random forests and (b) gradient boosting.

2.2.1. Bagging and random forests

Bagging (Breiman, 1996) uses an ensemble of sufficiently deep CART trees, $\{T(\mathbf{X};\ \Theta_b),\ b = 1, 2, ..., B\}$, without pruning. These trees are built on B bootstrap samples of the training dataset and for prediction, we take the average of B regression trees to get the prediction:

$$f_B(\mathbf{X}) = \frac{1}{B}\sum_{b=1}^{B} T(\mathbf{X};\ \Theta_b)$$

Random forests (Breiman, 2001) reduce correlation between the CART trees by selecting the best split from a random subset of explanatory variables at each node of a tree. In effect, the average prediction of multiple regression trees is expected to have lower variance than a single individual regression tree. Larger random sets of explanatory variables can improve the predictive capability of individual trees, but it can also increase the correlation between trees and void any gains from averaging multiple predictions. The bootstrap resampling of the data for training each tree also increases the variation between the trees. The accuracy of random forests depends on the strength of each individual tree and a measure of the dependence between them.

2.2.2. Gradient boosting

Gradient boosting (Freund & Schapire, 1997), or sometimes called gradient boosted regression trees, grows trees by sequentially putting more weights on the residuals from previous trees. The boosted tree is a sum of such trees:

$$f_B(\mathbf{X}) = \sum_{b=1}^{B} T_b(\mathbf{X};\ \Theta_b)$$

where $T_b(\mathbf{X};\ \Theta_b)$ are the regression trees. B is the number of iterations or the number of additive trees. In each step b, for $b = 1, \ldots, B$, we need to find the regression Θ_b based on the following optimisation problem:

$$\hat{\Theta}_b = \arg\min_{\Theta_b} \sum_{i=1}^{N} L(y_i, f_{b-1}(\mathbf{X}_i) + T_b(\mathbf{X}_i;\ \Theta_b))$$

where $\Theta_b = \{R_{mb}, c_{mb}\}_{m=1}^{M}$.

The loss function L can be the squared error loss function. For other differentiable loss functions, the above optimal solution can be obtained using numerical optimisation via gradient boosting. See Friedman (2001).

An alternative boosting algorithm, which is not as attractive for our empirical dataset, is delta boosting. However, for generalised linear models, Lee & Lin (2018) demonstrated that delta boosting outperforms the gradient boosting algorithm using claims data on collision coverage for vehicle insurance from a Canadian insurer.

2.3. Unbiased recursive partitioning

CART algorithms described above employ recursive binary partition. This greedy search causes some drawbacks. One drawback is overfitting, which can be resolved using a pruning process by

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

applying cross-validation. Another drawback is the resulting bias in variable selection, especially when the explanatory variables present many possible splits or missing values. This latter drawback is harder to remedy. Strobl *et al*. (2009) illuminated the bias in variable selections with many categorical and numerical variables or, even more unintuitively, many missing values. These explanatory variables are artificially preferred in recursive binary partition algorithms. See Strobl *et al*. (2007). This evidence alerts us to investigate variable selection and interpretations in VA valuation, especially when many benefit riders are present.

2.3.1. Conditional inference trees

As a remedy for the bias drawback, Hothorn *et al*. (2006) introduced a conditional inference framework for unbiased recursive partitioning, which has stopping criteria based on the statistical permutation test in Strasser & Weber (1999). Unbiased recursive partitioning framework can be applied to univariate continuous or discrete regression, censored regression, classification, ordinal regression, and multivariate regression. In this paper, we focus on univariate continuous regression. The conditional inference trees algorithm has been shown that the predictive accuracy is comparable to that produced by CART.

Guelman *et al*. (2014) applied conditional inference trees to personalised cross-sell marketing of insurance products, by building personalised treatment learning to select profitable policyholders. In particular, the objective of the marketing campaign was to determine the group of customers with existing auto insurance policies that should be optimally targeted to be offered an additional home insurance policy.

The main difference between CART and conditional inference trees algorithm is the variable selection at each split and the stopping criterion. To explain the details, we need to define a few terms. The conditional distribution of a statistic, $F(\mathbf{Y}|\mathbf{X})$, measures the association between the response variable and the explanatory variables.

It is possible that some explanatory variables x_{ij} are missing in real life data. The tree-based model ultimately finds membership for observations and assigns them to the terminal node, R_m. To define this membership for each observation, we introduce the vector of case weights denoted by $\mathbf{w} = (w_1, \ldots, w_n)$. Hence, for each terminal node, R_m, we have a vector of case weights $\mathbf{w} = (w_1, \ldots, w_n)$. The weight, w_i, can be any positive value if it is an observation in the specific terminal node. Without loss of generality, we can restrict the weight value to either one or zero. Define symmetric group, $S(\mathcal{L}_n, \mathbf{w})$, as all possible permutations of observations with weight one. For example, if $w_1 = 1, w_2 = 0, w_3 = 1, w_4 = w_5 = \ldots = w_n = 0$, then $S(\mathcal{L}_n, \mathbf{w}) = \{(\mathbf{Y}_1, x_{11}, x_{12}, \ldots, x_{1p}), (\mathbf{Y}_3, x_{31}, x_{32}, \ldots, x_{3p}), (\mathbf{Y}_3, x_{11}, x_{12}, \ldots, x_{1p}), (\mathbf{Y}_1, x_{31}, x_{32}, \ldots, x_{3p})\}$.

We now briefly describe unbiased recursive binary splitting. First, under a specific vector of case weights, apply statistical hypothesis test to determine if there is any dependency between the response variable and the explanatory variables. If there is a dependency, then find the most significant (strongest association) explanatory variable to perform the split and update the case weights. If there is no dependency, then stop the process.

In formulating the process, the null hypothesis is that all the explanatory variables are independent of the response variable

$$H_0 = \cap_{j=1}^{p} H_0^j$$

where $H_0^j : F(\mathbf{Y}|\mathbf{X}_{\cdot j}) = F(\mathbf{Y})$. Then use the conditional distribution of linear statistics in the permutation test (Strasser & Weber, 1999). Under H_0 and given all permutations of the response variable, the conditional expectation $\mu_j \in \mathbb{R}^{pjq}$ can be derived as follows (Hothorn *et al*., 2006):

$$\begin{aligned} \mu_j &= \mathbb{E}(T_{\mathbf{X}_j}(\mathcal{L}_n, \mathbf{w}) | S(\mathcal{L}_n, \mathbf{w})) \\ &= vec\left(\left(\sum_{i=1}^{n} w_i g_j(x_{ij}) \right) \mathbb{E}(h | S(\mathcal{L}_n, \mathbf{w}))^T \right) \end{aligned} \tag{1}$$

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

where

$$\mathbb{E}(h|S(\mathcal{L}_n, \mathbf{w})) = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i h(Y_i, (Y_1, Y_2, \ldots Y_n))$$

The linear statistic, $T_{X_\cdot}(\mathcal{L}_n, \mathbf{w})$, can be standardised since $T_{X_\cdot}(\mathcal{L}_n, \mathbf{w}) - \mu_\cdot$ is asymptotically normal under H_0 and conditioned on the symmetric σ-fields $S(\mathcal{L}_n, \mathbf{w})$. See Strasser & Weber (1999). There are a few ways of standardisation (Hothorn *et al.* (2006)). One way is based on the maximum of the absolute values of the standardised linear statistics defined by

$$z_{\max}(T_{X_\cdot}, \mu_\cdot, \Sigma_\cdot) = \max_{k=1,2,\ldots,pq} \left| \frac{(T_{X_\cdot} - \mu_\cdot)_k}{diag(\Sigma_\cdot)^{1/2}} \right|$$

Another way is based on the quadratic form defined by

$$z_{quad}(T_{X_\cdot}, \mu_\cdot, \Sigma_\cdot) = (T_{X_\cdot} - \mu_\cdot)\Sigma_\cdot^{+}(T_{X_\cdot} - \mu_\cdot)^T$$

where $\Sigma_\cdot^{+}$ is the Moore–Penrose inverse of $\Sigma_\cdot$ which may be computationally intensive. The Moore–Penrose inverse is the most common type of pseudoinverse. See Moors (1920) and Penrose (1955). For the case of the quadratic form, the test statistic asymptotically follows a chi-squared distribution with $rank(\Sigma_\cdot)$ degrees of freedom.

Since the test statistic $z(T_{X_\cdot}, \mu_\cdot, \Sigma_\cdot)$ cannot be compared directly due to possibly difference of scale in the explanatory variables, we use the p-value to choose the most significant explanatory variable. We select explanatory variable, X_{j^*}, that has the smallest p-value, $P_{j^*} = \arg\min_{j=1,\ldots,p} P_j$, where

$$P_j = \mathbb{P}_{H_0^j}(z(T_{X_j}, \mu_j, \Sigma_j) \geq z(t_{X_j}, \mu_j, \Sigma_j)|S(\mathcal{L}_n, \mathbf{w}))$$

and $t_{X_j} \in \mathbb{R}^{p_j q}$ is the observed test statistic from the dataset.

After picking the best explanatory variable X_{j^*}, we then find the best splitting point according to the splitting criterion. This criterion can be a simple binary split like CART, or a multiway split as in O'Brien (2004).The permutation test framework can be used to find the best split point.

Given a predefined significance level, α, if the null hypothesis, H_0, cannot be rejected, then stop the binary splitting process. Here, the p-value for the null hypothesis can be calculated using the Bonferroni-adjusted p-value$(1 - (1 - P_j)^p)$ or min-p-value resampling. For more advanced multiple testing, we refer the reader to Westfall & Young (1993).

The level of significance, α, controls the tree size. It can be tuned using cross-validation and this process can be similar to tree pruning in CART. In detail, one would initially set higher significance level, α, which leads to a larger tree, and then prune the terminal node by setting a lower significance level, $\alpha^* \leq \alpha$. The significance level can be predetermined according to a tolerance based on some actuarial expertise. Moreover, the significance level can be interpreted in the traditional statistical sense of balancing the Type I and Type II errors.

2.3.2. Conditional random forests

Random forests variable importance provides essential interpretation for the ensemble tree-based models although it is not reliable to perform variable selection according to variable importance scores. This is especially true for cases where the explanatory variables vary in the scale of measurement or number of categories. See Strobl *et al.* (2007). On the other hand, conditional random forests are created based on conditional inference trees that can provide unbiased variable selection.

The framework of conditional random forests is very similar to that of random forests. The conditional inference trees are built on the bootstrap sample or subsample of the original dataset. The random subset of the explanatory variables is considered at each split. However, there are few

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

differences between conditional random forests and random forests. First, after building all the conditional inference trees, the final ensemble model averages the observation weights extracted from each tree. It is a different aggregation scheme from the random forests which simply averages the prediction. Second, conditional random forests can extensively model censored, multivariate, as well as ordered response variable. Third and finally, when explanatory variables vary in their scale of measurement and the number of categories, which is very typical of a dataset, the conditional random forests can provide unbiased variable selection and variable importance based on conditional inference trees that use subsamples without replacement.

3. Parameter Tuning of Tree-based Models

As noted in the previous section, the tree-based model hyperparameter tuning (optimisation) usually involves cross-validation to perform model selection. Table 1 lists some common tuning hyperparameters for tree-based models. The unbiased recursive binary splitting framework can be implemented in both `party` and `partykit`.

3.1. Cross-validation

Cross-validation was introduced to fix the overoptimistic prediction accuracy on the same dataset that model trained. In other words, it is an attempt to avoid overfitting. See Mosteller & Tukey (1968), Stone (1974), and Geisser (1975).

Cross-validation is a way to examine tree-based model performance on hypothetical validation dataset when an exact validation set is not available. In detail, the dataset is separated into two disjoint parts: training dataset and validation dataset. The model is built on the training dataset and the prediction performance is examined on the validation dataset.

Various splitting strategies lead to different cross-validation techniques. Data splitting requires certain assumptions. For example, data are identically distributed and there must be independence between training dataset and validation dataset. If these assumptions are not satisfied, then some modifications are needed for the cross-validation. See Opsomer *et al.* (2001) and Leung (2005). Usually for insurance datasets, time dependency and outliers are present. It requires extra attention when we apply general cross-validation to perform model selection.

In the holdout method (Devroye & Wagner, 1979), the dataset is separated only once, and typically the validation dataset is smaller than the training dataset. The holdout method can be considered as the simplest cross-validation. While the holdout method suffers from a drawback that the results highly depend on data split, it is because the data in the validation dataset may have important information and this information is left out when we train model. In other words, the data segment easily leads to bias in the result. To deal with this issue, other cross-validation is discussed later.

Unlike the holdout method, other cross-validation generally splits dataset several times and averages prediction accuracy on validation dataset. The question arises on how to split the dataset. There are two types of splitting: exhaustive data splitting and partial data splitting.

3.1.1. Exhaustive cross-validation

Exhaustive cross-validation methods contain training and testing on all possible ways to divide the original dataset into a training dataset and a validation dataset. See Stone (1974), Allen (1974), Geisser (1974), and Shao (1993). There are two popular exhaustive cross-validation:

- Leave-one-out. Each data point is withdrawn from the original dataset and used as validation data. Obviously, it needs n runs. Leave-one-out cross-validation provides nearly unbiased estimation while it suffers from high variance.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

Table 1. R packages for tree-based models with their tuning hyperparameters

R package	Description
`rpart`	Classification and regression tree (CART)
cp	Complexity parameter
minsplit	Minimum number of observations in a node in order to be considered for splitting
maxdepth	Maximum depth of any node of the final tree
`randomForest`	Bagging and random forests
mtry	Number of explanatory variables randomly sampled as candidates at each split
nodesize	Minimum number of observations in the terminal nodes
ntree	Number of trees to grow/bootstrap samples
`gbm`	Gradient boosting
n.trees	Number of trees to fit/iterations/basis functions in the additive expansion
interaction.depth	Maximum depth of variable interactions(1 implies an additive model, 2 means a model with up to two-way interactions)
n.minobsinnode	Minimum number of observations in the terminal nodes
shrinkage	Shrinkage parameter(learning rate or step size reduction)
`party/partykit`	Conditional inference trees
teststat	Type of the test statistic to be applied for variable selection
splitstat	Type of the test statistic to be applied for split point selection
testtype	The way to compute the distribution of the test statistic
alpha	Significance level for variable selection
minsplit	Minimum sum of weights in a node in order to be considered for splitting
`party/partykit`	Conditional random forests
mtry	Number of explanatory variables randomly sampled as candidates at each split
ntree	Number of trees to grow/bootstrap samples

- Leave-p-out. Here p data points are withdrawn from the original dataset and it takes $\binom{N}{k}$ runs. In practice, for large dataset, exhaustive cross-validation is computationally intensive.

3.1.2. Non-exhaustive cross-validation

Non-exhaustive cross-validation apply partial data splitting. One well-known non-exhaustive cross-validation is k-fold cross-validation (Geisser, 1975). In k-fold cross-validation, the original dataset is segmented into k equal sized subsamples. In each run, one of the k subsamples is held out as validation dataset and model is built on the remaining $k-1$ subsamples. In total, there are k runs with each subsample selected only once as validation dataset. This process assures all the data points have a chance to belong in the training dataset and the validation dataset. This method takes the average of the prediction results on the k validation datasets.

When k is equal to the number of the dataset, the k-fold cross-validation becomes the leave-one-out cross-validation. It is an open question to choose the best k for the k-fold cross-validation. If the number k is selected, then the size of the training set is fixed as well as the number of splits. The bias of the k-fold cross-validation decreases with k since larger k leads to a larger training set. On the other hand, the variance of the k-fold cross-validation increases with k since a larger k leads to a larger number of validation procedure (i.e. more runs). Practically, the number k is

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

often chosen to be between 5 and 10. Ten-fold cross-validation has been shown to provide good model selection and point estimation. See Kohavi (1995).

Additional cross-validation techniques and statistical properties of cross-validation are discussed in Arlot & Celisse (2010) and Leung (2005).

3.2. Hyperparameter optimisation

3.2.1. Grid search

Grid search is the most traditional way to optimise hyperparameters. It is an exhaustive searching process on the manually specified subset of hyperparameter space. The selection of the best combination of hyperparameters is usually based on the cross-validation. Determining the subset involves setting bounds and discretisation that requires expertise on the model. Grid search is computationally intensive due to the explosive number of combinations of hyperparameters. For example, if one model has p parameters and each parameter has c choices, then the subset contains c^p combinations. Fortunately, grid search is essentially parallel since evaluating each combination of the hyperparameters is independent of each other. This makes grid search feasible given enough computational power.

3.2.2. Random search

Random search is to randomly search the grid of hyperparameters that is manually specified and it has similar performance compared to the grid search. It can outperform the grid search given the same computation constraint and time, especially when only a small number of hyperparameters affects the final performance of the model (Bergstra & Bengio, 2012). In other words, if the close-to-optimal region of hyperparameters occupies at least 5% of the search space, then a random search with a certain number of trials (typically 40–60 trials) will be able to find that region with high probability. Like grid search, it can be made parallel. When compared to grid search, random search requires less number of trials and allows including prior knowledge of how to sample.

3.2.3. Bayesian optimisation

Bayesian hyperparameter tuning establishes knowledge about the association between the hyperparameter settings and model performance tos improve selection for the next hyperparameter settings. Unlike previously discussed tuning methods, the selection of the following hyperparameter settings is no longer independent of the prior selection. In other words, it is sequential. Hence it cannot be easily made parallel. The hyperparameter tuning becomes an optimisation problem. There are several sequential global optimisation methods for finding the hyperparameter setting that maximise the model generalisation performance. One of the most popular techniques is Bayesian optimisation. Bayesian optimisation models generalisation performance as a sample from a Gaussian Process (GP) (Snoek *et al*., 2012), and creates a regression model to formalise the relationship between the model performance and the model hyperparameters.

Specifically, let function $f: \mathcal{H} \to \mathbb{R}$ model hyperparameter setting $h \in \mathcal{H}$ to a prediction accuracy of

$$h_{optimal} = \arg\max_{h \in \mathcal{H}} f(h)$$

within a domain $h \in \mathbb{R}^d$ which is a bounding box. d is the number of tuning hyperparameters. The function f is a realisation of a GP with mean μ and covariance kernel K, i.e., $f \sim \mathcal{GP}(\mu, K)$. The Bayesian optimisation assumes the prediction follows the normal distribution. After choosing the kernel function K, we can compute the mean and variance for this normal distribution. For further details, see Snoek *et al*. (2012) and Martinez-Cantin (2014).

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

There are also a few other automatic hyperparameter optimisation techniques: gradient-based, evolutionary and those based on tree-structured Parzen estimator. Gradient-based optimisation computes the gradient with respect to hyperparameters and then optimises the hyperparameters using gradient descent. See Bergstra *et al.* (2011) and Larsen *et al.* (1996).

4. Applications to VA Valuation

In this section, we demonstrate the application of tree-based models to address a computational problem arising from the valuation of variable annuity products. We use the hierarchical k-means algorithm (Nister & Stewenius, 2006) to select the representative VA policies.

4.1. Data description

The dataset is a synthetic dataset that contains 190,000 VA policies. For details, see Gan & Valdez (2017b). We select 680 VA contracts from the dataset as training dataset, select 340 VA contracts as validation dataset and preform prediction on the 190,000 VA contracts. We select validation dataset to speed up the hyperparameter tuning process and model selection. Some summary statistics of the training dataset are provided in Table 2.

4.2. Hyperparameter optimisation results

As shown in Table 6, the tree-based models have less computational time. Everything comes with a price. Since building the tree-based models needs to perform hyperparameter tuning, it involves more techniques than traditional statistics methods. As mentioned in Section 3.2, hyperparameter optimisation can be achieved with three schemes. In this section, we provided the grid search results, which provide the best prediction results amongst the three schemes. For results produced by the other two efficient optimisation schemes, readers are referred to Appendix A.

For all the model calibration, we train the model on the training dataset (680 VA contracts). Since the dataset is not large, it allows us to perform grid search and ten-fold cross-validation. All the hyperparameters mentioned below are resulted from the grid search method.

For the regression tree, we grow a full tree using recursive binary splitting with a minimum number of five observations in a region for a split to be attempted. Then we prune this fully grown tree using cost-complexity pruning with "one standard deviation rule" regularisation parameter of 1.084e-02.

In detail, as mentioned in Table 1, we tune two common tuning hyperparameters "cp" and "minsplit" for the regression tree base on the training dataset using ten-fold cross-validation. As described in Section 3.2.1, the manually specified subset of hyperparameter space is the combination of "cp" which has ranged from 0.001 to 1 by the increment of 0.001 and "minsplit" which has ranged from 2 to 10 by the increment of 1. This subset has 9,000 combinations. With ten-fold cross-validation, the computation is heavy while it is feasible since our training dataset is small. We choose the optimal hyperparameter setting based on the minimum cross-validation error. The grid search results are shown in Table 3. Then compare prediction accuracy between the pruned tree which has minimum cross-validation and the pruned tree with "one standard deviation rule" on the validation dataset. Finally, we find the optimal hyperparameter setting as mentioned above. For the rest of the tree-based models, we will not report the detailed process of the grid search.

Figure 1 shows the regression tree plot. The first split is the product type and the following splits use the guaranteed benefit amount. The product type is the most important variable for the regression tree. At each split, if observations meet the split condition, then the algorithm assigns them to the left nodes, otherwise, the algorithm assigns them to the right. In each node, we have information about the number of observations and the prediction. The darker green colour means

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

Table 2. Summary statistics of the variables for the training dataset. Dollar amounts are in thousands ('000s)

Response variables	Description	Min.	First Q	Mean	Median	Third Q	Max.
fmv	Fair market value	−68.37	−5.55	64.63	11.7	64.84	1210.32
Continuous variables	Description						
gmwbBalance	GMWB balance	0	0	27.8	0	0	422.26
gbAmt	Guaranteed benefit amount	51.88	183.98	323.29	306.89	437.36	920.62
FundValue1	Account value of the first fund	0	0	32.02	12.62	46.76	629.89
FundValue2	Account value of the second fund	0	0	36.54	16.08	56.31	571.59
FundValue3	Account value of the third fund	0	0	26.78	11.81	36.64	458.78
FundValue4	Account value of the fourth fund	0	0	25.8	10.48	38.29	539.36
FundValue5	Account value of the fifth fund	0	0	22.29	10.54	34.71	425.92
FundValue6	Account value of the sixth fund	0	0	37.15	19.64	53.96	654.64
FundValue7	Account value of the seventh fund	0	0	28.78	12.88	42.56	546.89
FundValue8	Account value of the eighth fund	0	0	31.27	15.59	46.24	529.57
FundValue9	Account value of the ninth fund	0	0	31.93	13.9	45.17	599.44
FundValue10	Account value of the 10th fund	0	0	32.6	13.86	45.09	510.43
age	Age of the policyholder	34.52	42.86	50.29	51.36	57.21	64.46
ttm	Time to maturity in years	0.75	10.09	14.61	14.6	19.12	27.52
Categorical variables	Description						Proportions(%)
gender.M	Male policy holder						64.71
gender.F	Female policy holder						35.29
productType.ABRP	Indicate type GMAB with return of premium						8.82
productType.ABRU	Indicate type GMAB with annual roll-up						4.26
productType.ABSU	Indicate type GMAB with annual ratchet						6.03
productType.DBAB	Indicate type GMDB + GMAB with annual ratchet						5.00
productType.DBIB	Indicate type GMDB + GMIB with annual ratchet						5.88
productType.DBMB	Indicate type GMDB + GMMB with annual ratchet						5.74
productType.DBRP	Indicate type GMDB with return of premium						4.85
productType.DBRU	Indicate type GMDB with annual roll-up						6.62
productType.DBSU	Indicate type GMDB with annual ratchet						4.41
productType.DBWB	Indicate type GMDB + GMWB with annual ratchet						4.41
productType.IBRP	Indicate type GMIB with return of premium						5.74
productType.IBRU	Indicate type GMIB with annual roll-up						4.71
productType.IBSU	Indicate type GMIB with annual ratchet						4.85
productType.MBRP	Indicate type GMMB with return of premium						4.56
productType.MBRU	Indicate type GMMB with annual roll-up						5.29
productType.MBSU	Indicate type GMMB with annual ratchet						5.29
productType.WBRP	Indicate type GMWB with return of premium						4.12
productType.WBRU	Indicate type GMWB with annual roll-up						3.97
productType.WBSU	Indicate type GMWB with annual ratchet						5.44

the larger the value of the prediction. We can also observe that the following nodes depend on the previous splits and this is how the tree-based model detects interaction automatically.

Figure 2 shows the out-of-bag (OOB) error according to the number of trees in bagged trees. We can see that the OOB error stabilises after approximately 150 number of trees. Based on our experience, the OOB error stabilises after 200–300 number of trees for the most of the datasets.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

Table 3. Grid search results

	cp	minsplit	MinXerror
1	0.004	5	0.1126494
2	0.001	5	0.1139710
3	0.002	5	0.1139710
4	0.003	5	0.1139710
5	0.004	6	0.1159061
6	0.001	2	0.1161909
7	0.001	3	0.1161909
8	0.004	7	0.1168061
9	0.001	6	0.1172277
10	0.002	6	0.1172277

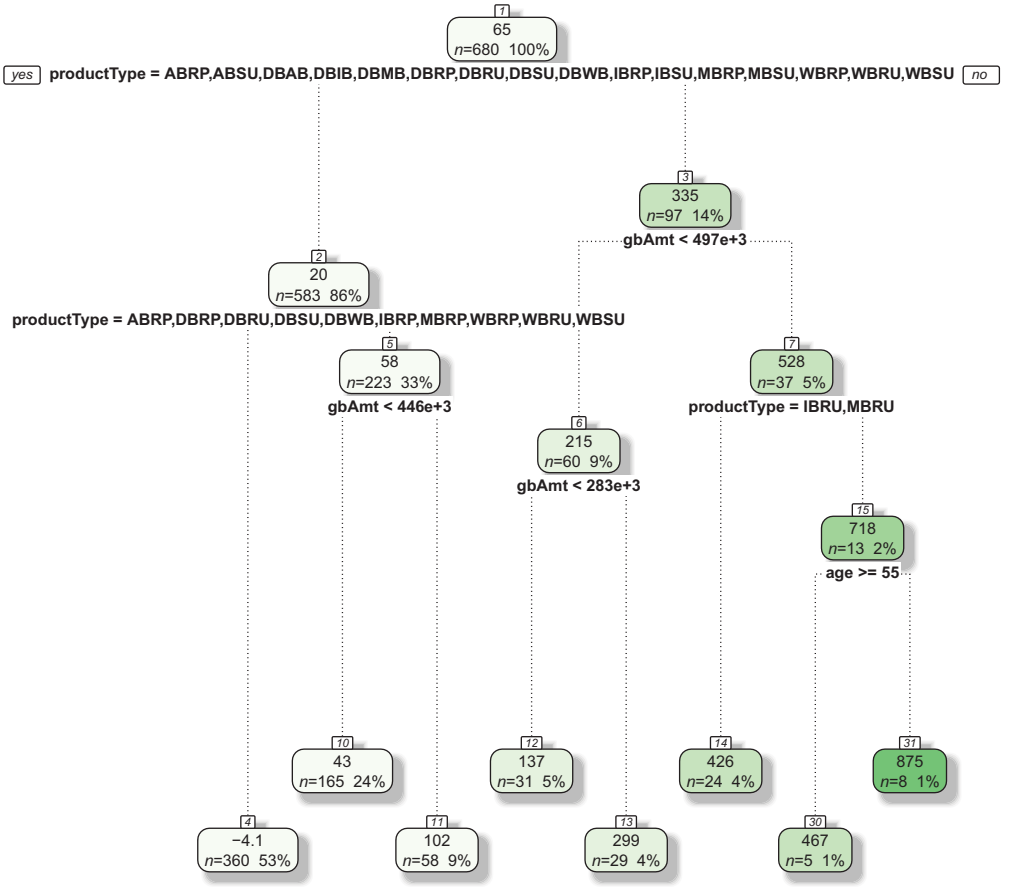

Figure 1. A regression tree. Dollar amounts are in thousands ('000s).

The bagged trees grows individual trees on the 200 bootstrap samples with 5 minimum numbers of observation at terminal nodes.

Figure 3 shows that the test error and the OOB error is minimised at 16, suggesting using all the explanatory variables at each split. In other words, the optimal random forests model is same as bagging.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

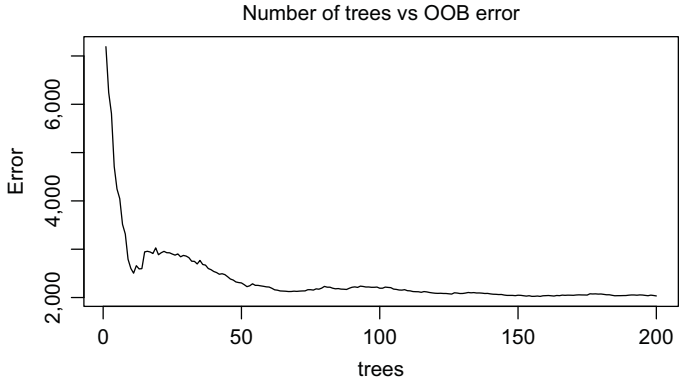

Figure 2. Determining the optimal number of trees in bagged trees.

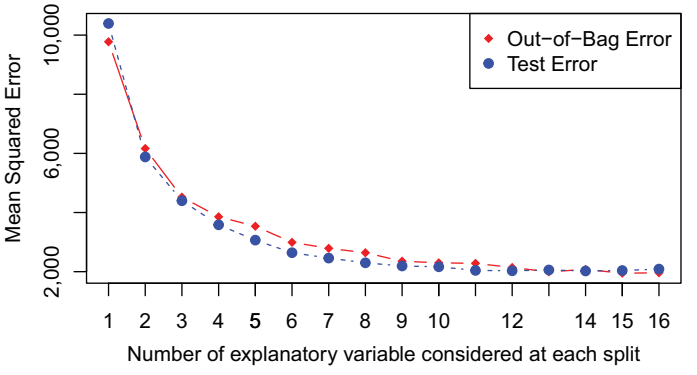

Figure 3. Random Forests tuning with the number of random subsets (m_{try}).

In this case, random forests is equivalent to bagged trees. Indeed, such a result may appear quite surprising. One way to explain the outcome is the dataset was synthetic dataset which used all the explanatory variables to create response variable, which is described in Gan & Valdez (2017b). Each tree requires using the full set of explanatory variables to achieve higher prediction accuracy. Another way to explain the outcome is missing important explanatory variables like the product type and guaranteed benefit amount at each split may decrease the prediction accuracy of random forests. However, as we can see in Figure 3, the marginal decrease in error after 8 of the random subset is minimal.

The final tuned gradient boosted regression trees has 1,000 iterations, 5 maximum depth of variable interactions, 5 minimum numbers of observations in the terminal nodes, and a learning rate of 0.1.

The conditional inference trees applies quadratic test statistics to perform the variable selection and choose the split point, applies Bonferroni adjustments to compute the distribution of the test statistic, and sets 0.04 as the significance level for variable selection with five minimum numbers of observations in the terminal nodes.

Figure 4 shows the simplified conditional inference tree plot. In this figure, the conditional inference tree is held at the maximum depth of three for the convenience of visualisation. The final model has indeed more branches and terminal nodes. However, it has the identical structure with this simplified conditional inference trees for the first few splits. Like the regression tree, the conditional inference tree also uses the product type at the first split. At each node, it shows the selected explanatory variable and its significance level. It has a similar structure to the regression tree, but it can provide the box plot at the terminal nodes thereby allowing the decision maker to

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

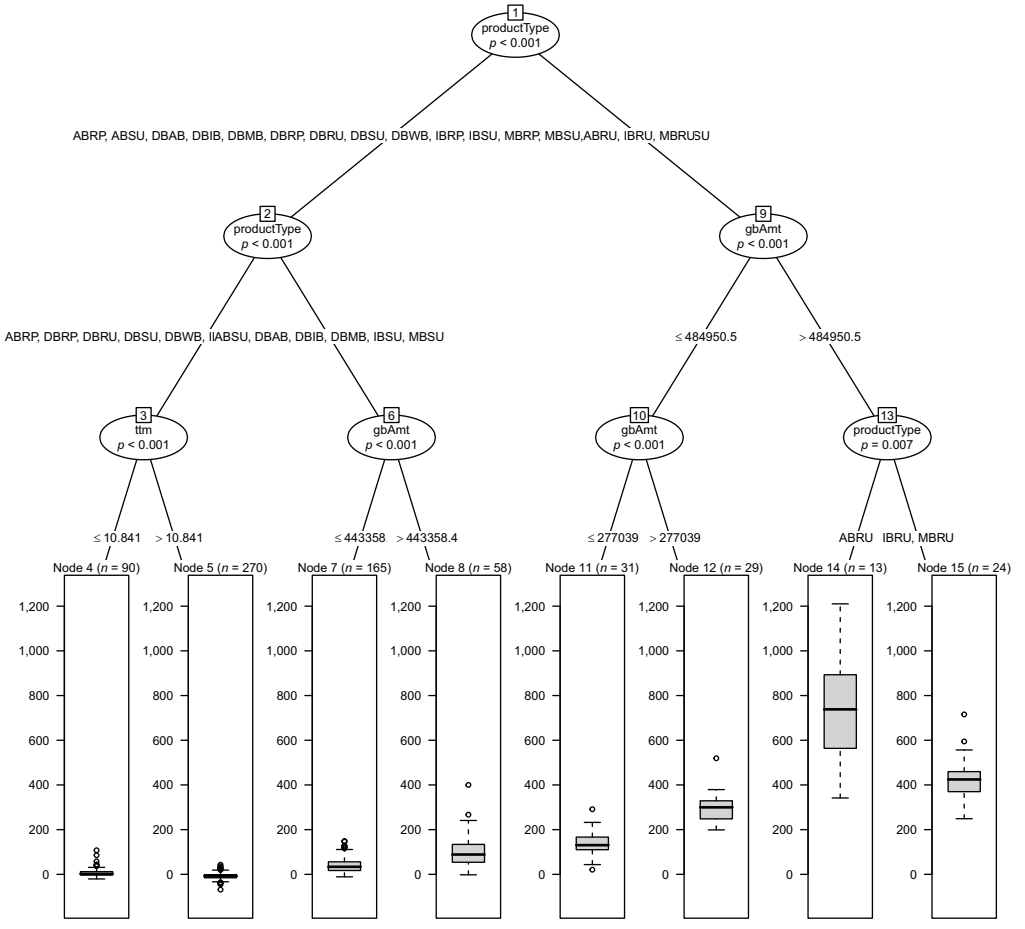

Figure 4. A conditional inference trees.

pick the desired predicted value. For example, we can use other quantiles to assign the predicted value at the terminal node.

The conditional random forest aggregates several hundreds of the conditional inference trees with five minimum observations at the terminal nodes. These trees also select the full set of explanatory variables at each split, which is consistent with the previous situation. In other words, the conditional inference random forests become conditional inference bagged trees.

In summary, we listed the final hyperparameters settings tuned by grid search and further constructed the final models based on these hyperparameters. We will discuss these final models further in the following sections.

4.3. Model performance and computational expense

We apply several validation measures to compare the performance of different tree-based models. In addition, we compare tree-based models with ordinary kriging and GB2 regression from previous studies. See Gan (2013) and Gan & Valdez (2018). All validation measures of prediction accuracy used in this comparison are defined in Table 4. Table 5 provides the numerical details about the prediction accuracy of these methods on the 190,000 VA contracts. Tree-based models are traditionally tuned by minimising the *MSE*. This may have resulted in less desirable values of *ME* and *PE*. However, when evaluating model performance particularly in a business setting, it is

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

Table 4. Validation measures

Validation measure	Description	Interpretation
Gini index	$Gini = 1 - \frac{2}{N-1}\left(N - \frac{\sum_{i=1}^{N} i\tilde{y}_i}{\sum_{i=1}^{N} \tilde{y}_i}\right)$ where $\tilde{y}$ is the corresponding to y after ranking the corresponding predicted values $\widehat{y}$	Higher Gini is better
Coefficient of Determination	$R^2 = 1 - \frac{\sum_{i=1}^{N} (\widehat{y}_i - y_i)^2}{\sum_{i=1}^{N} \left(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i\right)^2}$ where $\widehat{y}$ is predicted values	Higher R^2 is better
Concordance Correlation Coefficient	$CCC = \frac{2\rho\sigma_{\widehat{y}_i}\sigma_{y_i}}{\sigma_{\widehat{y}_i}^2 + \sigma_{y_i}^2 + (\mu_{\widehat{y}_i} - \mu_{y_i})^2}$ where $\mu_{\widehat{y}_i}$ and μ_{y_i} are the means $\sigma_{\widehat{y}_i}^2$ and $\sigma_{y_i}^2$ are the variances ρ is the correlation coefficient	Higher CCC is better
Mean Error	$ME = \frac{1}{N}\sum_{i=1}^{N}(\widehat{y}_i - y_i)$	Lower $\lvert ME\rvert$ is better
Percentage Error	$PE = \frac{\sum_{i=1}^{N} \widehat{y}_i - \sum_{i=1}^{N} y_i}{\sum_{i=1}^{N} y_i}$	Lower $\lvert PE\rvert$ is better
Mean Squared Error	$MSE = \frac{1}{N}\sum_{i=1}^{N} (\widehat{y}_i - y_i)^2$	Lower *MSE* is better
Mean Absolute Error	$MAE = \frac{1}{N}\sum_{i=1}^{N} \lvert\widehat{y}_i - y_i\rvert$	Lower *MAE* is better

Table 5. Prediction accuracy of different models

Model	*Gini*	R^2	*CCC*	*ME*	*PE*	*MSE*	*MAE*
Regression tree (CART)	0.786	0.845	0.917	−1.678	−0.025	3278.578	31.421
Bagged trees	0.842	0.918	0.954	−2.213	−0.033	1720.725	20.334
Gradient boosting	0.836	0.942	0.969	−1.311	−0.019	1214.899	19.341
Conditional inference trees	0.824	0.869	0.930	−0.905	−0.013	2754.853	26.536
Conditional random forests	0.836	0.892	0.940	−1.596	−0.024	2273.385	23.219
Ordinary kriging	0.815	0.857	0.912	0.812	0.012	3006.192	27.429
GB2	0.827	0.879	0.930	−0.106	−0.002	2554.246	27.772

best to examine and compare several validation measures. Although the values of *ME* and *PE* are negative for all tree-based models, the absolute values of *ME* and *PE* are small and comparable to those of other models. In addition, gradient boosting performs extremely well in terms of all the other validation measures.

In order to visually compare model performance, we introduce the heatmap. Figure 5 provides a heatmap comparing the performance of the models according to the validation measures. The purpose of this heatmap is for ease of comparison so that it has been organised by rescaling all the measures so that for each measure, a value of 100 is the best and a value of 0 is the worst. For *Gini*, R^2, and the *CCC* index, we know that the higher the value the better, so that for these measures, we find the highest value in each column and scale it to 100. The smallest value is then rescaled to 0. Other values between two boundaries are rescaled according to the distance. For all the other measures, since the smaller its absolute value the better, we multiplicatively invert (take the reciprocal of) the original absolute value and then apply the same idea of rescaling. We also colour coded the figure in the sense that dark red represents the worst and dark blue represents the

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

Table 6. Computational expenses

Model	Computation time (seconds)
Regression tree (CART)	0.13
Bagged trees	2.70
Gradient boosting	4.69
Conditional inference trees	0.25
Conditional random forests	1214.72
Ordinary kriging	277.49
GB2	23.44

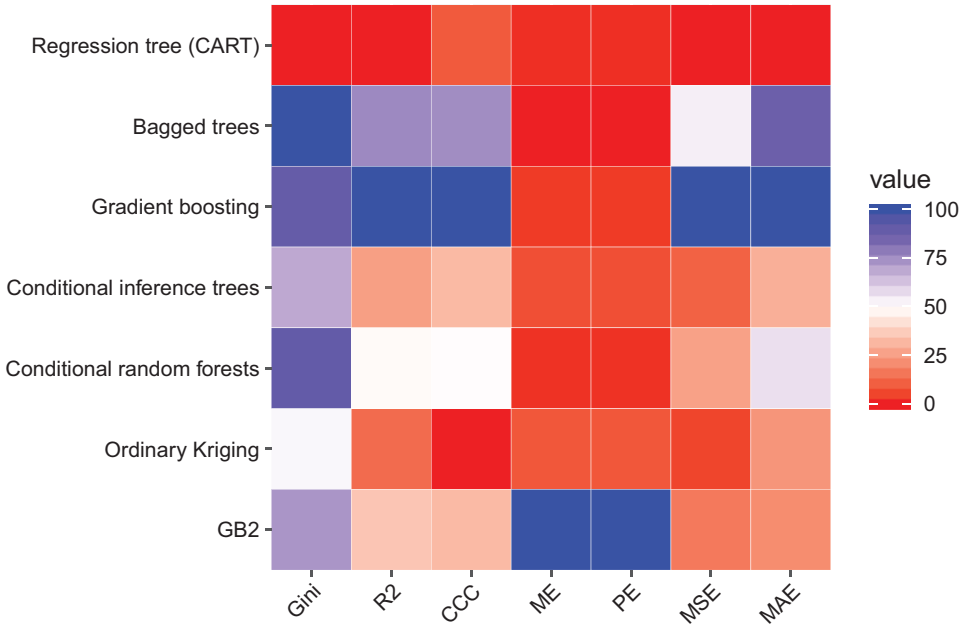

Figure 5. A heatmap of model performance according to the various validation measures.

best. The performance of any model in between is relatively measured according to their degree of closeness to each of these colours. A similar heatmap for such model comparison has appeared in Quan & Valdez (2018).

We can find that gradient boosting has the overall best performance. It is worth mentioning that conditional inference trees has better prediction performance under the mean error and percentage error, in comparison to other tree-based models. At the portfolio level, the conditional inference trees produces good results. Under the two validation measures *ME* and *PE*, the GB2 method outperforms other methods.

Table 6 provides details about the computational expenses, which include training time and prediction time. We find that the tree-based models based on the CART algorithm are more computationally efficient than the conditional inference framework. The extra computational cost associated with conditional inference trees can likely be deduced from the additional step of performing hypothesis test at each split. Moreover, tree-based models outperform other previous methods.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

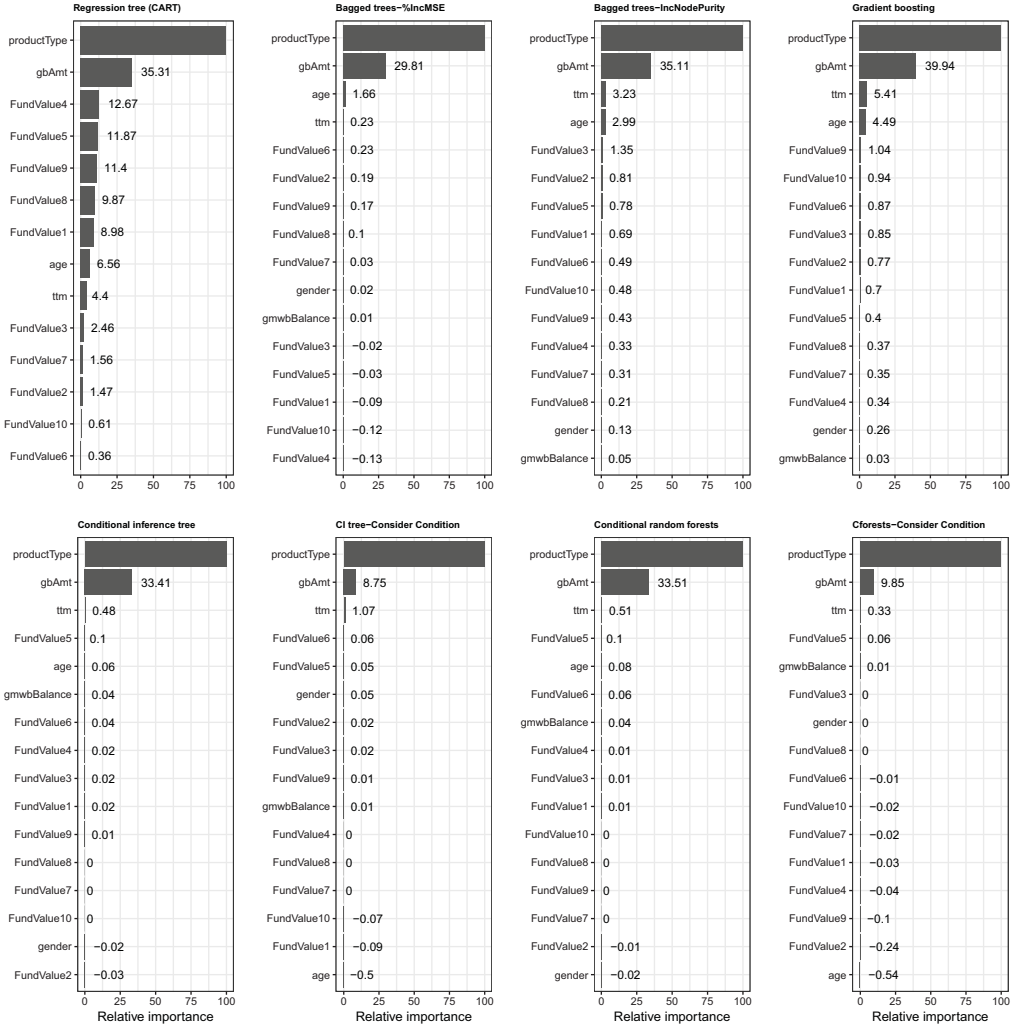

Figure 6. Variable importance for tree-based models (all the explanatory variables).

4.4. Variable importance

Figure 6 shows the variable importance for all five tree-based models. The first three models are based on the regression trees that are generated by the traditional CART algorithm. The last two are based on the unbiased conditional inference trees. For random forests (bagged trees), there are two types of variable importance: one based on *%IncMSE* and another based on *IncNodePurity*. *%IncMSE* based on the difference in prediction error after a random permutation values of the explanatory variables on the test set. *IncNodePurity* can be calculated based on the cumulation of improvement of the loss at each splitting which inherited from single trees. There are also two ways to calculate variable importance for conditional inference tree and conditional random forests. If conditions are considered, the importance of each explanatory variable is computed by permuting within a grid defined by the explanatory variables that are associated to the response variable. The resulting variable importance score is conditional in the same sense as beta coefficients in regression models, but it represents the effect of a variable in both main effects and interactions. We can see that all the tree-based models are able to select the top two important

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

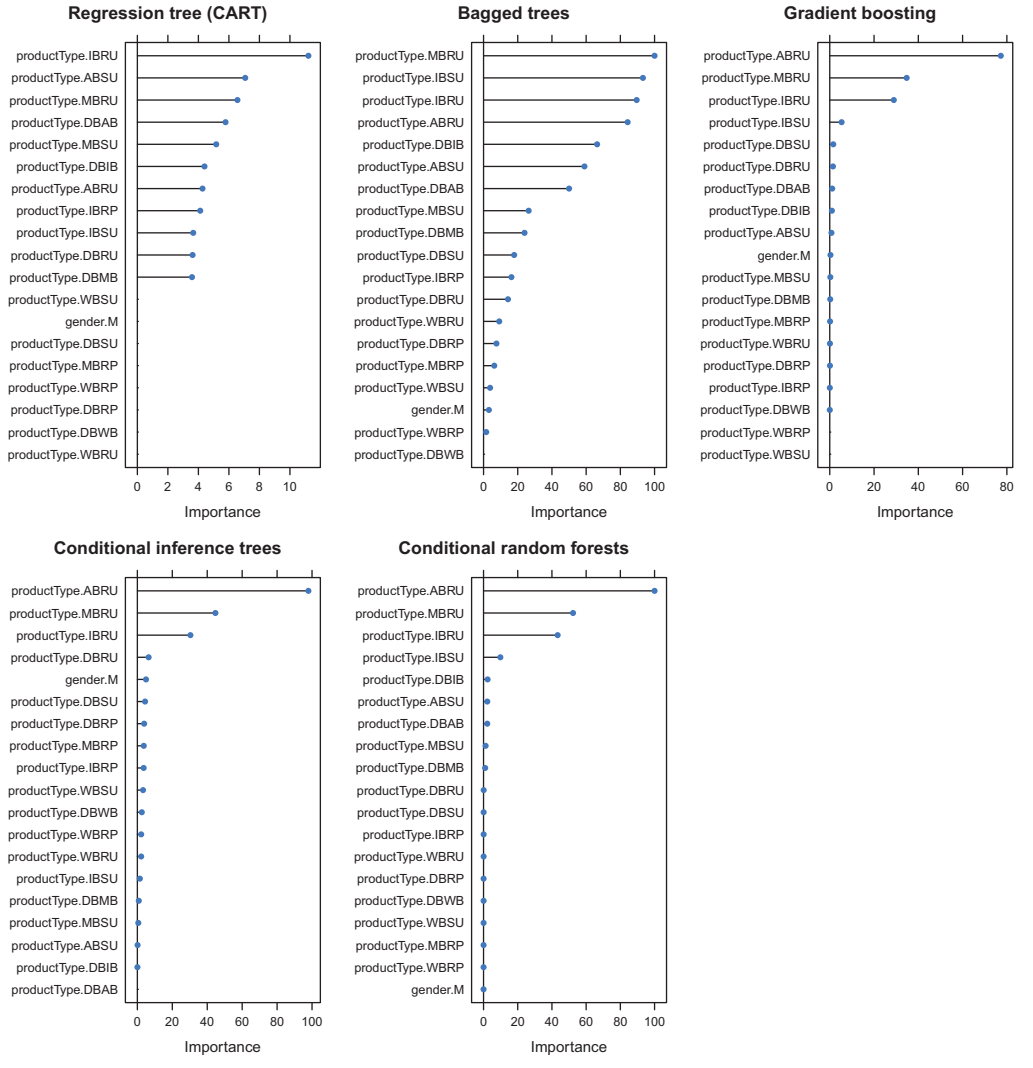

Figure 7. Variable importance for tree-based models (categorical explanatory variables).

variables, product type and guaranteed benefit amount, while the order of the subsequent variable importance is quite different between models. Interestingly, in contrast to the other methods, the CART did not choose the age of the policyholder and the time to maturity. It is widely recognised that the CART algorithm has the bias in variable selection and this appears in our dataset since we have the type of product with many levels. Controversially, conditional inference tree-based models have a similar selection of variable importance when compared to the bagged trees and gradient boosted regression trees, even though their structures of selecting the split variable are different.

As in the previous discussion, we note that the product type, which has several categories, as deemed most important variable in all tree-based models. Figure 7 shows the dummified categorical variable importance for all five tree-based models. The first three are based on the regression trees that are generated by the traditional CART algorithm. The last two are based on the unbiased conditional inference tree. We observe the similar result that there exists a bias in variable selection in the traditional framework when compared to the conditional inference framework. For example, ABRU, MBRU, and IBRU are the most important product type. Gradient boosting is able to select these same variables.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

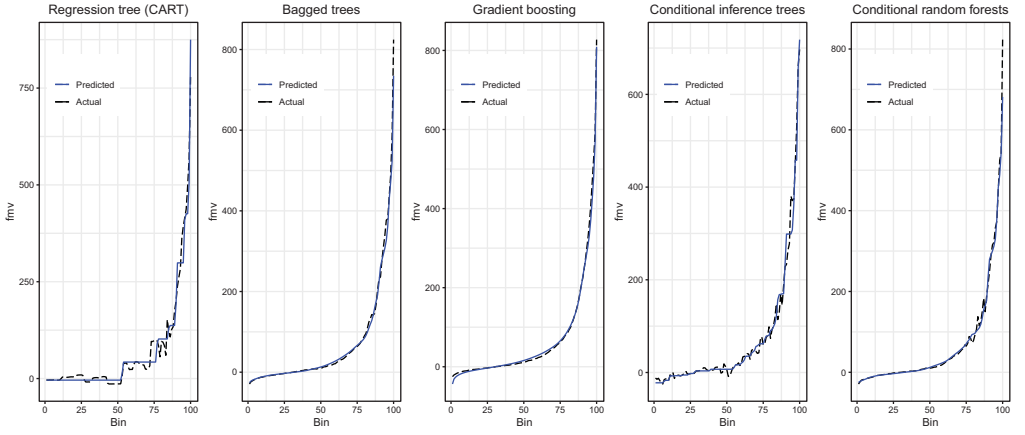

Figure 8. Lift curve plots.

4.5. Model performance visualisation

Figure 8 shows the lift curves, which are plotted using ordered predictions of the models and benchmark fair market values calculated by Monte Carlo simulation according to ranks. The lift curve is drawn by bining the test sample, 190,000 VA contracts, into 100 segments and taking the average of the ordered predictions and benchmark fair market values. All the average points are then connected. If the curve for the benchmark fair market values closely follows that of the prediction curve, we could say that the model prediction is sound. In examining the lift curve plots for the various models, we observe that bagged trees, gradient boosting, and conditional random forests appear to have the most reliable prediction.

To further visualise a comparison of the performance of the various tree-based models, Figure 9 shows a scatter plot between the fair market values calculated by Monte Carlo simulation and those predicted by metamodels. Beside the scatter plot, we also provide the density plot for both the predicted values and those calculated by Monte Carlo simulation. Not surprisingly, we see that single tree-based models only provide few constant predictions. On the other hand, the ensemble tree-based models provide more smoothed predictions. Overall, gradient boosting provides the best prediction performance.

5. Concluding Remarks

This paper explores tree-based models and their extensions in developing metamodels for predicting fair market values of the guarantees embedded in VA contracts. Tree-based models are increasingly widely used in various disciplines such as psychology, ecology, biology, economics, and insurance. As demonstrated in this paper, besides computational efficiency and predictive accuracy, they have many advantages as an alternative predictive tool. First, unlike ordinary kriging and the GB2 framework, tree-based models are considered as non-parametric models that do not require distribution assumptions. Second, tree-based models can perform variable selection by assessing the relative importance. Such variable selection is usually essential in actuarial science for purposes such as risk classification and collection of risk variables. This paper further showed how to refine variable selection of a categorical variable with several categories. Third, tree-based models, especially with single smaller sized trees, are straightforward to interpret by a visualisation of the tree structure. This visualisation was illustrated both in the case of regression tree and conditional inference tree. Finally, when compared to other metamodels for prediction purposes, tree-based models require less data preparation as they preserve the original scale to be more interpretable.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

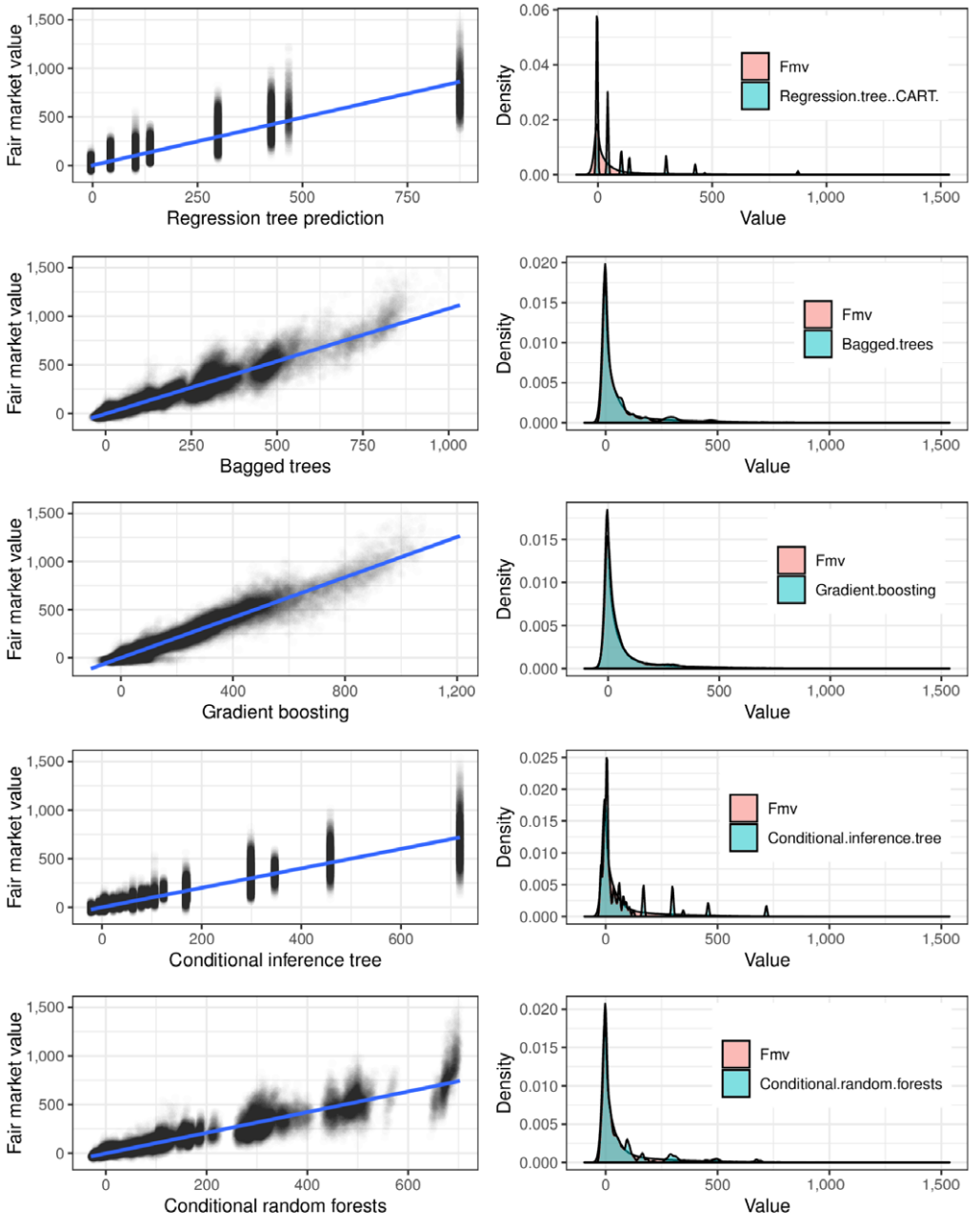

Figure 9. Prediction and actual fair market values.

To empirically demonstrate the use of tree-based models as a predictive tool, this paper uses the synthetic dataset with 190,000 VA contracts as described in Gan & Valdez (2017b). The objective here is to examine the performance of tree-based models in the valuation of fair market values of the product guarantees for large portfolios of VA contracts. In doing so, we compare the predictive accuracy of five tree-based models: regression tree (CART), bagged trees, gradient boosting, conditional inference trees, conditional random forests, as well as two parametric metamodels: ordinary kriging and GB2 models. For model comparison, we use a training dataset based on

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

a representative small sample selected by a clustering algorithm and for computing the model performance, we use the entire dataset as the test dataset. Broadly speaking, tree-based models outperform the parametric metamodels considered in this paper and are generally very efficient in producing more accurate predictions. While in several respects, the gradient boosting ensemble method is considered the most superior amongst all models considered, this paper further demonstrated the many good qualities of conditional inference trees. Such tree-based models can be used for other types of insurance and actuarial applications, especially for datasets that contain categorical explanatory variables with several levels.

Acknowledgements. We thank the Society of Actuaries for the funding support provided in the completion of this research project through our Centers of Actuarial Excellence (CAE) grant on data mining.

References

Allen, D.M. (1974). The relationship between variable selection and data agumentation and a method for prediction. *Technometrics*, **16**(1), 125–127.

Arlot, S. & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, **4**, 40–79.

Bergstra, J. & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, **13**, 281–305.

Bergstra, J.S., Bardenet, R., Bengio, Y. & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems* (pp. 2546–2554).

Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**(2), 123–140.

Breiman, L. (2001). Random forests. *Machine Learning*, **45**(1), 5–32.

Breiman, L., Friedman, J.H., Olshen, R.A. & Stone, C.J. (1984). *Classification and Regression Trees*. Taylor & Francis Group, LLC, Boca Raton, FL.

Dang, O., Feng, M. & Hardy, M.R. (2019). Efficient nested simulation for conditional tail expectation of variable annuities. *North American Actuarial Journal*, **24**(2), 187–210.

Devroye, L. & Wagner, T. (1979). Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory*, **25**(5), 601–604.

Feng, B.M., Tan, Z. & Zheng, J. (2020). Efficient simulation designs for valuation of large variable annuity portfolios. *North American Actuarial Journal*, **24**(2), 275–289.

Freund, Y. & Schapire, R.E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**(1), 119–139.

Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, **29**(5), 1189–1232.

Gan, G. (2013). Application of data clustering and machine learning in variable annuity valuation. *Insurance: Mathematics and Economics*, **53**(3), 795–801.

Gan, G. (2018). Valuation of large variable annuity portfolios using linear models with interactions. *Risks*, **6**(3), 71.

Gan, G. & Lin, X.S. (2015). Valuation of large variable annuity portfolios under nested simulation: a functional data approach. *Insurance: Mathematics and Economics*, **62**, 138–150.

Gan, G. & Lin, X.S. (2017). Efficient greek calculation of variable annuity portfolios for dynamic hedging: a two-level metamodeling approach. *North American Actuarial Journal*, **21**(2), 161–177.

Gan, G. & Valdez, E.A. (2017a). Modeling partial greeks of variable annuities with dependence. *Insurance: Mathematics and Economics*, **76**, 118–134.

Gan, G. & Valdez, E.A. (2017b). Valuation of large variable annuity portfolios: Monte Carlo simulation and synthetic datasets. *Dependence Modeling*, **5**, 354–374.

Gan, G. & Valdez, E.A. (2018). Regression modeling for the valuation of large variable annuity portfolios. *North American Actuarial Journal*, **22**(1), 40–54.

Geisser, S. (1974). A predictive approach to the random effect model. *Biometrika*, **61**(1), 101–107.

Geisser, S. (1975). The predictive sample reuse method with applications. *Journal of the American statistical Association*, **70**(350), 320–328.

Guelman, L., Guillén, M. & Pérez-Marn, A.M. (2014). A survey of personalized treatment models for pricing strategies in insurance. *Insurance: Mathematics and Economics*, **58**, 68–76.

Gweon, H., Li, S. & Mamon, R. (2020). An effective bias-corrected bagging method for the valuation of large variable annuity portfolios. *ASTIN Bulletin*, **50**(3), 853–871.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

Hardy, **M.** (2003). *Investment Guarantees: Modeling and Risk Management for Equity-Linked Life Insurance*, vol. 215. John Wiley & Sons, Hoboken, NJ.

Hejazi, **S.A. & Jackson**, **K.R.** (2016). A neural network approach to efficient valuation of large portfolios of variable annuities. *Insurance: Mathematics and Economics*, **70**, 169–181.

Hejazi, **S.A.**, **Jackson**, **K.R. & Gan**, **G.** (2017). A spatial interpolation framework for efficient valuation of large portfolios of variable annuities. *Quantitative Finance and Economics*, **1**(2), 125–144.

Hothorn, **T.**, **Hornik**, **K. & Zeileis**, **A.** (2006). Unbiased recursive partitioning: a conditional inference framework. *Journal of Computational and Graphical Statistics*, **15**(3), 651–674.

Ishwaran, **H.** (2007). Variable importance in binary regression trees and forests. *Electronic Journal of Statistics*, **1**, 519–537.

Kohavi, **R.** (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conferences on Artificial Intelligence (IJCAI)* (pp. 1137–1145), vol. 14. Montreal, Canada.

Larsen, **J.**, **Hansen**, **L.K.**, **Svarer**, **C. & Ohlsson**, **M.** (1996). Design and regularization of neural networks: the optimal use of a validation set. In *Neural Networks for Signal Processing VI. Proceedings of the 1996 IEEE Signal Processing Society Workshop* (pp. 62–71).

Lee, **S.C. & Lin**, **S.** (2018). Delta boosting machine with application to general insurance. *North American Actuarial Journal*, **22**(3), 405–425.

Leung, **D.H.-Y.** (2005). Cross-validation in nonparametric regression with outliers. *The Annals of Statistics*, **33**(5), 2291–2310.

Lin, **X.S. & Yang**, **S.** (2020). Fast and efficient nested simulation for large variable annuity portfolios: a surrogate modeling approach. *Insurance: Mathematics and Economics*, **91**, 85–103.

Liu, **K. & Tan**, **K.S.** (2020). Real-time valuation of large variable annuity portfolios: a green mesh approach. *North American Actuarial Journal*, In Press.

Martinez-Cantin, **R.** (2014). Bayesopt: a Bayesian optimization library for nonlinear optimization, experimental design and bandits. *The Journal of Machine Learning Research*, **15**(1), 3735–3739.

Moors, **E.** (1920). On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, **26**, 394–395.

Morgan, **J.N. & Sonquist**, **J.A.** (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, **58**(302), 415–434.

Mosteller, **F. & Tukey**, **J.W.** (1968). Data analysis, including statistics. *Handbook of Social Psychology*, **2**, 80–203.

Nister, **D. & Stewenius**, **H.** (2006). Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (pp. 2161–2168), vol. 2.

O'Brien, **S.M.** (2004). Cutpoint selection for categorizing a continuous predictor. *Biometrics*, **60**(2), 504–509.

Opsomer, **J.**, **Wang**, **Y. & Yang**, **Y.** (2001). Nonparametric regression with correlated errors. *Statistical Science*, **16**(2), 134–153.

Penrose, **R.** (1955). A generalized inverse for matrices. *In Mathematical Proceedings of the Cambridge Philosophical Society* (pp. 406–413), vol. 51. Cambridge University Press.

Quan, **Z. & Valdez**, **E.A.** (2018). Predictive analytics of insurance claims using multivariate decision trees. *Dependence Modeling*, **6**(1), 377–407.

Shao, **J.** (1993). Linear model selection by cross-validation. *Journal of the American Statistical Association*, **88**(422), 486–494.

Snoek, **J.**, **Larochelle**, **H. & Adams**, **R.P.** (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems* (pp. 2951–2959).

Stone, **M.** (1974). Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society. Series B (Methodological)*, **36**(2), 111–147.

Strasser, **H. & Weber**, **C.** (1999). On the asymptotic theory of permutation statistics, Vienna University of Economics and Business Administration, Augasse 2–6, A–1090 Vienna, Austria.

Strobl, **C.**, **Boulesteix**, **A.-L.**, **Zeileis**, **A. & Hothorn**, **T.** (2007). Bias in random forest variable importance measures: illustrations, sources and a solution. *BMC Bioinformatics*, **8**(1), 25.

Strobl, **C.**, **Malley**, **J. & Tutz**, **G.** (2009). An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological Methods*, **14**(4), 323.

Westfall, **P.H. & Young**, **S.** (1993). *Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment*. John Wiley & Sons, Inc., New York, NY.

Xu, **W.**, **Chen**, **Y.**, **Coleman**, **C. & Coleman**, **T.F.** (2018). Moment matching machine learning methods for risk management of large variable annuity portfolios. *Journal of Economic Dynamics and Control*, **87**, 1–20.

A. Other Hyperparameter Optimisation Results

If the training dataset is significantly big, it is not ideal to perform the grid search. Instead, the random search or the Bayesian optimisation are preferred in this situation. In this appendix, we provide the comparison of these two hyperparameter tuning methods.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

Table A.1. Random search and Bayesian optimisation

Model	Training MSE	Test MSE	Runtime
		Random search	
Regression tree (CART)	3846.0	3717.8	4.6 seconds
Bagged trees	1875.5	1829.4	120.7 seconds
Gradient boosting	2552.6	3061.3	251.1 seconds
Conditional inference trees	4090.0	3778.5	8.4 seconds
Conditional random forests	2792.1	2421.1	921.3 seconds
		Bayesian optimisation	
Regression tree (CART)	3609.7	3726.2	1.7 minutes
Bagged trees	1992.3	1860.2	38.7 minutes
Gradient boosting	1487.3	1918.2	38.8 minutes
Conditional inference trees	3871.9	3779.2	6.4 minutes
Conditional random forests	1992.3	1912.8	29.4 minutes

For the random search, we use `caret` package to perform 50 random searches on hyperparameter space for all 5 tree-based models. We select the model with minimum ten-fold cross-validation root mean squared error (RMSE) and then find the prediction accuracy for each tree-based model on the whole dataset. Since the random search can be parallel, we use six nodes with this computer configuration (Operating system: Microsoft Windows 10 Education, 64bit; Processor: Intel Core i7-7700K at 4.2 GHz, turbo up to 4.5 GHz, 8 MB cache, four cores; Memory: 32 GB).

For the Bayesian optimisation, the GP regression model applies a radial basis kernel to the covariance function of the multivariate normal prior. The GP regression model needs some initial set of hyperparameters and its model performance and makes predictions over the bounding box of the hyperparameters space. In our setting, there are ten initial sets of hyperparameters. According to this, we can obtain the estimated mean and variance for prediction accuracy (RMSE). Finally, we perform 50 searches applying the Bayesian optimisation using `caret` package and `rBayesianOptimisation` package. Again, to be fair in comparison, we select the model with minimum ten-fold cross-validation RMSE and then find the prediction accuracy for each tree-based model on the whole dataset. Bayesian optimisation is performed without parallelism.

Table A.1. provides details about the training (680 contracts) mean square error (TrainMSE), the testing (whole contracts) mean square error (TestMSE), and computational expenses (Time) which include training time (hyperparameters tuning) and prediction time. We can see that the Bayesian optimisation is slower than the random search. Bayesian optimisation can provide better hyperparameter setting in the case of complex situations like gradient boosting with four hyperparameters. We should point out that the results shown in Table A.1 are based on simple illustration and comparison of using the random search and Bayesian optimisation to optimise various tree-based models' hyperparameters. The prediction results should not be compared directly with those based on the grid search that are shown in Table 5 since the dimension of the grid is not comparable.

In practice, for the large-scale dataset, we can apply a few random searches and send results to the initial setting for the Bayesian optimisation. After performing the Bayesian optimisation, we can have a better understanding of the dataset and perform the grid search on the hyperparameter space which is narrowed down by the previous experiment.

https://doi.org/10.1017/S1748499521000075 Published online by Cambridge University Press

Table B.1. Prediction accuracy of deltas, based on R^2

	Delta1	Delta2	Delta3	Delta4	Delta5
Regression tree (CART)	0.505	0.431	0.273	0.441	0.563
Bagged trees	0.663	**0.690**	0.635	0.733	0.729
Gradient boosting	**0.705**	0.685	**0.668**	**0.753**	**0.741**
Conditional inference trees	0.304	0.450	0.276	0.575	0.534
Conditional random forests	0.527	0.626	0.507	0.607	0.631
Ordinary kriging	0.479	0.427	0.446	0.391	0.411
GB2	0.479	0.330	0.376	0.359	0.406

Table B.2. Prediction accuracy of deltas, based on *ME*

	Delta1	Delta2	Delta3	Delta4	Delta5
Regression tree (CART)	1.243	**0.059**	0.739	0.403	0.917
Bagged trees	0.712	0.134	0.570	0.992	1.054
Gradient boosting	0.551	0.660	0.791	0.851	1.006
Conditional inference trees	−0.348	−0.401	0.474	0.512	**0.107**
Conditional random forests	**0.268**	−0.248	0.193	0.160	0.370
Ordinary kriging	−0.573	0.143	**−0.069**	**−0.132**	0.448
GB2	−1.393	−1.608	−1.804	−1.770	−0.848

Table B.3. Prediction accuracy of deltas, based on *PE*

	Delta1	Delta2	Delta3	Delta4	Delta5
Regression tree (CART)	−0.056	**−0.004**	−0.049	−0.035	−0.074
Bagged trees	−0.032	−0.010	−0.038	−0.086	−0.086
Gradient boosting	−0.025	−0.048	−0.053	−0.073	−0.082
Conditional inference trees	0.016	0.029	−0.032	−0.044	**−0.009**
Conditional random forests	**−0.012**	0.018	−0.013	−0.014	−0.030
Ordinary kriging	0.026	−0.011	**0.005**	**0.011**	−0.036
GB2	0.063	0.117	0.072	0.153	0.069

B. Modelling Partial Greeks of Variable Annuities

The synthetic dataset created in Gan & Valdez (2017b) also contains the partial dollar deltas on five market indices. We apply the same techniques described in Section 4 to model these five partial dollar deltas, which were studied by Gan & Lin (2017) and Gan & Valdez (2017a).

Tables B.1, B.2, and B.3 show the prediction accuracy of the tree-based models and traditional approaches on the 190,000 VA contracts on five dollar deltas. The best performance numbers are shown in boldface. From these tables, we observe that gradient boosting has consistent good performance in terms of R^2. However, bagged trees perform better than gradient boosting for *Delta*2. From Tables B.3 and B.2, it can be more challenging to judge the overall best model. Nevertheless, tree-based models have good prediction performance on the Greeks of VA portfolios with respect to *ME* and *PE* measures.

Cite this article: Quan Z, Gan G and Valdez E (2022). Tree-based models for variable annuity valuation: parameter tuning and empirical analysis. *Annals of Actuarial Science*, **16**, 95–118. https://doi.org/10.1017/S1748499521000075