Comparison of Machine Learning Methods for Insurance Premium Prediction

University of Waterloo
Computational Mathematics Research Paper

by

Meiyu Zhou

Supervisor: Prof. Ken Seng Tan

Abstract

The goal of my research is to compare different machine learning methods to predict pure premium for auto-insurance claim data. Based on the result from article *Insurance Premium Prediction via Gradient Tree-Boosted Tweedie Compound Poisson Model*, this paper compares generalized additive model, TDboost, Xgboost, Gradient-boost, and random forest for extension. Since the claim data is assumed to follow Tweedie distribution, models use profile likelihood approach to estimate optimized index and dispersion parameters. The Gini Index, Absolute Mean Error, Bootstrap and Computational time are considered to assess the performance of each model.

1. Introduction

In recent years, machine learning has developed rapidly to solve challenges in insurance industry, which include underwriting and loss prevention, product pricing, claims handling, fraud detection, sales and customer experience[20]. Insurance premium is typically determined in such a way that it costs the expected payout of the policy, as well as taking into consideration the administration and profit of the insurance company to ensure sustainability of the business. One of the most commonly target variables used in machine learning regression in insurance industry is claim size, which is one of the crucial factors to set premium. In the age of big data, machine learning could be an option for company to gain the information and insight of business by learning from data. In other words, the expected learning result comes with highly correlation between the claim size and information about policyholder in regression model.

Generalized linear models (GLM) [22] are commonly used for predicting claim size in the past. Data usually are sampled from exponential, normal, gamma, Poisson, and binomial distribution to fit model. However, the distribution of claim size is highly right-skewed, which zero claim accounts for large proportion. Jørgensen and de Souza (1994) [14] assumed Poisson arrival of claims and gamma distributed costs for individual claims, which imply that the claim size follows a Tweedie compound Poisson distribution. The Tweedie compound Poisson distribution can model data consists of zeros and positive number. The structure of the logarithmic mean is restricted to a linear form, which can be too rigid for real application [28]. Yang, Qian and Zou (2018) [28] proposed a gradient tree-boosted algorithm and apply it to Tweedie compound Poisson models(TDboost) for pure premium. The paper also compares GLM, generalized additive model(GAM), and TDboost model. From our implementation, both GAM and TDboost appear to produce more accurate result for predicting pure premium. Based on the result from Yang, Qian, and Zou [28], this article explores gradient boosting (GB), XGBoost and random forest and discusses their application to auto insurance claims data (Yip and Yau 2005 [29]).

Boosting method is one of the ensemble learning and consists of base learners. The basic idea for boosting method is to reduce prediction error by training weak learner and correcting its predecessor. The combination of each weak learner can output more accurate prediction. Gradient boosting [7] is one of the most popular machine learning methods for regression and classification. Each base learner is used to reduce residual errors that output from previous predictor. Gradient boosting applied widely across various fields, such as face alignment [26], epidemiology [8], and web search ranking [23]. Extreme gradient boosting (XGBoost) [3] is also popular candidate in Kaggle competition. It is a implementation of gradient boosting for fast, memory efficient and high accuracy [24]. Another ensemble

learning, random forest is one of bagging algorithms, which uses tree-structured classifier to predict response. The weakness of random forest is suffering the extremely imbalanced data set. In order to minimize prediction error, the model focuses on majority class [2]. If assuming response variable claim size is sampled from Poisson distribution, prediction can improve from random forest. Moreover, random forest also can detect nonlinear effect and interactions among explanatory variables. The variable importance in random forest is also crucial for further analysis in insurance industry.

The remainder of this article has been organized as follows. In Section 2, we review the compound Poisson distribution and Tweedie model. In Section 3, we discuss the concept of boosting method and random forest. Model results and assessment are presented in Section 4. Finally, the concluding remark and further research are provided in Section 5.

2. Tweedie's Compound Poisson Model

The insurance claim data in real application usually consists of positive losses and zeros losses. The distribution of claim data are highly right skewed. For such discrete data, Bent Jørgensen and Marta C. Paes De Souza [14] applied Tweedie's compound Poisson model to fit claim data.

Let Y denote the total claim size. In insurance claim size modeling, Y is typically modelled as a random sum as follows:

$$Y = \begin{cases} 0 & if N = 0 \\ Y_1 + Y_2 + ... + Y_N & if N = 1, 2, ..., \end{cases} \tag{1}$$

where N is a discrete integer value random variable that captures the total number of claim occurrence over a given time period and Y_i, i=1, 2, ..., denote the size (or severity) of the i-th claim and are assumed to be iid. In our examples, we assume N follows Poisson distribution (i.e. N $\sim$ Pois(λ)) and each i-th claim Y_i follows the Gamma distribution (i.e. $Y_i \sim$ Gamma(α,β)). We further assume N and Y_i are independent. Also, Y is is the response variable in our application.

Under the above assumptions, it is easily to establish the following results: The expected value of Y is $\mu = \frac{\lambda\alpha}{\beta}$, and the variance is $\frac{\lambda\alpha(\alpha+1)}{\beta^2} = \phi\mu^\rho$, where $\mu > 0$ and dispersion $\phi > 0$. The parameterization satisfy

$$\lambda = \frac{\mu^2 - \rho}{\phi(2-\rho)}, \quad \alpha = \frac{2-\rho}{\rho-1}, \quad \frac{1}{\beta} = \phi(\rho-1)\mu^{\rho-1}. \tag{2}$$

The distribution function of Y can be written as

$$f_Y(y|\alpha,\beta,\lambda) = \begin{cases} e^{-\lambda} & if y = 0 \\ e^{-\beta y} e^{-\lambda} \sum_{n=1}^{\infty} \frac{\beta^{n\alpha}}{\Gamma(n\alpha)} y^{n\alpha-1} \frac{\lambda^n}{n!} & if y > 0. \end{cases} \tag{3}$$

Tweedie distribution belongs to Exponential Dispersion Models(EDM)[13], which has the density as

$$f_Y(y|\theta,\phi) = a(z,\phi) exp\{\frac{z\theta - \kappa(\theta)}{\phi}\}, \tag{4}$$

where a$(\cdot)$ and $\kappa(\cdot)$ are known normalizing function and cumulant function, respectively. To transform density function (3) in the form of Tweedie distribution (4), the mean-variance relation gives

$$\theta = \begin{cases} \frac{\mu^{1-\rho}}{1-\rho} & if \rho \neq 1 \\ log(\mu) & if \rho = 1 \end{cases}, \quad \kappa(\theta) = \begin{cases} \frac{\mu^{2-\rho}}{2-\rho} & if \rho \neq 2 \\ log(\mu) & if \rho = 2. \end{cases} \tag{5}$$

Thus, the compound Poisson model for Y has Tweedie model form with $1 < \rho < 2$ and $\mu > 0$. If we take moment generate function, we get Poisson distribution when $\rho = 1$ and Gamma distribution when $\rho = 2$.

The log-likelihood function of Tweedie model can be written as

$$log f_Y(y|\mu,\phi,\rho) = \frac{1}{\phi}(y\frac{\mu^{1-\rho}}{1-\rho} - \frac{\mu^{2-\rho}}{2-\rho}) + log a(y,\phi,\rho), \tag{6}$$

where a$(\cdot)$ can be written as

$$a(y,\phi,\rho) = \begin{cases} \frac{1}{y} \sum_{n=1}^{\infty} \frac{y^{n\alpha}}{(\rho-1)^{n\alpha} \phi^{n(1+\alpha)} (2-\rho)^n n! \Gamma(n\alpha)} & if y > 0 \\ 1 & if y = 0. \end{cases} \tag{7}$$

3. Methodology

Given an auto-insurance data and label each record as i, let N_i denote the number of claims occurred and Y_i denote the size of each claim in N_i. Let w_i denote the policy

duration. Thus, the total claim amount is the sum of all claim size that occurred. In this article, the prediction based on ratio between total claim amount and duration $\frac{Y_i}{w_i}$, which known as pure premium. The pure premium is part of an insurance premium which can reflect the basic costs of loss, not including over-head or profit [4].

In our application with given pure premium y_i, vector of predictors $\mathbf{x_i}$ and duration w_i, $\{(y_i, \mathbf{x_i}, \mathbf{w_i})\}_{\mathbf{i=1}}^{\mathbf{n}}$, the log-likelihood function can be written as

$$\begin{aligned} l(F(\cdot), \phi, \rho | \{(y_i, \mathbf{x_i}, w_i)\}_{i=1}^{n}) &= \sum_{i=1}^{n} log f_Y(y_i | \mu_i, \frac{\phi}{w_i}, \rho) \\ &= \sum_{i=1}^{n} \frac{w_i}{\phi}(y_i \frac{\mu_i^{1-\rho}}{1-\rho} - \frac{\mu_i^{2-\rho}}{2-\rho}) + loga(y_i, \frac{\phi}{w_i}, \rho). \end{aligned} \tag{8}$$

3.1 Generalized Additive Model

Generalized additive model (GAM) is a method to add one or more nonlinear functions into generalized linear model by taking the function turning into blocks and using some mixture model to fit the smoother. The predicted values of GAM can arise from exponential family refer to continuous outcomes, discrete outcomes,and proportion outcomes, etc. and the outputs does not depend on prior model, which known as data-driven [27]. GAMs has the form as

$$g(\mu) = \alpha + \sum_j f_j(x_j), \tag{9}$$

where α is error term and $f_j(x_j)$ is smoothing function. Fitting this model without any structure would be a big challenge. Thus, we create a broad class of function to fit building blocks, which using local scoring algorithm to estimate smoothing function $f_j(x_j)$ nonparametrically, using a scatterplot smoother as a building block [10]. There are many types of scatterplot smoothers such as cubic splines, B-splines, and Locally weighted least square regression (loess) which uses the tricubic function to weight each observation in building block. The constrain for GAM requires functions f_j are unique and centered about 0, which is $E(f_j(x_j)) = 0$ for all j. If f_j follows form as addition of linear function, the generalized additive model is generalized linear model[27].

Generalized additive model is extension of generalized linear model. The replacement between additive and linear means the model no longer only considers linear predictor and

linear in outcomes, but instead regarding linear predictor as parameter. The most familiar additive model is polynomial model. However, the fitting result for polynomial regression is unstable and the regresses are very correlated, which leading to multicollinearity. The consequences of multicollinearity are unstable estimate and highly sensitive to outliers [11]. Another important concept of GAM is smoothing. The smoothness requires function to be continuous and differentiable so the derivative gives idea of slope and allows numerically approximation. As mentioned above, loess smoother is more generalized smoothing method, which uses local data to estimate response variable. It does not require the functional structure between predictors and response variable. The only restriction is function form must be smooth. However, the less restriction on function, the model is more rely on the choice of smoothing parameter and increase computational time.

Spline is another critical concept in smoothing, which is piecewise polynomial functions as shown below:

$$\begin{aligned} y_i &= \beta_0 + \beta_1 x_i + \sum_k^K u_k (x_i - \kappa_k)_+ + \epsilon_i \\ (x_i - \kappa_k)_+ &= \begin{cases} 0 & if x \leq \kappa_k \\ x - \kappa_k & if x > \kappa_k, \end{cases} \end{aligned} \tag{10}$$

where u_k defined as k-th order spline, which has continuous derivative of orders 1,...k-1, at its knot. To determine how smooth the spline is is also question for spline. The smoothness is determined by the number of knots and the basis function, which to minimize prediction error. Penalised splines allow additional parameter to measure how smooth to be. Even though we set number of knots, it does not constrain the smoothness. Consider minimizing, penalized spline using the following criteria:

$$SS^* = \sum_{i=1}^{n} [y_i - f(x_i)]^2 + h \int_{x_{min}}^{x_{max}} [f''(x)]^2 dx, \tag{11}$$

where the first term is the residual sum of square and the second term is the multiplication between smoothing parameter and roughness penalty. If h is large, the model gives more important role to roughness than least square, which implies greather smoothness.

In this article, we use MGCV in R to apply GAM on auto-insurance data fitting with

thin plate spline regression.

3.2 TDboost

TDboost was derived by Yi, Wei, and Hui [28], which integrates the boosted Tweedie model into gradient boost methods. Assume parameters ρ and ϕ are known, the goal is to estimate function F$(\cdot)$. Since the boosting method involves base learners, we define minimizer function $F^*(\cdot)$ as base learn over a class of $\mathbb{F}$, and can be written as,

$$\begin{aligned} F^*(\mathbf{x}) &= \arg\min_{F\in\mathbb{F}}\{-l(F(\cdot),\phi,\rho|\{(y_i,\mathbf{x_i},w_i)\}_{i=1}^n)\} \\ &= \arg\min_{F\in\mathbb{F}}\sum_{i=1}^n \Psi(y_i,F(x_i)|\rho) \\ where& \\ \Psi(y_i,F(x_i)|\phi) &= w_i\{-\frac{y_i exp[(1-\rho)F(\mathbf{x_i})]}{1-\rho}+\frac{exp[(2-\rho)F(\mathbf{x_i})]}{2-\rho}\}, \end{aligned} \tag{12}$$

The initial base learner is a constant function, which can minimize negative log-likelihood,

$$\begin{aligned} \hat{F}^{[0]} &= \arg\min_{\eta}\sum_{i=1}^n \Psi(y_i,\eta|rho) \\ &= log(\frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i}). \end{aligned} \tag{13}$$

Assume current stage is m, the estimated parameter $\xi^{[m]}$ for base leaner h($\mathbf{x_i}$, $\xi^{[m]}$) is,

$$\begin{aligned} \hat{\xi}^{[m]} &= \arg\min_{\xi^{[\mathbf{m}]}}\sum_{i=1}^n [u_i^{[m]}-h(\mathbf{x_i};\xi^{[\mathbf{m}]})]^2 \\ where& \\ u_i^{[m]} &= -\frac{\partial\Psi(y_i,F(x_i)|\rho)}{\partial F(x_i)}|_{F(x_i)=\hat{F}^{[m-1]}(x_i)} \\ &= w_i\{-y_i exp[(1-\rho)\hat{F}^{[m-1]}(x_i)]+exp[(2-\rho)\hat{F}^{[m-1]}(x_i)]\}, \end{aligned} \tag{14}$$

where u_i is negative gradient vector and uses L-terminal node regression tree as base learner,

$$h(x_i, \xi^{[m]}) = \sum_{l=1}^{L} u_l^{[m]} I(x \in R_l^{[m]}), \tag{15}$$

where $R_l^{[m]}$ and $u_l^{[m]}$, which are two parameters in base leaner, are calculated by "best-fit" algorithm with a least-square splitting criterion [12]. The expansion coefficient $\beta^{[m]}$ can be estimated by line search as,

$$\begin{aligned} \beta^{[m]} &= \arg\min_{\beta} \sum_{i=1}^{n} \Psi(y_i, \hat{F}^{m-1}(x_i) + \beta h(x_i; \hat{\xi}^{[m]})|\rho) \\ &= \arg\min_{\beta} \sum_{i=1}^{n} \Psi(y_i, \hat{F}^{m-1}(x_i) + \beta \sum_{l=1}^{L} \bar{u}_l^{[m]} I(x_i \in \hat{R}_l^{[m]})|\rho), \end{aligned} \tag{16}$$

where $\bar{u_i^{[m]}} = mean_{i:x_i \in \hat{R}_l^{[m]}}(u_i^{[m]})$ is a constant value within each region $\hat{R}_l^{[m]}$. Applying a separate line search method to each region, we can solve β by finding constant $\eta_l^{[m]}$ to improve performance of current estimation for each region $\hat{R}_l^{[m]}$, and optimal $\hat{\eta}_l^{[m]}$ is,

$$\hat{\eta_l^{[m]}} = log\{\frac{\sum_{i:x_i \in \hat{R}_l^{[m]}} w_i y_i exp[(1-\rho)\hat{F}^{[m-1]}(x_i)]}{\sum_{i:x_i \in \hat{R}_l^{[m]}} w_i exp[(2-\rho)\hat{F}^{[m-1]}(x_i)]}\}, l = 1, ...L. \tag{17}$$

Thus, the update estimated function $\hat{F}^{[m]}(x)$ for each region $\hat{R}_l^{[m]}$ is,

$$\hat{F}^{[m]}(x) = \hat{F}^{[m-1]}(x) + v\hat{\eta}_l^{[m]} I(x \in \hat{R}_l^{[m]}), l = 1, ...L, \tag{18}$$

where $0 < v \leq 1$ is learning rate.

Prior to estimating predictor function, we need to estimate index parameter ρ and dispersion ϕ for Tweedie distribution. Yi, Wei, and Hui [28] use profile likelihood to estimate these parameter. From section 2, the parameterization shows estimation μ depends on ρ. If ρ is given, $\mu^*(\rho)$ can be solved by (12) and $\phi^*(\rho)$ can use golden section search and parabolic interpolation in following optimization problem,

$$\phi^*(\rho) = \arg\max_{\phi}\{l(\mu^*(\rho), \phi, \rho\}. \tag{19}$$

To estimate ρ, the authors maximize the profile likelihood with equal spaced 50 values $\{\rho_1, ..., \rho_{50}\}$ on (0,1),

$$\rho^* = \arg\max_{\rho\in\{\rho_1,...,\rho_{50}\}} \{l(\mu^*(\rho), \phi^*(\rho), \rho)\}. \tag{20}$$

3.3 Gradient Boosting

Gradient boosting [7] consecutive, non-parametric machine learning methods to update more accurate estimation at each stage and minimize loss function. The TDboost introduced in previous subsection is an extension of Gradient boosting, which involves training base learner and ensembles all base learner to construct powerful model. The loss function in our application is (8). The goal is to estimate function $\hat{F}(\cdot)$ which fits predictor variables to response variable by minimizing the expected value of loss function $\Psi(y, F(x))$,

$$\begin{aligned} \hat{F}(x) &= \arg\min_{F(x)} \Psi(y, F(x)) \\ &= \arg\min_{F(x)} E_x[E_y(\Psi[y, F(x)])|x], \end{aligned} \tag{21}$$

where the expectation formula indicates minimizes the expected loss function over the response variable with known predictor variable x. Assume the current stage is m-1, then the function estimation at m-th iteration is defined as,

$$\begin{aligned} \hat{F}^{[m]} &= \hat{F}^{[m-1]} + \beta^{[m]} h(x; \xi^{[m]}) \\ (\beta^{[m]}, \xi^{[m]}) &= \arg\min_{\beta,\xi} \sum_{i=1}^{N} \Psi(y_i, \hat{F}^{[m-1]}) + \beta h(x_i, \xi). \end{aligned} \tag{22}$$

Gradient boosting is proposed to update base learner by using gradient descent like approach. Applying negative gradient of loss function to each observation as basis and check the rest to make sure no such repeating errors, the updated weak learner will fit the

basis at each iteration.Finally, the expansion coefficient β can be estimated by minimization problem [21],

$$
\begin{aligned}
g^{[m]} &= E_y[\frac{\partial \Psi(y, F(x))}{\partial F(x)}|x]_{F(x)=\hat{F}(x)^{[m-1]}} \\
\beta^{[m]} &= \arg\min_{\beta} \sum_{i=1}^{N} \Psi[y_i, \hat{F}^{[m-1]}(x_i) + \beta h(x_i, \xi^{[m]})].
\end{aligned}
\tag{23}
$$

The finalized estimation function can be written as,

$$
\hat{F}^{[m]}(x) = \hat{F}^{[m-1]}(x) + v\beta^{[m]} h(x; \xi^{[m]}), \tag{24}
$$

where v is learning rate, which is used to control step size for updating. The small value of learning rate becomes more shrinkage and causes longer time for computation. However, the appropriate learning rate can help with over-fitting and predict more accurate result [7].

3.4 Xgboost

XGBoost[3] is ensemble machine learning algorithm which stands for eXtreme Gradient Boosting. It can achieve more efficiency and less computational resource than gradient boosting method by exploiting sparsity-aware algorithm and weighted quantile sketch in algorithm and considering hardware perspective such as, cache access patterns, data compression and sharding [3].

As for the algorithm improvement, the loss function of XGBoost is,

$$
\begin{aligned}
L(\phi) &= \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \\
where \quad \Omega(f) &= \eta T + \frac{1}{2}\lambda \|w\|^2,
\end{aligned}
\tag{25}
$$

here T is the number of leaves in trees, w is the weight for each leaf and η is learning rate. The regularization term is used to smooth leaf score and avoid over-fitting. Assume the current stage is t, to solve optimization problem for objective function,

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t), \tag{26}$$

the algorithm uses second order Taylor expansion to make the prediction more accurate,

$$\begin{aligned} L^{(t)} &\simeq \sum_{i=1}^{n} [l(y_i, \hat{y}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \\ where \quad g_i &= \frac{\partial l(y_i, \hat{y}^{(t-1)})}{\partial \hat{y}^{(t-1)}} \quad and \quad h_i = \frac{\partial^2 l(y_i, \hat{y}^{(t-1)})}{\partial (\hat{y}^{(t-1)})^2}. \end{aligned} \tag{27}$$

After removing constant part, the simplified loss function at stage t can be written as,

$$\begin{aligned} \tilde{L}^{(t)} &= \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \eta T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2] + \eta T, \end{aligned} \tag{28}$$

To minimize the loss function, we set derivative as 0 to solve optimal number for weights as,

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}. \tag{29}$$

Thus, the optimal solution is,

$$\tilde{L}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \eta T. \tag{30}$$

In boosting tree problem, it is necessary to find best split point. Generally, the exact greedy algorithm is used because it can pick the optimal split point from all possible candidates. However, considering the memory cost, XGBoost also supports the approximate algorithm. The weighted quantile sketch is used to propose split candidates for equal weighted dataset. Let $D_k\{(x_{1k}, h_1), ..., (x_{nk}, h_n)\}$ represents the set of k-th feature value

and second order gradient for training set and feature are distributed by its percentile, the rank functions r_k,

$$r_k(z) = \frac{1}{\sum_{(x,h)\in D_k}} \sum_{(x,h0\in D_k, x<z} h, \tag{31}$$

and the candidate split point $\{s_{k1}, ..., s_{kl}\}$ defined as,

$$|r_k(s_{k,j}) - r_k(s_{k,j+1})| < \epsilon, \tag{32}$$

where ϵ is approximation factor. The sparsity-aware split finding handles the situation such as missing data, categorical variable and frequent zero entries. The algorithm can assign the optimal value to missing data by learning from data or user directly specified value.

As for the hardware perspective, XGBoost designs block to store data by compressed column format, and each column stores ordered feature value. In approximate algorithm, different subset of dataset can be stored in multiple block and allows parallel algorithm to find split point. The block structure can improve the computation burden of split finding. If the data size is too large to fit in RAM, blocks store on disk in the out-of-core setting. In order to avoid cache miss or running out of memory, cache-aware prefetching algorithm is designed for exact greedy algorithm. The basic idea of this algorithm is to fetch the statistics and store in internal buffer, which is allocated in each thread. Thus, the computation and read/write can be processed at same time. There are two techniques used to improve out-of-core computation, which are block compression and block sharding.

3.5 Random Forest

Random forest [1] is another ensemble learning algorithm, which belong to bagging method that is different from the boosting method introduced in previous sections. The key features for random forest are training data is randomly sampled to build trees and predictors are picked from random subset to split nodes. It only involves two parameters which are number of variables randomly samples as candidates at each split (m_{try}) and number of trees to grow (n_{tree}). For both regression and classification problem, the algorithm for random forest states [17],

1. Draw n_{tree} bootstrap samples from the original data. At training time, each tree learns from a random sample of dataset with replacement. Even though each tree

has strong correlation with particular sampled data, the entire forest can reduce the variance and avoid over-fitting.

2. For each bootstrap samples, the split point is chosen from random sampled m_{try}. Generally, the optimal value for m_{try} is square root of total number of feature.

3. The prediction are made by aggregating the prediction from n_{tree} trees.

The out-of-bag error can be criteria for parameter tuning. At each bootstrap, predict the data that exclude from bootstrap sample by trees trained with bootstrap sample. The data that exclude from bootstrap data is called out-of-bag (OOB) data. The error rate of model can be estimated by OOB predictions. By using grid search approach to find possible combination of parameters, the combination with lowest OOB error is selected to fit final model.

4. Model Assessment

4.1 Gini Index

The Lorenz curve [18] is proposed in 1905, which is used represents the distribution of income. In the economy perspective, x-axis represents the proportion of population and y-axis represents the proportion of income. If the income distribution is perfect match with population distribution, the Lorenz curve gives 45 degree line, which is called "Line of equality". The area between line of equality and Lorenz curve indicates the discrepancy between the income distribution and population distribution. The Gini index is calculated by two times of that area [9]. In the cases of insurance claim, the Lorenz curve used to compare a premium to a loss distribution. In order to consistent the policyholder groups, the relativity premium is used to connect the losses to the premium.

The calculation of Gini index and concept of ordered Lorenz curve are explain in [6]. Let B(x) represent the "base premium", which specified as each model. Let C(x) be the "competing premium", which are remaining model. Each observation is sorted by relativity R(x),

$$R(x) = \frac{C(x)}{B(x)}. \tag{33}$$

The ordered premium distribution is,

$$\hat{D}_p(s) = \frac{\sum_{i=1}^{n} B(x_i) I(R(x_i) \leq s)}{\sum_{i=1}^{n} B(x_i)}, \tag{34}$$

and the ordered loss distribution is,

$$\hat{D}_L(s) = \frac{\sum_{i=1}^{n} y_i I(R(x_i) \leq s)}{\sum_{i=1}^{n} y_i}. \tag{35}$$

The graph of ordered Lorenz curve has coordinate $(\hat{D}_p(s), \hat{D}_L(s))$. The Gini index in our case is twice area between line of equality and Lorenz curve [28] because the curves below the line of equality represents a profitable situation for insurer.

The selected score for Gini index is based on "mini-max" strategy, which is selected from all maximum score from competing premium with respect to each base premium, and select the maximum score among them. The corresponding base premium is the best choice for Gini index. The reason to use "mini-max" strategy is the maximum value in first step indicate the largest separation between the loss distribution and premium distribution, and the minimum value in second step indicates the score is least vulnerable to alternative scores. [6]

4.2 Mean Absolute Error

The mean absolute error is used to measure the model error. The MAE is calculated as,

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|. \tag{36}$$

The MAE can measure the closeness of the prediction value to the exact response value. Even though the distribution of insurance claim data contains high proportion of zeros and highly right skewed, the MAE is the more intuitive approach to determine which model is more accurate. The test are computed for every trained model. Thus, model with the lowest MAE can be considered as the most accurate one.

4.3 Bootstrap

Bootstrap is another method to validate models. It creates samples with replacement

from training data n times. The model is used to fit into sampled data, and use testing data to calculate prediction error. Finally, repeating above procedure K times, and calculate the variance for prediction error to measure how robust each model is to change of training data.

Let $\bar{\theta}$ be the bootstrap of prediction error and θ_i be the prediction error that fit into i-th resampled training data. The bootstrap variance is measured as,

$$\hat{Var}_\theta = \frac{1}{K-1}\sum_{i=1}^{K}(\theta_i - \bar{\theta})^2. \tag{37}$$

5. Application and Results

5.1 Data

The auto-insurance data we analyzed is retrieved from the SAS Enterprise Miner database [29]. There are totally 10,296 policyholders' record. In our analysis, we use 16 predictors, which consists of 8 numerical variables and 8 categorical variables. The value of response variable, pure premium, is calculated by ratio between total claims and duration, where the duration is assumed at 5. Table 1 provides detailed information about the involved predictors we used in our model.

As for data pre-processing, we take logarithmic transformation to BLUEBOOK and standardise the AGE, RETAINED, TRAVTIME, and transformed BLUEBOOK with mean 0 and standard deviation 1. The dummy variables are created for categorical variables with more than two levels, such as, JOBCLASS, MAX_EDUC and CAR_TYPE. The distribution of response variable is shown in Figure 1, which exhibits highly right skewed.There are approximately 61% policyholders do not have claims and 29.6% policyholders have claims up to $2,000. Even though those two groups cover the largest proportion of number of claims, the sum of their claims amount only comprises 36% of total claim amount. The zero-inflated data can affect the prediction error significantly.

The training set comprises 50% of data used for mode training, and the remaining 50% of data used for model evaluation.

Variable	Type	Description
AGE	Numerical	Driver's Age
BLUEBOOK	Numerical	Value of Vehicle
HOMEKIDS	Numerical	Number of children
KIDSDRIV	Numerical	Number of driving children
MVR_PTS	Numerical	MVR violation records
NPOLICY	Numerical	Number of policies
RETAINED	Numerical	Number of years as customer
TRAVTIME	Numerical	Distance to work
AREA	Categorical	Home/work area:"Rural", "Urban".
CAR_USE	Categorical	Primary use of the vehicle: "Commercial", "Private". 9. BLUEBOOK:"
CAR_TYPE	Categorical	Type of the car: "Panel Truck", "Pickup", "Sedan", "Sports Car", "SUV", "Van".
GENDER	Categorical	Gender of the driver: "F", "M".
JOBCLASS	Categorical	"Unknown", "Blue Collar", "Clerical", "Doctor", "Home Maker", "Lawyer", "Manager", "Professional", "Student".
MAX_EDUC	Categorical	Max education level:"¡High School", "Bachelors", "High School", "Masters", "PhD".
MARRIED	Categorical	Married or not: "No", "Yes".
REVOKED	Categorical	Whether the dirver's license was invoked in the past 7 years: "No", "Yes"

Table 1: Information for predictors in auto-insurance claim data

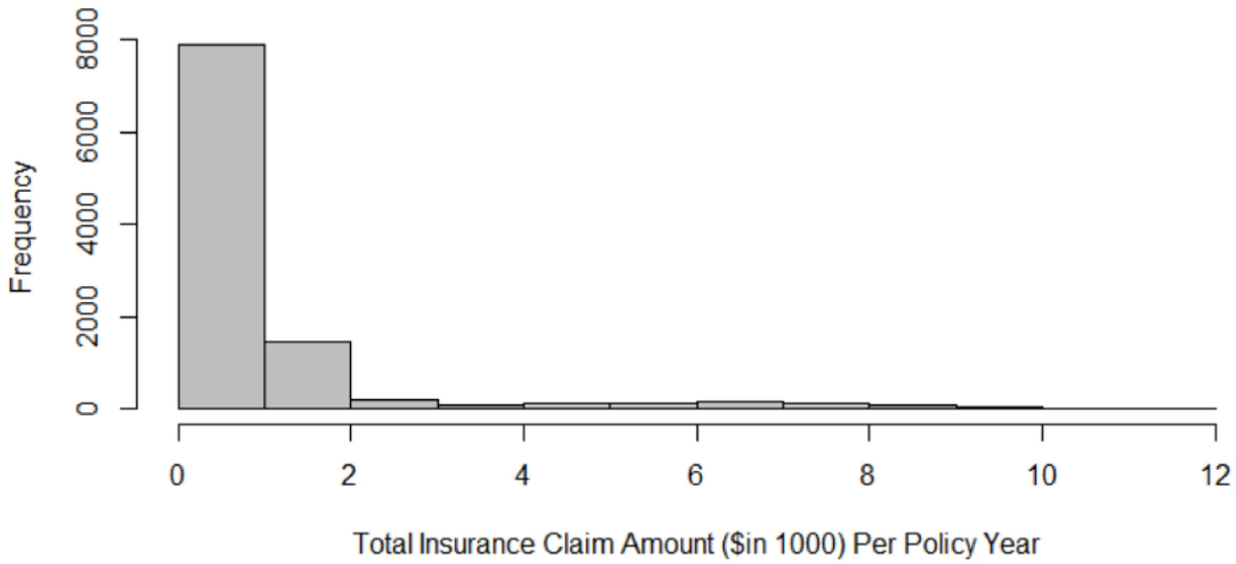

Figure 1: Histogram of Total Insurance claim amount

5.2 Tuning Parameter

5.2.1 Tweedie Distribution

The index and dispersion parameters need to be estimated by profile likelihood. The index parameter are split into 50 numbers with equal spaced value from 1.1 to 1.9. The profile log-likelihood plot shows in Figure 2. The curve represents the profile likelihood function of ρ. The dotted line shoes the estimated value $\rho^* = 1.361224$ corresponding to the maximum likelihood. The corresponded dispersion is $\phi^* = 2.305138$.

5.2.2 Model Hyperparameter

The hyperparameter in model cannot be estimated from data, which is external to model. In our application, we use the grid search approach to tune the hyperparameter for XGBoost and random forest. The grid search is used to find the optimal set of hyperparameters that has the lowest prediction error.

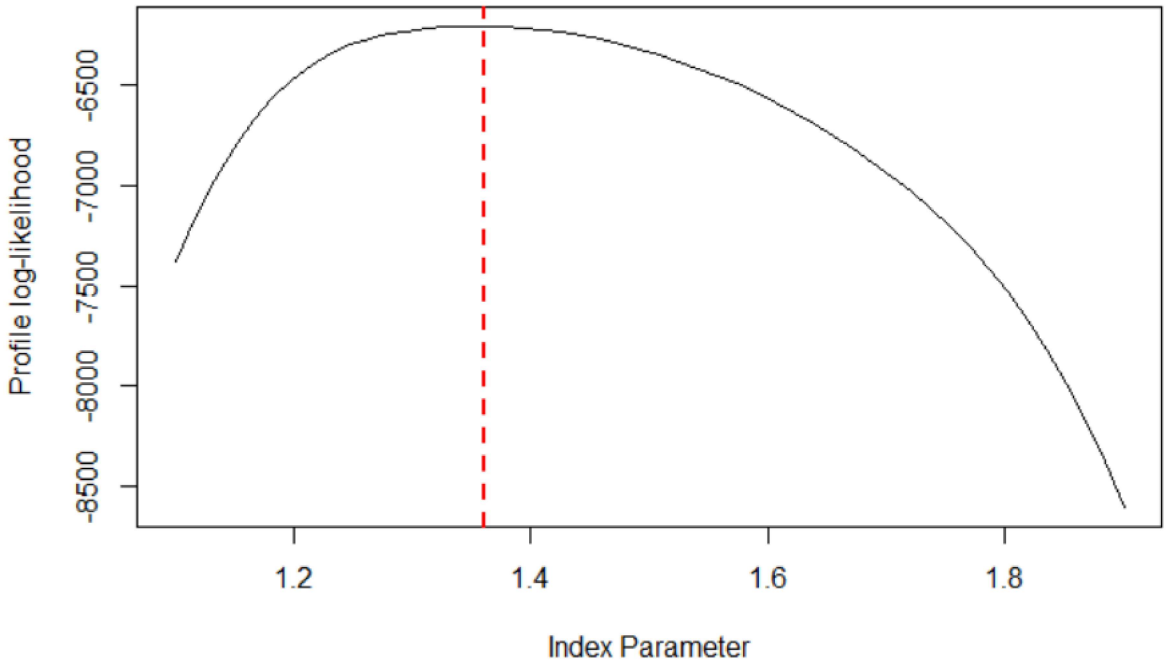

Figure 2: Profile Log-likelihood function

The tuned hyperparameteter for XGBoost are,

1. lambda: L2 regularization term on weights. The model will be more conservative, if lambda is large.
2. subsample: ratio to resample the training data, which used to prevent overfitting. The subsample occur once in each iteration. Range is between 0 and 1.
3. colsample_bytree: ratio to resample columns when construct each tree. It occurs at each iteration. Range is between 0 and 1.

The tuning result for XGBoost is listed in Table 2. The lowest test mean absolute error is preferred. Thus, {lambda, SubSample, SubColumn} = {1, 0.5, 0.6} for XGBoost.

The tuned hyperparameter for random forest are,

1. mtry: Number of variables that randomly selected as candidates at each split.

	Train-mae	Test-mae	SubSample	SubColumn	lambda
1	0.7498	0.8360	0.5	0.5	1
2	0.7588	0.8299	0.6	0.5	1
3	0.7386	0.8200	0.5	0.6	1
4	0.7555	0.8408	0.6	0.6	1
5	0.7450	0.8246	0.5	0.5	3
6	0.7553	0.8243	0.6	0.5	3
7	0.7590	0.8275	0.5	0.6	3
8	0.7563	0.8360	0.6	0.6	3

Table 2: XGBoost tuning result

2. node_size: Minimum size of terminal nodes. The larger node size, the smaller tree to grow.

3. sample_size: ratio to resample observations from original data.

The tuned result for random forest is listed in Table 3. The lowest out-of-bag error is preferred. Thus, {mtry, node_size, sample_size} = {10, 14, 0.632} for random forest.

	mtry	node_size	sample_size	OOB_RMSE
1	10	6	0.55	1.522354
2	11	11	0.55	1.521293
3	10	12	0.55	1.522118
4	10	13	0.55	1.522575
5	10	14	0.55	1.521393
6	10	15	0.55	1.522322
7	12	12	0.632	1.522569
8	12	13	0.632	1.522434
9	10	14	0.632	1.521256
10	10	14	0.7	1.522423

Table 3: Random Forest tuning result

5.3 Results

5.3.1 Gini Index

Table 4 shows result of Gini index. The models list in column is base premium and list in row is competing premium. To use "mini-max" strategy, we pick the most competitive model for each base premium, which are all XGBoost (12.858, 9.844, 7.795, and 14.801, respectively) except when XGBoost as base premium, 5.940. Among those candidates, 5.940 is the smallest value and it is attributed to XGBoost. Thus, the selected score for Gini index is XGBoost. The ordered Lorenz curve is displayed in Figure 3, where the black line is the line of equality and the coloured line the ordered Lorenz curve. The Figure also shows that the area between line of equality and the ordered Lorenz curve is relatively larger when picking XGBoost as competing premium and remaining model as base premium, which again supports XGBoost is the "best" model. In order to ensure the consistency of results, we randomly split original data 5 times as train set and test set, and train each time including the tuning parameters. The finalized Gini index is calculated by taking average of those 5 results. Table 5 shows the result of averaged Gini Indices, which provides additional confirmation that the XGBoost is the choice for Gini index.

	GAM	TDboost	GBM	XGBoost	RandomForest
GAM	0.000	9.670	11.215	12.858	9.710
TDboost	6.622	0.000	7.654	9.844	5.161
GBM	1.263	1.185	0.000	7.798	4.110
XGBoost	5.082	4.552	5.940	0.000	1.072
RandomForest	12.742	14.012	14.731	14.801	0.000

Table 4: Gini Indices

5.3.2 Mean Absolute Error

The procedure to measure more consistent result for mean absolute error is similar as computing Gini index. We trained each model by different training and testing set for 5 times, and take the average to measure which model is the most accurate. The corresponding prediction errors are shown in Table 6.

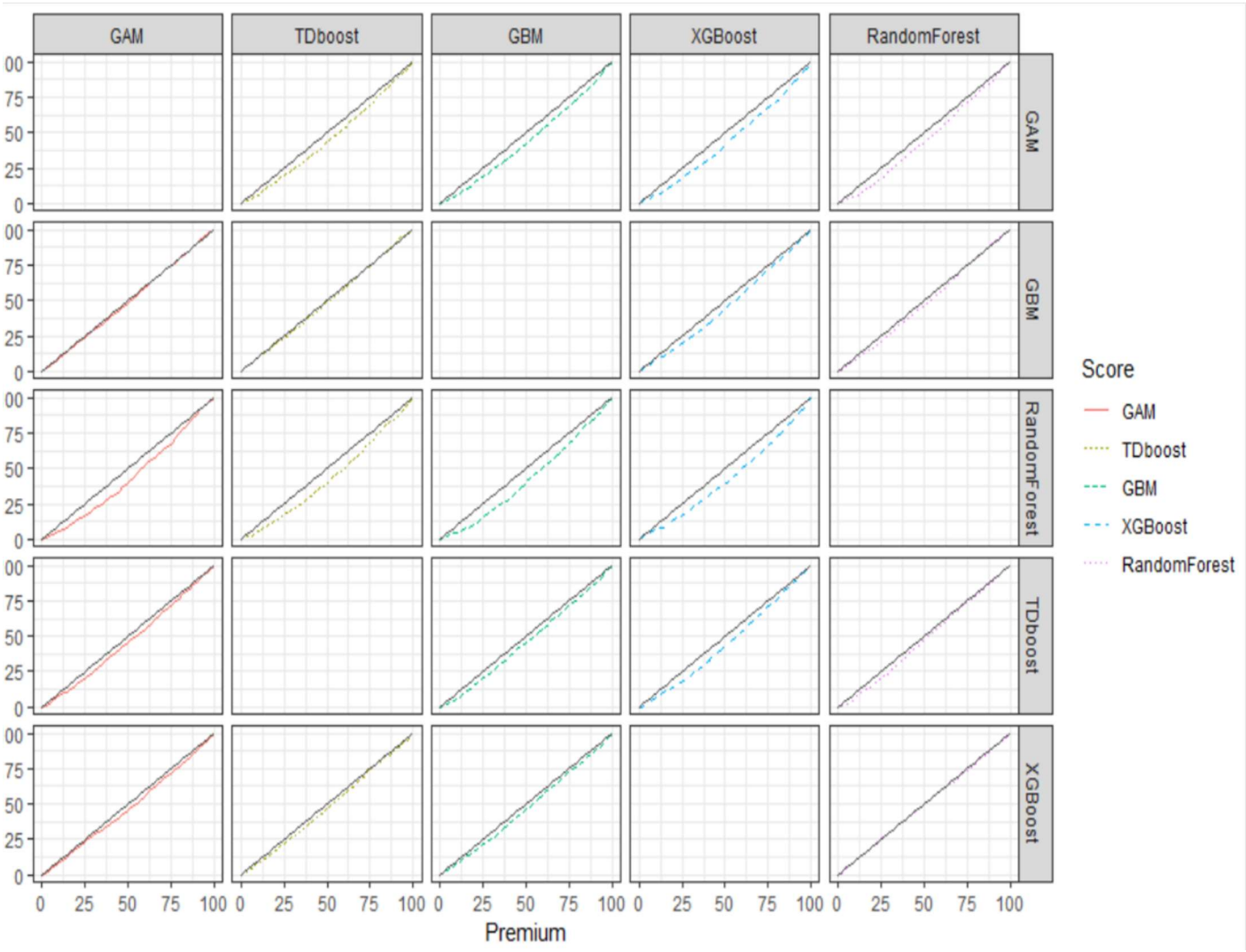

Figure 3: The ordered Lorenz Curve for auto-insurance data

The value of error is larger than usual because of the imbalanced value of response variable. As expected, the XGBoost performs best on test set every time. The random forest performs worst among those 5 models. Random forest is poorer to handle such regression problem with imbalanced data.

	GAM	TDboost	GBM	XGBoost	RandomForest
GAM	0.000	6.595	8.574	10.920	12.487
TDboost	9.078	0.000	8.772	11.863	12.637
GBM	5.168	-0.912	0.000	8.406	9.664
XGBoost	6.239	3.037	5.606	0.000	7.158
RandomForest	8.479	6.354	8.070	7.682	0.000

Table 5: Averaged Gini Indices

MAE	GAM	TDboost	GBM	XGBoost	RandomForest
1st Iteraion	0.8688	0.8397	0.8411	0.8317	0.8848
2nd Iteration	0.8428	0.8291	0.8267	0.8157	0.8610
3rd Iteration	0.8554	0.8257	0.8207	0.8089	0.8569
4th Iteration	0.8375	0.8229	0.8242	0.8030	0.8487
5th Iteration	0.8422	0.8126	0.8210	0.8056	0.8661
Average	0.8493	0.8260	0.8267	0.8130	0.8555

Table 6: Mean Absolute Error

5.3.3 Bootstrap

Variance RMSE$_{\times}10^{-4}$	GAM	TDboost	GBM	XGBoost	RandomForest
1st Iteraion	4.645	4.959	4.880	4.567	4.016
2nd Iteration	5.162	5.614	5.674	5.531	4.082
3rd Iteration	3.718	4.324	4.000	4.942	3.242
4th Iteration	5.246	5.609	5.516	5.431	4.171
5th Iteration	6.126	7.344	6.619	7.250	5.511
Average	4.980	5.570	5.556	5.544	4.204

Table 7: Bootstrap Variance in RMSE

The estimation of prediction error variance by bootstrap is presented in Table 7. For each iteration, 50 re-sampled bootstraps are used to perform for all models. The result shows the random forest has lowest variance in RMSE, which indicates random forest is more robust to alternative dataset. However, the XGBoost is quite sensitive to the change

of data.

5.3.4 Computational Time

The computational time for each model in each iteration is presented in Table 8. The time only involves the training time, and excluding the tuning part. Random forest costs the least amount of time, even 20 to 30 times less than other models. The time consume for XGBoost and Gradient boosting also indicates the system design for XGBoost saves half of time based on gradient boosting.

Computational Time/Sec	GAM	TDboost	GBM	XGBoost	RandomForest
1st Iteraion	13.47	69.7	111.7	45.75	3.27
2nd Iteration	28.16	68.41	107.17	43.87	3.58
3rd Iteration	23.27	77.63	126.85	62.64	3.5
4th Iteration	13.13	68.59	109.36	43.91	3.76
5th Iteration	34.02	71.61	111.53	46.09	3.73
Average	22.41	71.19	113.32	48.45	3.57

Table 8: Computational Time

5.4 Model Interpretation

5.4.1 SHAP Values

The SHAP (SHapley Additive exPlanation) values [19] can deal with the inconsistent problems that current feature attribution methods have for tree ensembles. [25] has proved the SHAP values are the only consistent and locally accurate feature attribution for tree ensembles. The prediction explanation methods use the features and original complex model as input, and output an explanation for prediction, which can regards as explanation model. The most explanation models have the same form, called Additive feature attribution methods,

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z_i' \tag{38}$$

where g is explanation model, z'$\in \{0,1\}^M$, M is the number of feature, and $\phi_i \in \mathbb{R}$. The effect of observation or nor observing a feature is calculated by define function h_x, which maps pattern of missing feature z'_i to feature space of original complex model f. Let S be the set of non-zeros indexes in z', and $E[f(x)|x_S]$ is expected value of function condition on a subset S of input feature. The SHAP value is calculated by [25],

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(M - |S| - 1)!}{M!} [f_x(S \cup \{i\}) - f_x(S)], \tag{39}$$

where $f_x(S) = f(h_x(z')) = E[f(x)|x_S]$, and N is set of all input feature.

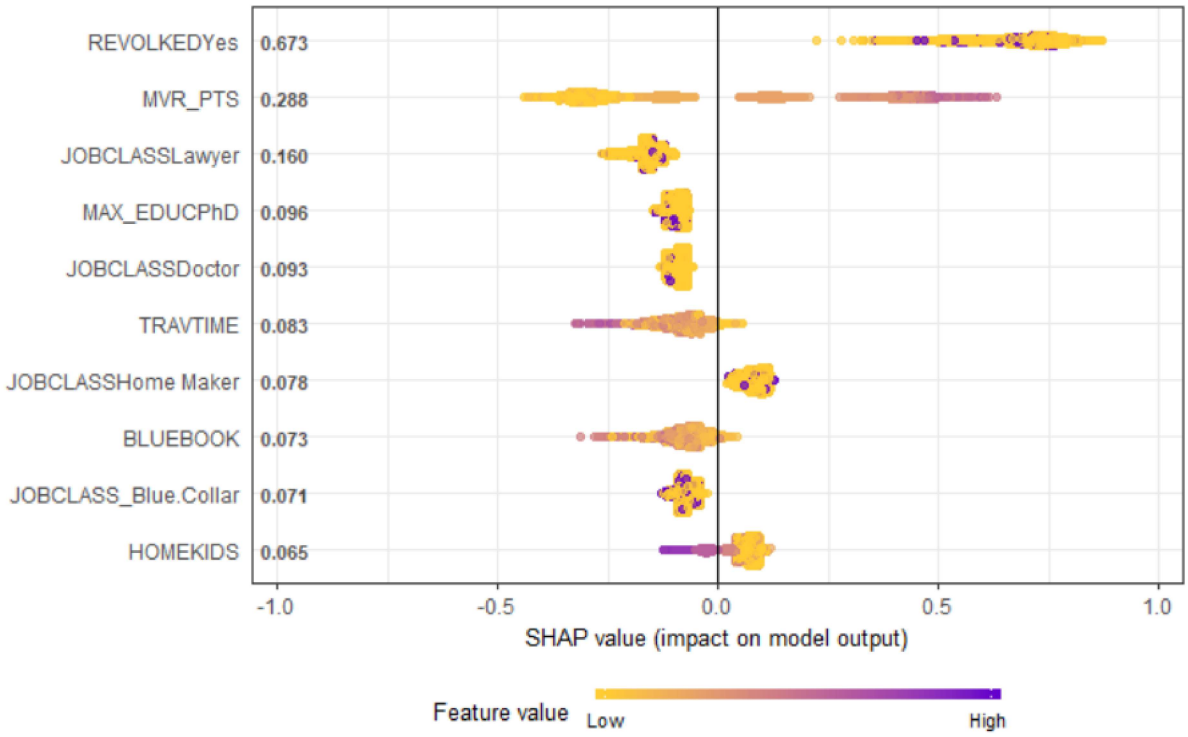

Figure 4: SHAP summary plot for XGBoost on auto-insurance claim data

The visualization of SHAP value is called SHAP summary plot. The traditional variables importance plot only presents the global feature importance by bar chart. SHAP summary plot presents how the distribution and range of each feature impacts prediction and how feature value correlates to the impact. The feature sorted by global importance

$\sum_{j=1}^{N} |\phi_i^{(j)}|$, which ranked in descending order. Each dot in plot represents an observation corresponding to its SHAP value. The horizontal location indicates whether the effect of feature value is associated with lower or higher prediction. The color represents the value of feature [5]. Figure 4 presents the SHAP value for XGBoost. Revoked (whether license revoked in 7 years) at baseline is the most important factor for insurance premium. The color has smooth gradient if the impact for output has changes as the changes of the feature value smoothly. The density of MVR_PTS (Motor vehicle record point) plot shows how different record points are in data, and the gradient colors show smooth increase in prediction as record points increase. The majority of people have negative impact on prediction with whether policyholders are group of laywer, PHD, doctor or blue collar.

5.4.2 SHAP Interaction Value

The interaction effect of pairs of features on prediction can be calculated by Shapley interaction index [16],

$$\Phi_{i,j} = \sum_{S \subseteq N \setminus \{i,j\}} \frac{|S|!(M - |S| - 2)!}{2(M-1)!} \nabla_{ij}(S), \tag{40}$$

when $i \neq j$, and

$$\begin{aligned} \nabla_{ij}(S) &= f_x(S \cup \{i,j\}) - f_x(S \cup \{i\}) - f_x(S \cup \{j\}) + f_x(S) \\ &= f_x(S \cup \{i,j\}) - f_x(S \cup \{j\}) - [f_x(S \cup \{i\}) - f_x(S)]. \end{aligned} \tag{41}$$

In Equation (40) the SHAP interaction value is same between feature i and feature j, where $\Phi_{i,j} = \Phi_{j,i}$. The main effect for model result is defined as,

$$\Phi_{i,i} = \phi_i - \sum_{j \neq i} \Phi_{i,j}. \tag{42}$$

The partial SHAP dependence plots shows the impact on model output when the values of target features are fixed. The plot explains how model correlates to the features. The x-axis is the value of feature and the y-axis is the SHAP value of that feature. The dots still represent the observation in data and trend of curve shows how feature attribution changes as its value changes. The value of interacting feature is colored in plot. In Figure 5, the red curve is LOESS (locally estimated scatterplot smoothing). It shows the low value of vehicle is more preferred when the distance to work is short.

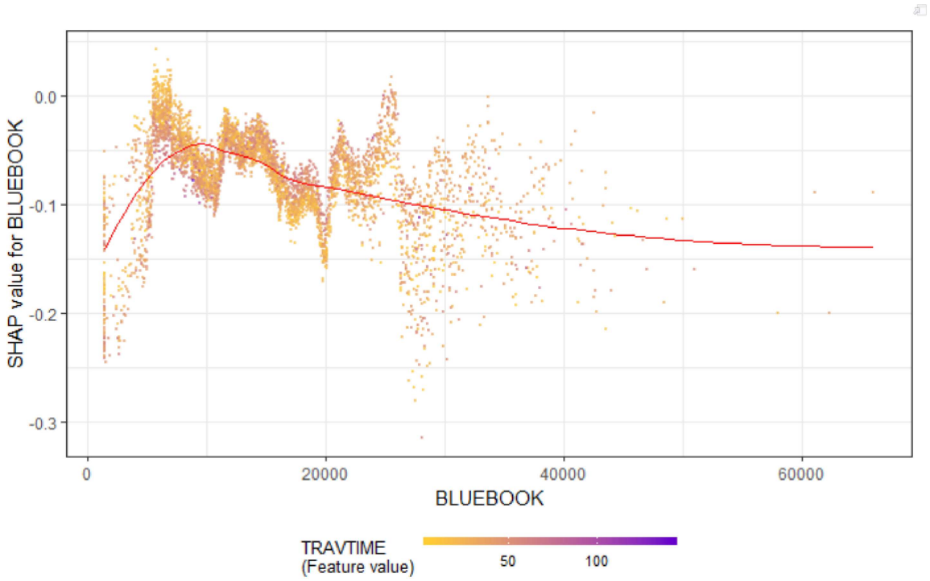

Figure 5: SHAP dependence plot for XGBoost on auto-insurance claim data

6. Conclusion

This research explores 5 machine learning methods to predict the pure premium for auto-insurance. Prior to training the models, defining the distribution of data is extremely crucial step. The TDBoost proposed in 2018 focuses on predicting the zero-inflated insurance data. As we assign the Tweedie distribution to GAM, gradient boosting, and XGBoost, they all have the better performance and comparable to TDboost. However, since the random forest is an ensemble of unpruned trees, the prediction error is not as expected as other methods. The random forest is a user friendly algorithm, which not only has the lowest computation consuming, but also easy for tuning parameter. Tweedie GAM is widely used in acturial science for application in insurance, which is a strong statistical model. It has statistical tools for the further investigation and has strong interpretability. This research is a preliminary particle experiment to try different methods for predicting premiums.

In the model assessments, we use the ordered Lorentz curve and the associated Gini

index as one of the criteria, which is a statistical measure to evaluate the performance of each model. The XGBoost outperforms other models, which is an insight for further research to focus on XGBoost extension. The extension is not limited by improving accuracy, but also the interpretability for complicated situation and more tends to statistical perspectives.

References

[1] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[2] Andy Liaw Chao Chen and Leo Breiman. Using random forest to learn imbalanced data. *University of California, Berkeley*, 2004.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. 2016.

[4] Money Control. Pure premium. https://www.moneycontrol.com/glossary/insurance/pure-premium_1435.html.

[5] Dataman. Explain your model with the shap values. *Toward Data Science*, 2019.

[6] Glenn Meyers Edward W. (Jed) Frees and A. David Cummings. Insurance ratemaking and a gini index. *Journal of Risk and Insurance*, 81(2), 2014.

[7] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29:1189–1232, 2001.

[8] Jerome H. Friedman and Jacqueline J. Meulman. Multiple additive regression trees with application in epidemiology. *Special Issue: 8th Biennial CDC and ADSTR Symposium on Statistical Methods Issues Associated with Complicated Designs and Data Structure*, 22(9):1365–1381, 2003.

[9] Corrado Gini. Variabilità e mutabilità. *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi*, 1912.

[10] Trevor Hastie and Robert Tibshirani. Generalized additive models; some applications. *Lecture Notes in Statistics*, 82(398):371–386, 1987.

[11] Ayantee Jana and Sujit Ray. Generalized additive model. https://www.youtube.com/watch?v=jS47pBu_gN8&t=247s, 2015.

[12] Trevor Hastie Jerome Friedman and Robert Tibshirani. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[13] Bent Jorgensen. Exponential dispersion models. *Journal of the Royal Statistical Society*, 49:127–162, 1987.

[14] Bent Jorgensen and Marta C. Paes De Souza. Fitting tweedie's compound poisson model to insurance claims data. *Scandinavian Actuarial Journal*, (1):69–93, 1994.

[15] Rob Kaas. Compound poisson distribution and glms—tweedie's distribution. *In Handelingen van het Contactforum*, pages 3–43, 2001.

[16] Ivan Kojadinovic Katsushige Fujimoto and Jean-Luc Marichal. Axiomatic characterizations of probabilistic and cardinal-probabilistic interaction indices. *Games and Economic Behavior*, 55:72–99, 2006.

[17] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *Forest*, 23, 2001.

[18] M. O. Lorenz. Methods of measuring the concentration of wealth. *Publications of the American Statistical Association 9*, 70:209–219, 1905.

[19] Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.

[20] RAVI MALHOTRA and SWATI SHARMA. Machine learning in insurance.

[21] Alexey Natekin and Alois Knoll. Gradient boosting machiens, tutorial. 2013.

[22] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society*, 135:370–384, 1972.

[23] Srinivas Vadrevu Kilian Weinberger Ya Zhang Oliver Chapelle, Pannagadatta Shivaswamy and Belle Tseng. Multi-task learning for boosting with application to web search ranking. *ACM New York, NY, USA@2010*, pages 1189–1198, 2010.

[24] Szilard Pafka. Benchmarking random forest implementation. *DataScience.LA*, 2015.

[25] Gabriel G. Erion Scott M. Lundberg and Su-In Lee. Consistent feature attribution for tree ensembles. *CoRR*, 2019.

[26] Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[27] Thomas W. Yee and Neil D. Mitchell. Generalized additive models in plant ecology. *Journal of Vegetation Science*, 2(5):587–602, 1991.

[28] Wei Qian Yi Yang and Hui Zou. Insurance premium prediction via gradient tree-boosted tweedie compound poisson. *American Statistical Association Journal of Business Economic Statistics*, 36(3), 2018.

[29] Karen C.H. Yip and Kelvin K. W. Yau. On modeling claim frequency data in general insurance with extra zeros. *Insurance: Mathematics and Economics*, 36:153–163, 2005.