XBO Model: a Useful Method for Optimizing the Efficiency of Life Insurance Companies

Kesen Zhang*

Department of Mathematics, King's College, London, United Kingdom

*Corresponding author: kesen_zhang@protonmail.com

Keywords: Customer Assessment, XBO model, Life Insurance Company.

Abstract: A life insurance policy can pay off any debt one left behind, which will put a burden on the family. Debts such as mortgages, credit cards, car loans, and even the funeral expenses can have a significant impact on one's family and their lifestyle. In a one-click shopping world with on-demand everything, only 40% of US household own individual life insurance due to the antiquated application process from life insurance companies. Customers need to provide extensive information for identification of risk classification and eligibility, which includes scheduling medical exams, a process that takes an average of 30 days. Gradually, customers tend to fall off. Here we propose an approach named 'XGBoost Based on Offset' estimator to solve such a problem. XBO is a predictive model that accurately classifies customer's level using a more automated approach. With technology frameworks and the application of mathematical statistics, we show that XBO estimator makes the identification quicker and less labour-intensive for new and existing customers while maintaining privacy boundaries. By automatically separating clients to eight ranks, this model can facilitate the pricing process of life insurance policy and can make a difference for the life insurance companies' risk management.

1. Introduction

In contemporary, life insurance is essential; as it protects one's family and lets them leave them a non-taxable amount at the time of death. It is also used to cover one's mortgage and loans, such as car loan [1]. Typically, life insurance is determined based on the needs and goals of the owner. In exchange for premium payments, the insurance company provides a lump-sum payment, known as a death benefit, to beneficiaries upon the insured's death.

XGBoost is short for eXtreme Gradient Boosting package. It is an efficient and scalable implementation of gradient boosting framework by additive logistic regression and greedy function approximation [2][3]. The kit includes an efficient linear model solver and tree-learning algorithm. It supports various objective functions, including regression, classification and ranking. The package is made to be extendible so that users are allowed to define their own objectives easily [4].

However, in the US, only 40% of households have individual life insurance. The complex application process mainly causes this problem. For instance, if a person wants to buy insurance, in addition to choosing the right insurance company and package, he must also take an average 30-day medical examination by a third-party agency. The complicated and lengthy process makes many interested customers find it troublesome. This situation has a tremendous negative impact on both insurance companies and consumers. Insurance companies cannot get more customers, and consumers cannot get protection for themselves and their families. The work presented in this paper provides a model and a method to generate data with the help of a model that we named 'XGBoost Based on Offset' estimator to solve such a problem.

2. Data and Methods

2.1 Data

In the dataset, we are provided over a hundred variables describing attributes of life insurance applicants, and a "Response" variable, which is an ordinal measure of risk that has eight levels. Besides, The data volume of the dataset is 59381.The main focus and goal for the model are to yield another completely self-sufficient data set with the aim of having similar statistical properties as the original data set. To yield such results, the model must go through several steps to be able to complete.

Firstly, we chose Random Forest to calculate the importance of each feature. Because there are too many features (around 125), we believed that selecting some of the most important feature for training would have better results. To simulate the insurance evaluation process accurately, we decided to cover 15 of the most important features and the results have been shown in Table 1.

Table 1. The most important features and corresponding explanation.

Feature Name	Explanation
BMI	Normalized BMI of the applicant
BMI_Age	Normalized age of the applicant
Ins_Age	Combination of BMI and Ins_Age
Ht	Normalized height of the applicant
Wt	Normalized weight of the applicant
Product_Info_1-7	A set of normalized variables relating to the product applied for
Employment_Info_1-6	A set of normalized variables relating to the employment history of the applicant
InsuredInfo_1-6	A set of normalized variables providing information about the applicant
Insurance_History_1-9	A set of normalized variables relating to the insurance history of the applicant
Family_Hist_1-5	A set of normalized variables relating to the family history of the applicant
Medical_History_1-41	A set of normalized variables relating to the medical history of the applicant
Medical_Keyword_1-48	A set of dummy variables relating to the presence of/absence of a medical keyword being associated with the application
Product_Info_2_char	A set of characters in Product_Info_2
Product_Info_2_num	A set of numbers in Product_Info_2
Medical_Keywords_Count	The sum of figures in Medical_Keyword_1-48

Data sources: The original dataset comes from the Kaggle website [5].

There are other features that we decided to exclude from the estimator due to the low percentage of data found in the sample, such as Id, Medical_History_32, Medical_History_24,

Medical_History_15, Medical_History_10 and Family_Hist_5.

2.2 Methods

2.2.1 Data Processing

There are two different ways for us to deal with missing values. One is to delete the missing value; the other one is using other value to fill the blank. Firstly, we decided to delete features with a missing rate of more than 60%. For the rest of continuous data, we put the mean value into the blank and put the median value into blank for discrete data. Besides, for XGBoost, we use an extreme value (such as -10000) to fill the blank due to its feature. We also found that the data is concentrated in an integer range. However, in terms of the quantity, the amount may be similar to the number of medical treatments. Therefore, it is not a category feature, so we use minmaxscaler to scale the features to range. Min-max scaling (many people call this normalization) is quite simple: values are shifted and rescaled so that they end up ranging from 0 to 1. We do this by subtracting the min value and dividing by the max minus the min [6]. Then we choose RobustScaler to make the data more descriptive. Besides, we found in the process of exploring that if the two features are multiplied, the variance will be smaller. Thus, we combined BMI and Ins_Age.

2.2.2 Package Selection

XBO uses Numpy, Pandas, Matplotlib, Sklearn, Scipy and Seaborn, which are packages in Python.

We selected Numpy because it is the fundamental package for scientific computing with Python. Besides, it can also be used as an efficient multi-dimensional container for generic data. Arbitrary data-types can be defined. This allows NumPy to integrate seamlessly and speedily with a wide variety of databases [7]. We chose Pandas because it is an open-source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language [8]. The reason why we use Matplotlib is that it is a Python 2D plotting library that produces publication quality, figures in various hard copy formats and cross-platform interactive environments. Matplotlib tries to make easy things easy while making hard things possible [9]. We selected SciPy because it is a Python-based ecosystem of open-source software for mathematics, science, and engineering [10]. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other common tasks in science and engineering. Moreover, we use Sklearn because it is a free software machine-learning library for the Python programming language. It has various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN. It is designed to interoperate with the Python mathematical and scientific libraries NumPy and SciPy [11]. Finally, We chose Seaborn, as it is a Python data visualization library based on matplotlib. It provides an advanced interface for drawing attractive and informative statistical graphics [12].

3. XBO Model Implementation

First of all, we split the dataset to a training set and testing set, taking the test_size of 0.25. For the metrics, quadratic weighted kappa is used to measure the performance of the model. It calculates the similarity between our prediction and the actual values, ranging from minus one to one. The positive one means a perfect match, while the negative one means not. Generally, point six is a relatively good result.

According to the characteristics of the problem, we preliminarily judge that the problem belongs to classification; thus, tree-ensemble models are considered. Decision tree, random forest, gradient boosted decision tree, and XGBoost are used in turn, with grid search to find their respective better parameter values. When adjusting the objective learning function of XGBoost, it is found that linear regression can get the best results compared with other objectives. Therefore, linear, ridge, lasso regressions are taken into consideration for training, but their effect is found is not as good as XGBoost's. In summary, XGBoost model is finally chosen for learning.

After the optimal position of XGBoost is confirmed, some methods are tried to elevate the kappa score, such as deleting some of the previous preprocessing operations. The operations include restoring all dropped features, eliminating the use of random forest to filter functions, and eliminating scaling of data, and using. In essence, 10000 impute all missing values, which lead to a higher score. Therefore, XGBoost itself are found out to having an excellent adaptive ability.

In the model evaluation, we put the training set and testing set into the different models to test their kappa score and over-fitting degree. As can be seen from the Figure1, the accuracy from the decision tree to XGB is increasing. There are three main reasons why XGB has such excellent performance:

(1) XGBoost chooses to use L1 or L2 regularization to get a lower variance, thus avoiding over-fitting to some extent,

(2) XGBoost reduces over-fitting by column subsampling, and improves the speed of training, making the process of adjusting parameters more convenient,

(3) In dealing with missing values, for samples with missing values, XGBoost can automatically learn its splitting direction.

It is found that GBDT has a high degree of over-fitting-low training error vs high generalization error and accuracy will be sacrificed to diminish the overfit.

4. Results and Discussion

When trying to elevate the final assessing score, a unique offset method is discovered, referring to the senior's idea. In the process, eight figures applied to eight risk groups are estimated to offset some bias between the predicted result and actual result, thus improving the accuracy rate. Breaking down to steps, firstly data for different classification results are selected out to groups from 1 to 8. Secondly, 'fmin_powell' function is used to find the optimal offset figure for each group to minimize the self-created function 'f(x)', which is equal to maximize the score of 'predicted response plus offset over actual response'. Finally, these eight offset figures obtained from the train set will be applied into the test set to diminish bias, and increasing scores in kappa from 0.505 to 0.658 (Figure 1).

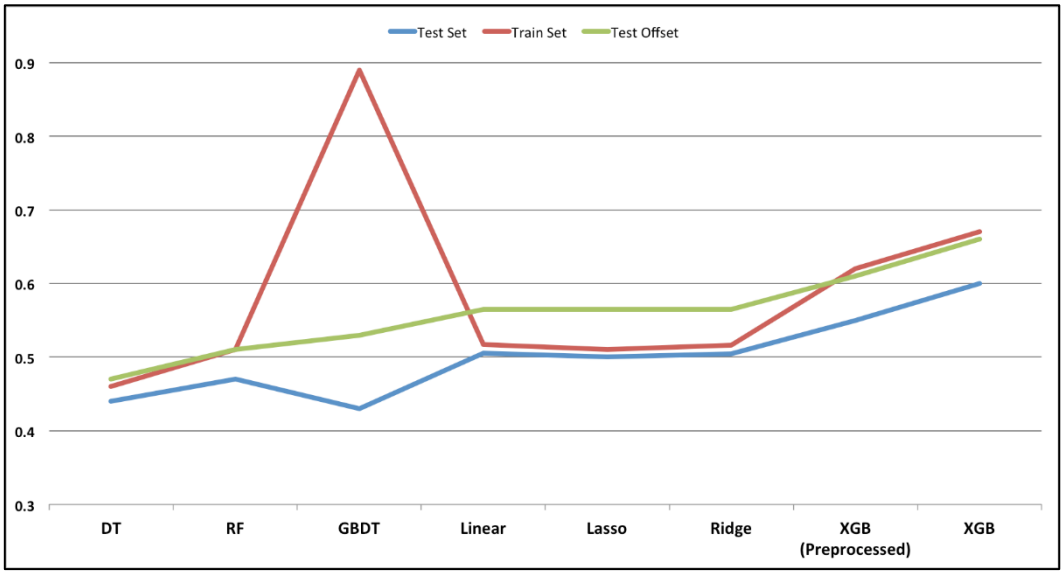

Figure 1. Quadratic Weighted Kappa of different models.

In summary, using "XGBoost model based on the offset" method, the final kappa score reached 0.658, while the 1st place winner achieves 0.679. Meanwhile, top five powerful features are figure out through built-in function in XGBoost model, all of which are continuous data and most are referring to a primary physical condition such as Body Mass Index (BMI), Age and Weight.

5. Conclusion

Through the application of the technical framework and mathematical statistics, we show that the XBO estimator can identify identities faster while ensuring the status of new and existing customers, and reduce labour intensity while maintaining privacy boundaries. Firstly, efficiency in processing insurance applications, young customers with reasonable body mass index and relatively good condition of family medical history, will be directly insured without taking a medical exam. Also, this technology makes online life insurance application possible in the future, if citizens' personal medical record is stored in the database, the insurance company can retrieve clients' ranks and arrange different policies respectively. Secondly, reducing risks, medical exam results and personal information collected together can form a double validation, making the premium formulation more rigorous, reducing the risk of insurance companies, and in some extreme cases, agents can be more confident to reject the insurance request from the clients for high chances. This model can make a difference in companies' risk management. Also, it can facilitate the pricing process of the life insurance policy; the model automatically separates clients to eight ranks. When the intermediary first meets with the customer, he or she can give the price with the assist of the model. It will be easier for him or her to decide the type of plan should be recommended according to the customer's economic situation.

The potential improvement can be applied to have a better prediction: as for business impact, note that the features' names in the given dataset are unclear; we cannot express the model with an explicit formula. The critical issue here is to find the boundaries that maximize the kappa score. In our case, we used manually designed offset with eight bias values to achieve this. Boundaries are initialized according to the original 'Response' distribution of the training dataset. To have a better fit, we can define a function to examine the boundary values in a small range around the current boundary. The steps are repeated until none of the limitations is changed during a stage. Also, a different method can be used to fill the NaN's, and the precision may improve for other models that are based on linear regression. Furthermore, it is possible to get the same precision level with fewer features selected, which will also reduce the computed duration.

Acknowledgements

I would like to thank Dr. Ren for his valuable feedback. I would also like to congratulate Tianqi Chen, Tong He and other contributors of the opensource xgboost package. This work was supported in part by Chenyu Xu, Zhaoning Qi and Xiabing Hu.

Reference

[1] Why Buy Life Insurance? What Is It Used For? How Do I Choose It? | iA Financial Group. [EB/OL]. [2020-04-12] https://ia.ca/advice-zone/family/why-take-life-insurance.

[2] Friedman J H. Greedy function approximation: a gradient boosting machine [J]. Annals of statistics, 2001: 1189-1232.

[3] Friedman J, Hastie T, Tibshirani R. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)[J]. The annals of statistics, 2000, 28(2): 337-407.

[4] Chen T, He T, Benesty M, et al. Xgboost: extreme gradient boosting [J]. R package version 0.4-2, 2015: 1-4.

[5] Prudential Life Insurance Assessment | Kaggle [EB/OL]. [2020-04-11] https://www.kaggle.com/c/prudential-life-insurance-assessment/rules.

[6] Géron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems [M]. O'Reilly Media, 2019.

[7] Numpy [EB/OL]. [2020-04-17] https://numpy.org.cn/en.

[8] Pandas Documentation — Pandas 1.0.3 Documentation [EB/OL]. [2020-04-10] https://pandas.pydata.org/docs.

[9] Matplotlib: Python Plotting — Matplotlib 3.1.3 Documentation [EB/OL]. [2020-04-11] https://matplotlib.org/3.1.3/index.html.

[10] Scipy.Org [EB/OL]. [2020-04-10] https://www.scipy.org.

[11] Rubia, E. Multicore Data Science With R And Python [EB/OL]. [2020-04-15] https://blog.dominodatalab.com/multicore-data-science-r-python.

[12] Anaconda. Seaborn: Anaconda Cloud [EB/OL]. [2020-04-16] https://anaconda.org/anaconda/seaborn.