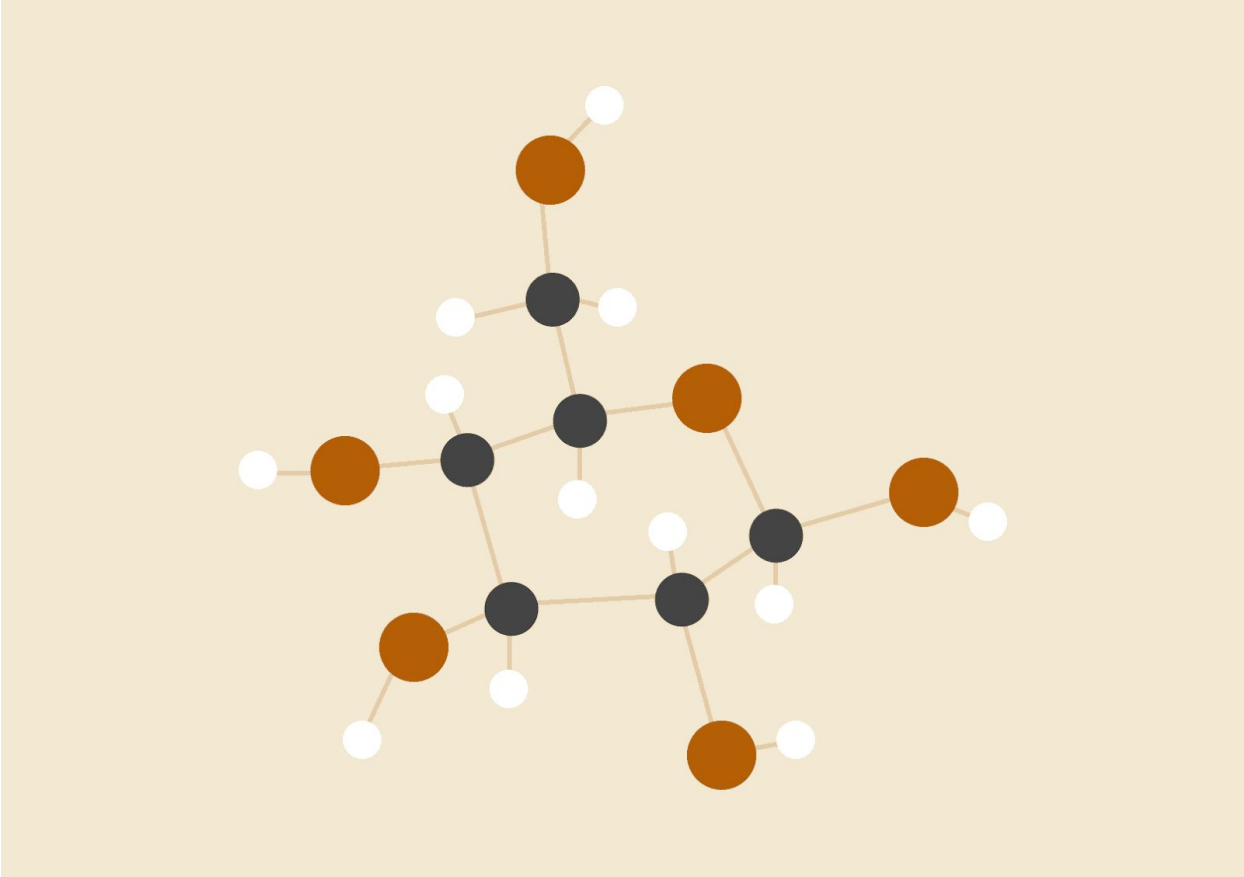
XBO Model: Prudential Life Insurance Assessment

Zhaoning Qi, Chenyu Xu, Kesen Zhang and Xiabing Hu



July 2019

ABSTRACT

In a one-click shopping world with on-demand everything, the life insurance application process is antiquated. Customers provide extensive information to identify risk classification and eligibility, including scheduling medical exams, a process that takes an average of 30 days. Gradually, people are turned off. Only 40% of US households own individual life insurance. This paper propose an approach to such a problem that we named the XBO estimator.

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 a solution in this paper that makes the identification quicker and less labor intensive for new and existing customers while maintaining privacy boundaries.

Keyword: Insurance Assessment, Mathematical statistics

1. INTRODUCTION

In the US, only 40% households have individual life insurance. This problem is mainly caused by the complex application process. For instance, if a person wants to buy an 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 long process gave up many interested customers. This situation has a very large negative impact on both insurance companies and consumers. Insurance companies can not get more customers, and consumers can not 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 XBO. XBO is a XGBoost based on offset model and with the help of statistic analysis, XBO is able to generate congruous results with the original data set.

Outline This paper is structured as follows: Section 2 presents the background in insurance industry. Section 3 states the problem and during section 4 we present the process of data preprocessing. In section 5 and 6 we present the implementation of XBO and the results of the model. Finally section 7 and 8 present the conclusions and future work.

2. BACKGROUND

A life insurance policy is a contract with an insurance company. 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.

Typically, life insurance is chosen based on the needs and goals of the owner. Term life insurance generally provides protection for a set period of time, while permanent insurance, such as whole and universal life, provides lifetime coverage.

In the middle of the 17th century, the Italian banker L.Tontine proposed a pension scheme, which was later called the “Tontine Law” and was officially implemented in 1689. The Tontine Law stipulates that each person pays francs and raises a total of 1.4 million francs. After the insurance expires, it is required to pay 10% per year, and the subscribers are divided into groups according to age. For older people, the interest rate is more. The characteristic of the “Tontine Law” is to pay interest to the survivors of the group. If all members of the group die, they will stop paying.

In 1693, the very famous astronomer Edmond Halley compiled the first life table based on the statistics of the citizens' deaths in the city of Breslo, Silesia, which accurately represented the death rate of each age and provided the life insurance calculation. in accordance with. In the 1840s and 1950s, Thomas Simpson made a rate schedule based on Halley's life table. After that, James Dodson calculated the premium according to the age difference and proposed the theory of “balanced insurance premium”, which promoted the development of life insurance. The London Fair Insurance Company, established in 1762, is a life insurance organization that is truly based on the insurance technology foundation.

3. PROBLEM

The main focus and goal for the model is to yield another completely self-sufficient data set with the goal 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.

In order to simulate the insurance evaluation process accurately, we decided to cover 15 of the most important features: BMI, BMI_Age, Ins_Age, Ht, Wt, Product_Info_1-7, Employment_Info_1-6, InsuredInfo_1-6, Insurance_History_1-9, Family_Hist_1-5, Medical_History_1-41, Medical_Keyword_1-48, Product_Info_2_char, Product_Info_2_num

and Medical_Keywords_Count.

BMI is the normalized BMI of applicant.

Ins_Age is the normalized age of applicant.

BMI_Age is the combination of BMI and Ins_Age.

Ht is the normalized height of applicant.

Wt is the normalized weight of applicant.

Product_Info_1-7 is a set of normalized variables relating to the product applied for

Employment_Info_1-6 is a set of normalized variables relating to the employment history of the applicant.

InsuredInfo_1-6 is a set of normalized variables providing information about the applicant.

Insurance_History_1-9 is a set of normalized variables relating to the insurance history of the applicant.

Family_Hist_1-5 is a set of normalized variables relating to the family history of the applicant.

Medical_History_1-41 is a set of normalized variables relating to the medical history of the applicant.

Medical_Keyword_1-48 is a set of dummy variables relating to the presence of/absence of a medical keyword being associated with the application.

Product_Info_2_char is a set of characters in Product_Info_1-7.

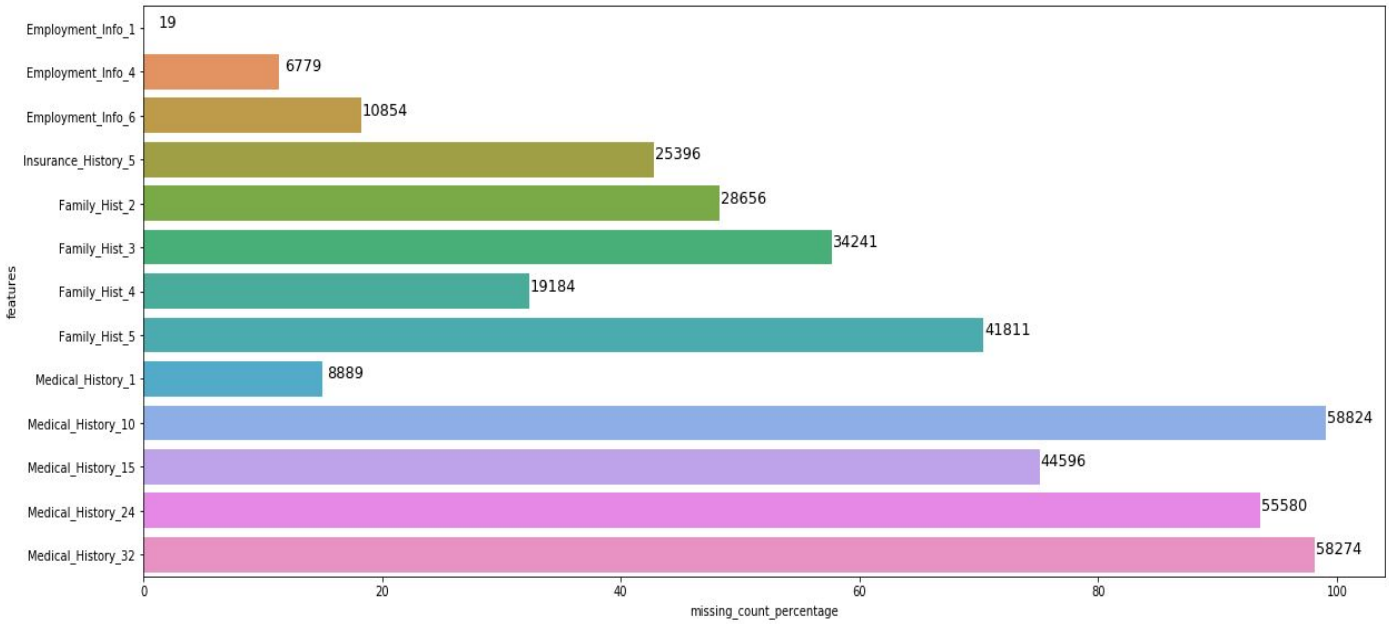
Product_Info_2_num is a set of numbers in Product_Info_1-7.

Medical_Keywords_Count is the sum of figures in Medical_Keyword_1-48.

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.

4. DATA PREPROCESSING

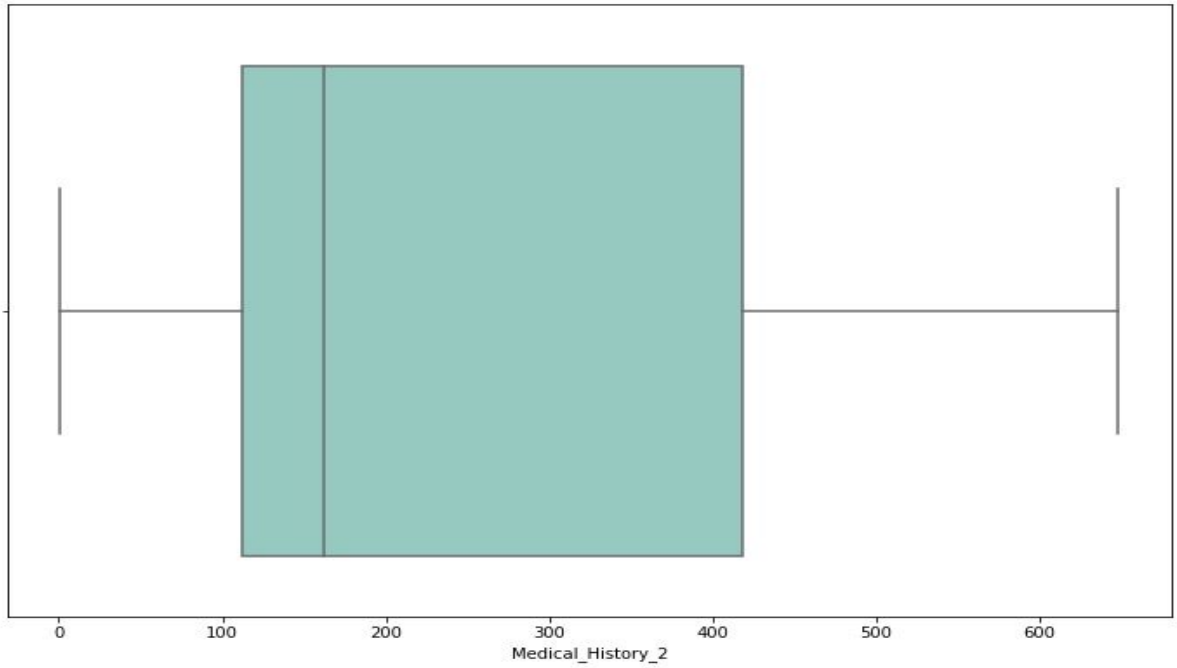
4.1. Date Cleaning

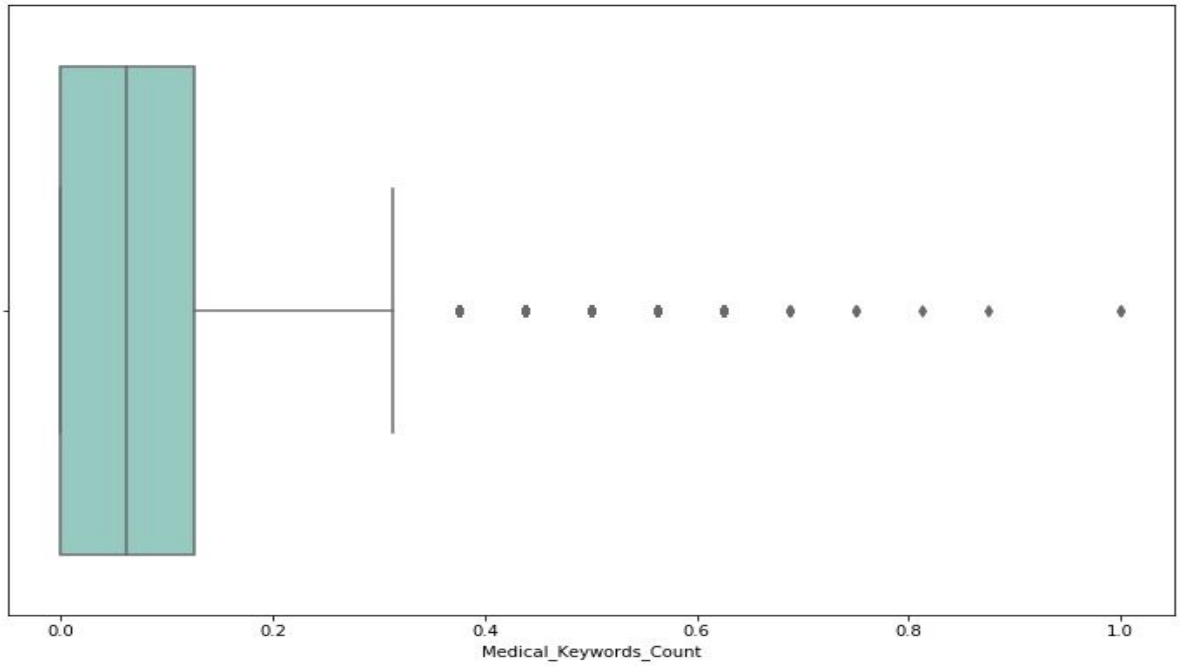


There are two different ways for us to deal with missing value. One is delete the missing value, the other one is using either value to fill the blank. Firstly, we decided to delete features with a missing rate of more than 60%. Therefore, according to the figure above, we deleted following features: Id, Medical_History_32, Medical_History_24, Medical_History_15, Medical_History_10 and Family_Hist_5. 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 a extreme value (such as -10000) to fill the blank due to its feature.

4.2. Scaling

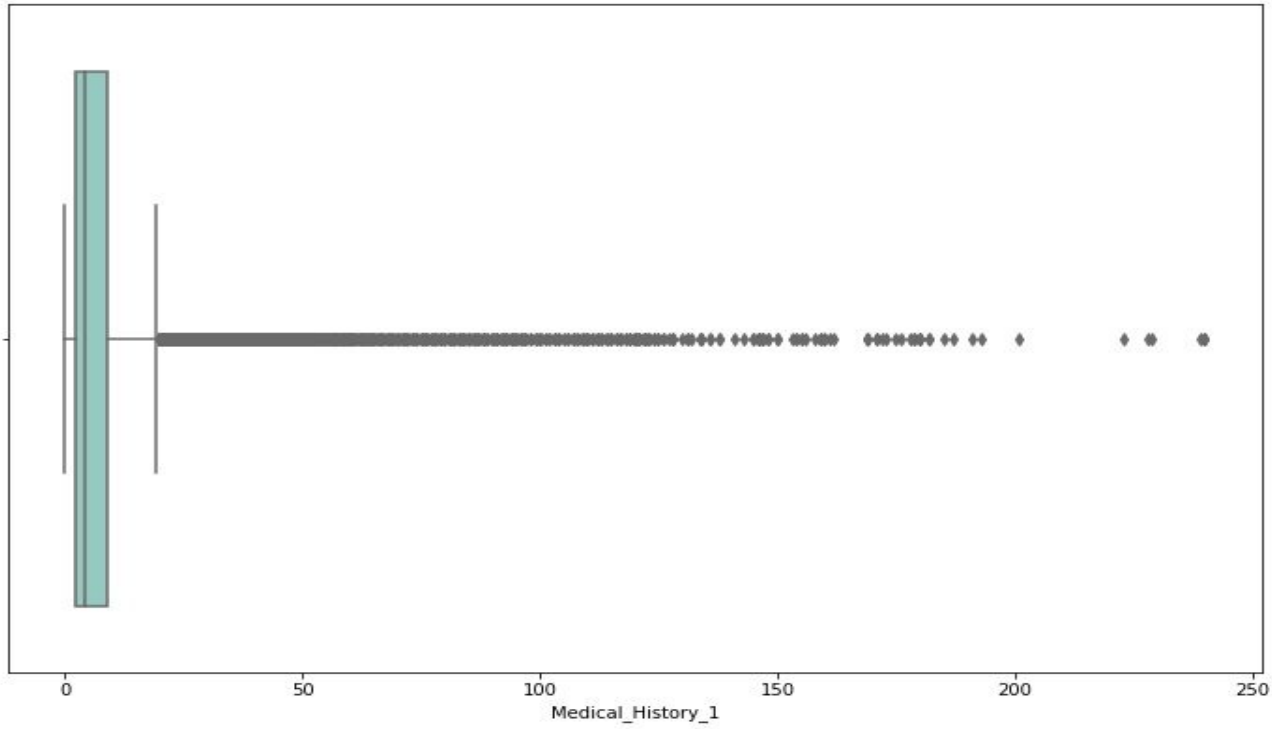
4.2.1. Scaling features to range -- MinMaxScaler





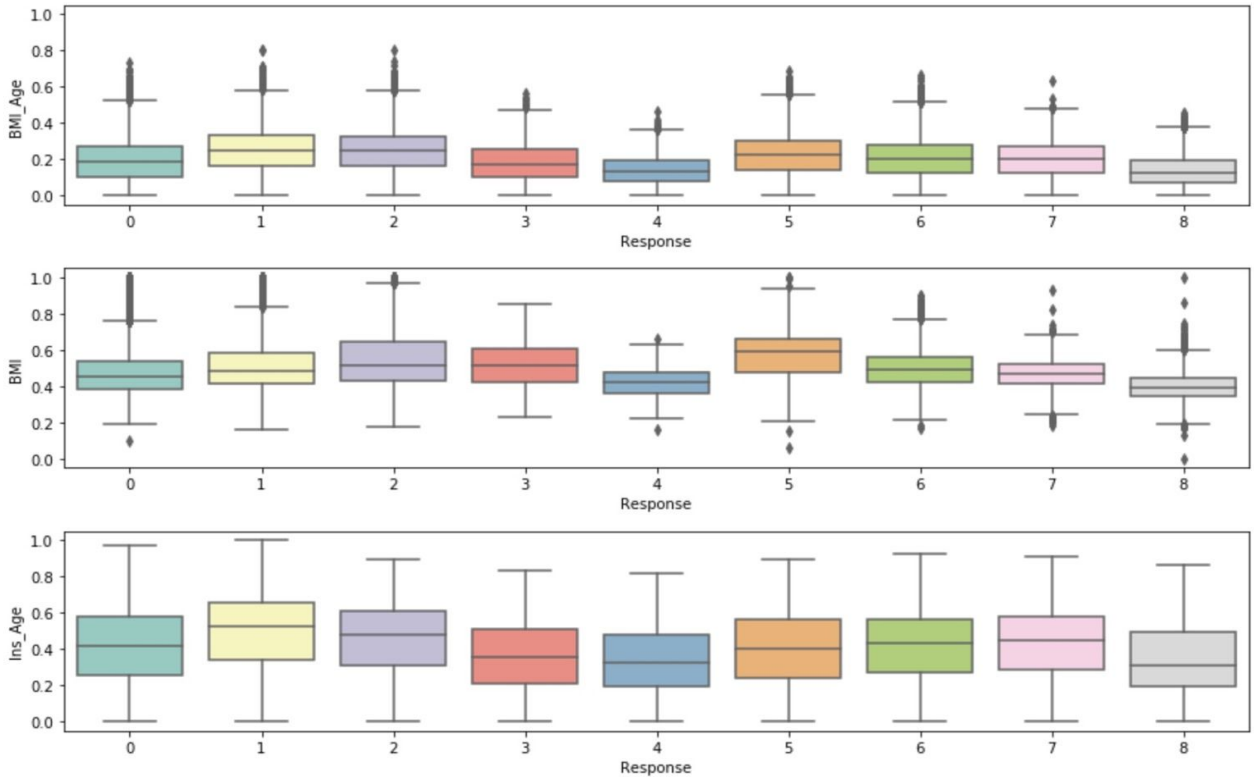
According to the graphs above, it is obvious that the data is concentrated in an integer range. But in terms of the quantity, the amount may be similar to the number of medical treatments. Therefore, it is not a category feature so we use minmaxscaler to scale the features to range. Min-max scaling (many 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. Scikit-Learn provides a transformer called MinMaxScaler for this. It has a feature_range hyperparameter that lets you change the range if you don't want 0–1 for some reason. (Géron, n.d.)

4.2.2. Scaling data with outliers -- RobustScaler



As can be seen from the figure, the Outliers of this set of data are much wider, so it would be a better choice for us to use RobustScaler to make the data more descriptive.

4.3. Feature Creation



4.3.1. Multiplication

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.

4.4. Encoding features

4.4.1. Label Encoder

We will use Label encoder when the type of target features is float to transfer the features into integer.

4.4.2. One Hot

Originally, we intended to use One-Hot to deal with categorical features, but after training the model, we found that the data will become very sparse and the importance of each feature is much lower. Based on this, we thought that One-Hot is not useful and decided to stop using it.

4.5. Feature selection

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. So we try to train with the first 95% of the important features as the next training set.

5. MODEL AND IMPLEMENTATION

5.1. 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 of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases (Numpy.org, 2019).

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 (Pandas.pydata.org,2019).

The reason why we use Matplotlib is because it is a Python 2D plotting library which produces publication quality figures in a variety of hard copy formats and interactive environments across platforms. Matplotlib tries to make easy things easy and hard things possible (Matplotlib.org, 2012).

We selected SciPy because it is a Python-based ecosystem of open-source software for mathematics, science, and engineering (Scipy.org, 2019). SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

And we use Sklearn because it is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Finally, We chose Seaborn as it is a Python data visualization library based on

matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics (Seaborn.pydata.org, 2018).

The functions mentioned above are important for our model.

5.2. Process Overview and Scheduling

First of all, we split the dataset to 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 perfect match, while the negative one means totally 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 learning objective 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 are 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 features, and eliminating scaling of data, and using - 10000 impute all missing values, which lead to higher score. Therefore, XGBoost itself are found out to having a good adaptive ability.

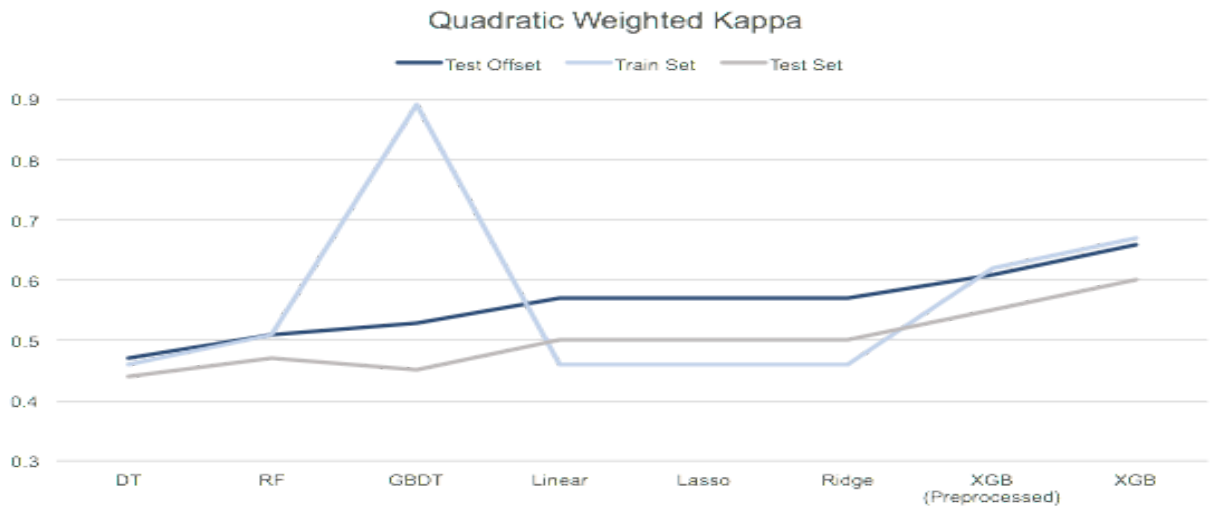
In the model evaluation, we put training set and testing set into the different models to test their kappa score and over-fitting degree.

As can be seen from the graph, the accuracy from decision tree to XGB is increasing. There are three main reasons why XGB has such a good performance:
(1) XGBoost chooses to use L1 or L2 regularization to get 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.



6. RESULTS

when trying to elevate the final assessing score, a special offset method are discovered, referring to the senior's idea. In the process, 8 figures applied to 8 risk groups are estimated to offset some bias between the predicted result and actual result, thus improving 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 train set will be applied into test set to diminish bias, and enhancing scores in kappa.

In summary, using XGBoost model based on offset method, the final kappa score reached 0.658, while the 1st place winner achieve 0.679. Meanwhile top 5 influential features are figure out through built-in function in XGBoost model, all of which are continuous data and most are referring to basic physical condition such as Body Mass Index(BMI), Age and Weight.

7. CONCLUSION

As for Business impact, note that the features' names in the given dataset are unclear, we can't express the model with an explicit formula. However, some business insights can still be given after the research.

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 database, 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 risks. This model can make a difference for the companies' risk management. Also, it can facilitate the pricing process of life insurance policy, the model automatically separate clients to 8 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 recommend according to the customer's economic situation.

8. IMPROVEMENT

The potential improvement can be applied to have a better prediction:
The key issue here is to find the boundaries that maximize the kappa score. In our case, we applied manually designed offset with 8 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 boundaries are changed during a step.
Also, different method can be used to fill the NaN's, and the precision may improve for other models that are based on linear regression. And, it is possible to get the same precision level with fewer features selected, which will also reduce compute duration.

REFERENCE

Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow*. 1st ed. Sebastopol: O'Reilly Media, Inc.

Matplotlib.org. (2012). *Matplotlib: Python plotting — Matplotlib 3.1.1 documentation*. [online] Available at: https://matplotlib.org/ [Accessed 31 Jul. 2019].

Numpy.org. (2019). *NumPy — NumPy*. [online] Available at: https://numpy.org/ [Accessed 31 Jul. 2019].

Pandas.pydata.org. (2019). *Python Data Analysis Library — pandas: Python Data Analysis Library*. [online] Available at: https://pandas.pydata.org/ [Accessed 31 Jul. 2019].

Seaborn.pydata.org. (2018). *seaborn: statistical data visualization — seaborn 0.9.0 documentation*. [online] Available at: https://seaborn.pydata.org/ [Accessed 1 Aug. 2019].

Scipy.org. (2019). *SciPy.org — SciPy.org*. [online] Available at: https://www.scipy.org/ [Accessed 1 Aug. 2019].