Risk Classification of Policy Claim Using Various Boosting Techniques

Erwinna Chendra[a,1], Calista Irene Sugianto[b], and Agus Sukmana[c]
[a,c]*Center for Mathematics and Society, Parahyangan Catholic University, Indonesia*
Email: erwinna@unpar.ac.id, asukmana@unpar.ac.id
[b]*Department of Mathematics, Parahyangan Catholic University, Indonesia*
Email: calistasugianto@gmail.com

Abstract. Evaluating the acceptance of an insurance policy requires the underwriter's accuracy in assessing prospective customers' risk profiles. If it is discovered that a prospective customer has a high risk, the policy application will automatically be rejected. It will take time and substantial costs if someone relies on the traditional underwriting process. Therefore, this research tries to apply the boosting machine learning method in classifying the risk level of life insurance policies using data from Prudential Life Insurance Assessment 2015. There are four boosting techniques used, namely multi-class adaptive boosting (AdaBoost), extreme gradient boosting (XGBoost), light gradient boosting machine (LightGBM), and categorical boosting (CatBoost). XGBoost and LightGBM perform best when training through hyperparameter optimization and experimenting on test data; both have a weighted F1 score of 0.48, with LightGBM performing much faster.

Keywords. Underwriting, Classification, Risk Level, Boosting, Weighted F1.

1. Introduction

Based on the 2023 Insurance Barometer Study conducted by LIMRA and Life Happiness, 39% of potential consumers intend to buy a life insurance policy the following year. Generation Z and millennials dominate this percentage. This enthusiasm is supported by data that the total premium income of the life insurance industry in 2022 will increase by 9.8% compared to 2021. The COVID-19 pandemic in 2020 was one of the main factors in increasing life insurance momentum because this pandemic has claimed many lives from all age groups. If previously policies were more commonly owned by older age groups, now those who are young and healthy are starting to realize the critical role of having a life insurance policy.

The underwriter will analyze the risk profile of each individual who applies for an insurance policy to prevent approval for high-risk individuals. This evaluation process is known as the underwriting process. An underwriter will evaluate the age, health condition, employment, marital status, credit application history, and assets owned by the policy applicant. This data will determine how much premium needs to be paid and the potential policyholder's ability to pay these obligations. Suppose the insurance company sees that the candidate cannot pay off his debts or has a high possibility of claims shortly. In that case, the policy application will be rejected. The historical data is

[1] Corresponding Author: Erwinna Chendra, erwinna@unpar.ac.id.

also used to group customers based on their risk level so that the similarities between customers in one risk group can be seen. However, traditional underwriting procedures are often lengthy and costly. On average, the underwriting process can take four to six weeks, depending on the applicant's risk profile [1]. The longer it takes, the higher the company's expenses; therefore, predictive analysis techniques are introduced.

The insurance industry increasingly turns to predictive analysis to bolster its business efficiency. This powerful tool, successfully applied by Manulife Canada in analyzing HIV patient policies and by the Property and Casualty (P&C) business line in assessing disability claims, is a game-changer in the fight against adverse selection. Predictive analysis techniques, particularly machine learning models, enable the classification of customers according to their risk level, thereby instilling confidence in the insurance industry's robust risk management practices.

In this research, we will discuss the application of boosting machine learning in classifying life insurance policy risks into several levels. Boosting is known to perform well because it builds models iteratively by correcting errors in previous iterations. Boosting techniques developed starting from adaptive boosting (AdaBoost), which is usually applied for classification, then statistical boosting was created, which can be used for regression problems, such as gradient boosting machines (GBM) [2]. AdaBoost performs weight optimization, while GBM attempts to minimize the residual from the prediction results for each iteration. Similar research has proven that various GBM techniques are more accurate than decision trees, random forests, logistic regression, and other machine learning methods. In this research, the multi-class AdaBoost method and several GBM development methods will be used, such as extreme gradient boosting (XGBoost), light gradient boosting machine (LightGBM), and categorical boosting (CatBoost) in classifying life insurance policy risks. We will test the method on various experiments to find optimal conditions that improve model performance. We also will compare the classification abilities of all methods by observing the weighted F_1 score. The research aims to determine the optimal conditions for the dataset to build a risk classification model with various boosting methods. Then, decide whether applying clustering before building the risk classification model provides better performance, as recommended by [1]. Finally, we will compare the performance of boosting methods with optimized parameters based on accuracy, weighted F_1, and execution time.

2. Theoretical Foundations and Literature Review

2.1. Adaptive Boosting (AdaBoost)

The concept of boosting originates from developing a weak learner who performs slightly better than a random learner into a strong learner who performs much better than a random learner. Freund and Schapire [3] first introduced the boosting algorithm to learn the results of iterations and combine them into an accurate classifier. In the case of binary classification, weak learners can classify with an accuracy slightly higher than 50%, while strong learners have an accuracy close to perfect or greater than 99% [2]. The basic concept of boosting is manipulating training data by giving weight to each observation in each iteration; it will give greater weight to data misclassified in an iteration. As a result, the weak learner will produce different results in each iteration, paying particular attention to observations misclassified in the previous iteration. For example, $m =$

$1, \ldots, M$ are the iterations performed, and $\omega^{(m)} = \left(w_1^{(m)}, \ldots, w_n^{(m)}\right)$ are the weights of each n observations in the m-th iteration. In the final stage, the results of the M weak learner predictions are combined into a final prediction and then compared with the actual target. The base learners or basic models often used in boosting are linear functions and decision trees. Some nodes (root node, branch node, leaf node) form a decision tree, where a splitting procedure is carried out for each node except the leaf for each feature (see Figure 1). The best measuring tool for determining split features is the Gini index, entropy, or information gain [4].

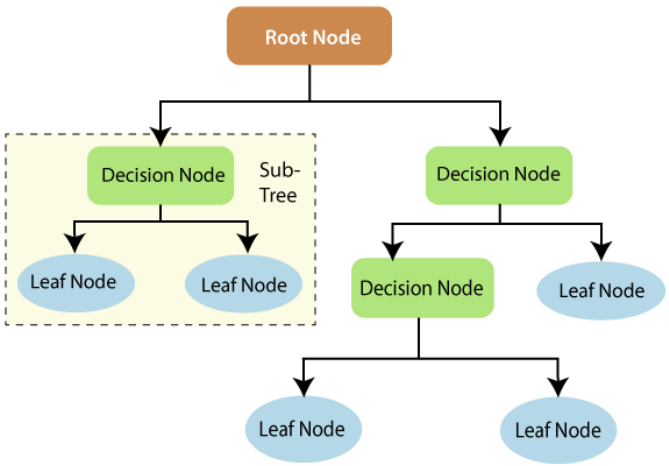

Figure 1. Decision tree illustration
Source: https://miro.medium.com/v2/resize:fit:828/format:webp/1*ekWgr-yVc-ba6DHC_9FeRA.png

AdaBoost is the first practical application of the boosting method, generally used in binary classification problems with simple decision tree weak learners. AdaBoost has adaptive properties, as the name suggests because it automatically adjusts its parameters at each iteration or reweights. By setting the initial weight of each observation equal, a decision tree for classification is built. If there are observations that are misclassified, they will be given greater weight in the next iteration. This iteration continues to repeat until a set limit, so the final decision model is a linear combination of each weak learner multiplied by the amount of say (α). The contribution's magnitude to an iteration's results is represented by the α value [5].

For example, given a dataset $(x_i, y_i)_{i=i}^n$ and the number of weak learner iterations is M. In the case of AdaBoost binary classification, the y_i value is converted to $y_i \in \{-1,1\}$. The final decision model of the AdaBoost method has the following equation:

$$T(x) = sign\left[\sum_{m=1}^{M} \alpha^{(m)} \cdot T^{(m)}(x)\right], \quad (1)$$

where $\alpha^{(m)}$ is the amount of say the mth iteration and $T^{(m)}(x)$ is the prediction result of the m-th weak learner for observation x. The 'sign' notation aims to capture the positive or negative sign of the calculation result. The AdaBoost algorithm uses the exponential loss function as the objective function to be minimized, namely:

$$E^{(m)} = \sum_{i=1}^{n} exp\left[-y_i \cdot S^{(m)}(x_i)\right], \quad (2)$$

where $S^{(m)}(x_i)$ is the number of linear combinations of the amount of say and weak learner until the m-th iteration. After decomposing and deriving equation (2), the amount of say formula can be obtained:

$$\alpha^{(m)} = \frac{1}{2} log \frac{1-err^{(m)}}{err^{(m)}}, \quad (3)$$

where $err^{(m)}$ is the weighted proportion of observations misclassified for an iteration. When dealing with multi-class cases, the Stagewise Additive Modeling algorithm using a Multi-class Exponential loss function (SAMME) is used, with a final decision model [6]:

$$T(x) = argmax_k \sum_{m=1}^{M} \left[\alpha^{(m)} \cdot I\left(T^{(m)}(x) = k\right)\right]. \quad (4)$$

The argmax notation will find the value of k that maximizes the number in square brackets for $k = 1, \ldots, K$.

2.2. Gradient Boosting Machine (GBM)

The application of boosting is not only for classification problems but can predict quantitative outcomes, as introduced by Friedman in statistical enhancement. One elaboration of statistical enhancement is the gradient enhancement machine (GBM), which has been proven to outperform other machine learning models and is classified as working quickly. Almost similar to AdaBoost, GBM improves model performance by providing greater focus on observations that are difficult to predict for each iteration. AdaBoost gives greater weight to misclassified data, while GBM identifies errors through large residuals. GBM will utilize the steepest descent algorithm to determine the best weak learner decision tree that minimizes a specific objective function [2].

Again, consider the dataset $(x_i, y_i)_{i=i}^{n}$ for binary classification $\{0,1\}$ and the number of weak learner iterations is M. Suppose the predicted probability of the i-th observation being in class 1 is f_i. GBM looks for the initial prediction value by minimizing the loss function, which is the sum of the negative log-likelihoods of all observations. The loss function can be expressed in f_i or log(odds)i or O_i, namely:

$$L(y_i, f_i) = -[y_i \cdot \log(f_i) + (1 - y_i) \cdot \log(1 - f_i)], \quad (5)$$

$$L(y_i, O_i) = -\sum_{i=1}^{n} [y_i \cdot O_i] + \sum_{i=1}^{n} \log\left(1 + exp(O_i)\right). \quad (6)$$

By reducing equation (6) to log(odds), we obtain that the initial probability prediction for all observations is the average of y_i. In GBM, the term output value (Ψ) is known for each weak learner iteration leaf node. The output value is obtained by minimizing the objective function:

$$\mathcal{L}^{(m)} = L\left(y_i, f_i^{(m-1)} + \psi_j^{(m)}\right), \quad (7)$$

where $\psi_j^{(m)}$ is the output value of the terminal region $j = 1, \ldots, J_m$ for m iterations. Through second-order Taylor series expansion in equation (7) and then deriving it with respect to ψ, we obtain:

$$\psi = \frac{\sum_{i=1}^{n} e_i^{(m)}}{\sum_{i=1}^{n} \left[f_i^{(m-1)} \cdot \left(1 - f_i^{(m-1)}\right)\right]}, \quad (8)$$

with $e_i^{(m)} = y_i - f_i^{(m-1)}$.

GBM will use the One-vs-All (OVA) approach when facing a multi-class case, as illustrated in Figure 2. For example, assume the response data is a class with three categories: unmarried, married, and divorced. In each iteration, GBM will build three binary classification models, namely (unmarried vs married, divorced), (married vs unmarried, divorced), and (divorced vs unmarried, married). The final class prediction will choose the model with a higher confidence level. For example, model one is 60% confident that the observation is unmarried, model two is 75% confident that the observation is married, and model three is only 50% confident. Based on these

percentages, GBM classifies the observation as married [7]. GBM, discovered in 1999, has evolved into various boosting techniques with better performance. Extreme gradient boosting (XGBoost) developed in 2016 is known for its high performance and regularization capabilities. In 2017, the light gradient boosting machine (LightGBM) was designed with high working speed and is relatively efficient. Then, categorical boosting (CatBoost) was introduced in 2018 to help process categorical features.

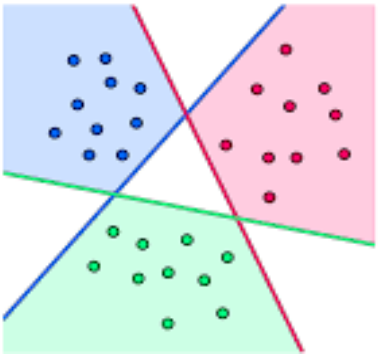

Figure 2. One-vs-all illustration
Source: https://jermwatt.github.io/machine_learning_refined/notes/7_Linear_multiclass_classification/7_2_OvA.html

XGBoost, a GBM technique development, was first introduced by Tianqi Chen and Carlos Guestrin [8]. XGBoost has a higher working speed than traditional GBM due to the improvement of the split-finding algorithm to find the best split. This algorithm consists of the basic exact greedy algorithm, approximate algorithm, and sparsity-aware split finding algorithm to handle missing data automatically [8]. The objective function of XGBoost is similar to equation (7), except that there are additional components:

$$\Omega\left(\psi_j^{(m)}\right) = \gamma \cdot J_m + \frac{1}{2} \cdot \lambda \cdot \left|\left|\psi_j^{(m)}\right|\right|^2, \tag{9}$$

with γ and λ are regularization parameters. As a result, the output value formula has an additional λ component in the denominator. When determining the splitting node, XGBoost looks for features that cause the most significant loss reduction, namely:

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in Rleft} g_i\right)^2}{\lambda+\sum_{i\in Rleft} h_i} + \frac{\left(\sum_{i\in Rright} g_i\right)^2}{\lambda+\sum_{i\in Rright} h_i} - \frac{\left(\sum_{i\in R} g_i\right)^2}{\lambda+\sum_{i\in R} h_i}\right] - \gamma, \tag{10}$$

for g_i and h_i are the loss function's first and second derivatives, respectively.

Microsoft developed LightGBM to obtain models quickly without involving all observations or features. LightGBM uses XGBoost as a baseline by adding the Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) applications [9]. The LightGBM decision tree selects the split feature s at node d based on the most significant variance value after splitting, namely:

$$\tilde{\mathrm{V}}_s(d) = \frac{1}{n}\left[\frac{\left(\sum_{x_i\in A_{left}} g_i + \frac{1-a}{b}\sum_{x_i\in B_{left}} g_i\right)^2}{n_{left}^s(d)} + \frac{\left(\sum_{x_i\in A_{right}} g_i + \frac{1-a}{b}\sum_{x_i\in B_{right}} g_i\right)^2}{n_{right}^s(d)}\right], \tag{11}$$

for A is the set of $(a.100)\%$ observations with large absolute values of residuals and B is the set of $(b.100)\%$ observations with small absolute values of residuals.

Prokhorenkova et al. introduce CatBoost to handle the prediction shift problem by applying the ordering principle technique [10]. CatBoost uses ordered boosting to modify the standard GBM when building a weak learner and uses a new algorithm in the initial processing of categorical features. CatBoost handles categorical features more

effectively by substituting a numeric feature equivalent to the target statistic, called ordered target encoding [10]. When selecting a splitting node in a decision tree, CatBoost looks for the feature that produces the largest cosine similarity value, namely:

$$\text{Cosine similarity} = \frac{\sum_{i\in d}(A_i \cdot B_i)}{\sqrt{\sum_{i\in d} A_i^2} \cdot \sqrt{\sum_{i\in d} B_i^2}}, \tag{12}$$

where A_i is the residual of observation i in the previous iteration and B_i is the output value of the leaf node before observation i was occupied.

In this study, the performance of the boosting model is measured by the weighted F_1 score because it considers the inequality of the response classes. The weighted F_1 formula is:

$$\text{Weighted } F_1 = \sum_{k=1}^{K} w_k \cdot F_{1k}, \tag{13}$$

where K represents the number of response classes, w_k represents the weight of the k-th response class, and F_{1k} is the weighted F_1 score of the k-th class [11].

3. Research Methods

The dataset used in this study is the attribute data of prospective life insurance policyholders published by Prudential Life Insurance Assessment for the Kaggle competition in November 2015. Prudential, as one of the largest insurance companies, sees the crucial role of predictive models in creating a more effective policy evaluation process, so it is expected that those interested in data analytics can participate in processing the dataset. This published data is most likely fictitious because it relates to customers' personal information, but it is still close to the reality in the field. The dataset consists of 59,381 observations and 127 variables (126 features and 1 response). There are 13 continuous features, five discrete features, 60 categorical features, and 48 dummy features. The response is a risk measure consisting of eight different levels.

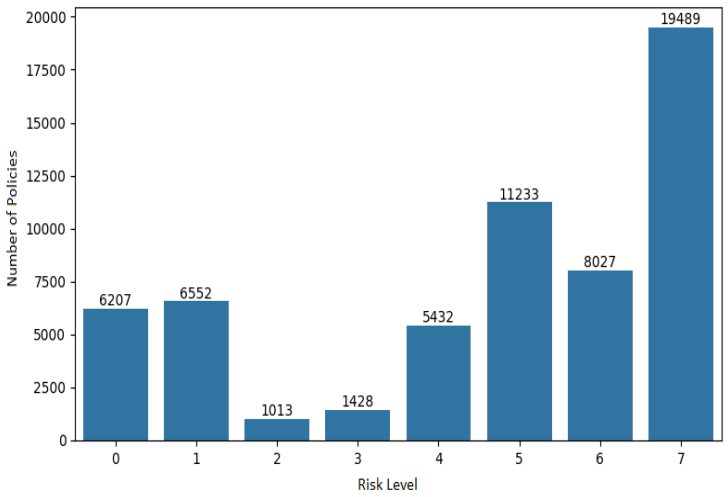

Figure 3. Response with eight risk levels barplot

Figure 3 shows the imbalance in the response level; risk level 7 has the largest number of observations, and risk level 2 has the smallest. The eight response levels are

ordinal, where level 0 means that the prospective policyholder is at low risk, so the higher the level represents, the greater the risk. Most categorical features have two to three different categories; only Medical_History_2 is divided into many categories. Unfortunately, the dataset used does not explain the meaning of the existing categories, so it is left as is. Empty data cannot be processed in machine learning, so it needs to be handled first. The prospective policyholder attribute data has thirteen columns with NaN values. Columns with more than 20% missing data will be deleted, but before that, it is necessary to see whether the nine features are worthy of being removed through a heat map. From the heat map analysis, we decide that the Insurance_History_5 and Family_Hist_4 features were not deleted, or only seven of the thirteen columns were removed. The remaining six columns must be filled in for missing data. Continuous numeric features are filled with mean values, and discrete numeric features are filled with median values [12].

In this study, all numeric features were converted into categorical ones by considering the dimensionality reduction technique to be used later. The conversion was done using the binning method into three categories per feature. Binning is a procedure for dividing the range of each category equally without making the number of observations in each category the same. After conversion, one-hot encoding was performed on the features to avoid ordinary problems and facilitate machine learning methods that can only accept numeric values. After encoding, the total features that were initially 119 became 910 features. The study set the proportion of training and testing data at 80% and 20%. We will conduct model training under five different conditions, and the best conditions were applied to clustering. The boosting method with the best weighted F_1 score during training would be tested on the test data under optimal training conditions. Hyperparameter optimization was also carried out for the boosting method selected for testing. During training, only two hyperparameters were not the default Python parameters, namely n_estimators (n) and learning_rate (η). The entire study was conducted on Google Collaboratory, a cloud-based platform based on Python.

4. Results

The performance of the four boosting methods during training is seen in five different conditions. This variation is done because there is an alleged problem of imbalance in response levels and dimensionality problems. Overcoming the imbalance is attempted through random sampling and combining risk levels 2, 3, and 4. Meanwhile, the dimensionality problem is overcome through dimensionality reduction with chi-squared feature selection. These five conditions consist of the following: the first is without handling imbalanced data or feature selection. The second condition is the application of random sampling without feature selection. The third condition uses six risk levels without feature selection. The fourth condition uses eight risk levels plus a chi-squared feature selection. The fifth condition uses six risk levels plus a chi-squared feature selection. During training, each method is run on three combinations of n and η, and each applied 5-fold cross-validation.

Table 1. Comparison of boosting performance for five training conditions

Condition	n and η	Mean *weighted* F_1 (SD *weighted* F_1)			
		Multi-class AdaBoost	XGBoost	LightGBM	CatBoost
8 Level Model	n=150;η=0.05	0.2694(0.0037)	0.4515(0.0036)	0.4619(0.0045)	0.4342(0.0044)
	n=100;η=0.1	0.2679(0.0032)	0.4553(0.0036)	0.4608(0.0041)	0.4387(0.0050)
	n=50;η=0.15	0.2707(0.0048)	0.4520(0.0032)	0.4591(0.0039)	0.4322(0.0032)
8 Level Model + *Random Sampling*	n=150;η=0.05	0.2561(0.0025)	0.4528(0.0034)	0.4608(0.0037)	0.4400(0.0042)
	n=100;η=0.1	0.2566(0.0020)	0.4550(0.0054)	0.4608(0.0056)	0.4448(0.0031)
	n=50;η=0.15	0.2564(0.0030)	0.4518(0.0047)	0.4562(0.0027)	0.4397(0.0039)
6 Level Model	n=150;η=0.05	0.2767(0.0033)	0.4643(0.0018)	0.4713(0.0016)	0.4418(0.0022)
	n=100;η=0.1	0.2776.0.0031)	0.4666(0.0029)	0.4721(0.0024)	0.4488(0.0031)
	n=50;η=0.15	0.2766(0.0033)	0.4638(0.0024)	0.4710(0.0032)	0.4414(0.0018)
8 Level Model + Feature Selection	n=150;η=0.05	0.2694(0.0037)	0.4520(0.0034)	0.4627(0.0051)	0.4363(0.0049)
	n=100;η=0.1	0.2679(0.0032)	0.4558(0.0031)	0.4625(0.0055)	0.4426(0.0044)
	n=50;η=0.15	0.2707(0.0048)	0.4524(0.0023)	0.4612(0.0037)	0.4356(0.0034)
6 Level Model + Feature Selection	n=150;η=0.05	0.2767(0.0033)	0.4644(0.0028)	0.4711(0.0022)	0.4445(0.0024)
	n=100;η=0.1	0.2776(0.0031)	0.4671(0.0031)	0.4705(0.0023)	0.4517(0.0041)
	n=50;η=0.15	0.2766(0.0033)	0.4635(0.0026)	0.4707(0.0031)	0.4446(0.0019)

Based on the observation of the weighted F_1 and the program running time for each condition, we conclude that overcoming the imbalance by combining several minority risk levels was able to slightly improve $(+0.01)$ the multi-class classification ability of AdaBoost, CatBoost, XGBoost, and LightGBM. Random sampling is not an excellent solution to overcome the imbalance in the dataset handled because it does not improve performance and slows down the results. Furthermore, selecting features based on the chi-squared statistical test can significantly reduce the model construction time and extract important features. The model with additional selection did not experience much change in the weighted F_1 score. Therefore, the experimental design chosen for clustering was the fifth condition, combining several minority response levels and applying chi-squared feature selection. The selected features were 297 out of 910, with the top three being BMI_3, Wt_2, and Medical_Keyword_15. LightGBM performed the best in various condition modifications; the execution time required was also the

minimum. The second method occupies the XGBoost position, whose score deviation often overlaps with LightGBM—followed by CatBoost in the third position and multi-class AdaBoost in the fourth position. CatBoost requires the longest execution time compared to the other three boosting methods, most likely due to the ordering principle algorithm. Figure 4 visually compares the weighted F_1 scores under optimal conditions, and Figure 5 enlarges Figure 4 without including the results of the AdaBoost method.

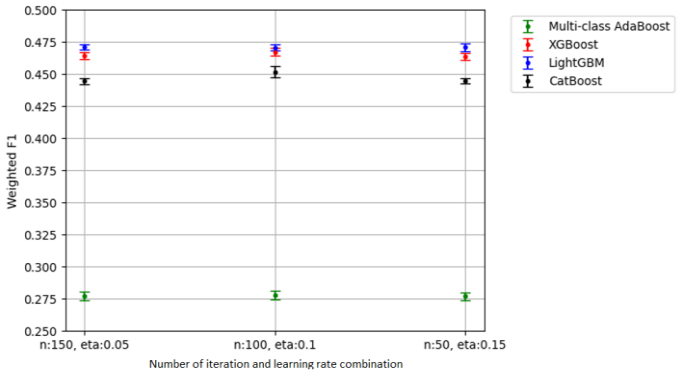

Figure 4. Comparison of weighted F_1 scores with six levels and feature selection

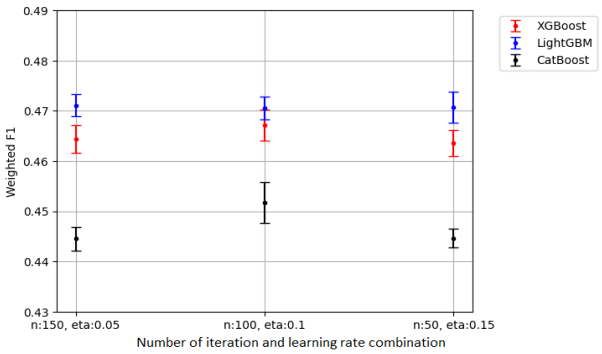

Figure 5. Comparison of weighted F_1 scores with six levels and feature selection without AdaBoost

According to the suggestion to use clustering techniques [1], k-mode clustering was carried out before applying the boosting technique to the data of six risk levels consisting of 297 categorical features. Clustering aims to collect observations with similar feature characteristics into one group, so the model can learn more optimally. According to the elbow method and silhouette score results, only two clusters are needed to group all observations. Through the k-mode algorithm, which is repeated five times with random center selection, the distribution of response levels in the two clusters is quite balanced. The weighted F_1 plot in Figure 6 shows the difference in performance of the boosting method when using some and all of the data. Notes 1, 2, and 3 indicate the combination of the number of iterations and the learning rate. Note 1 for $n = 150$ and $\eta = 0.05$, note 2 for $n = 100$ and $\eta = 0.1$, while note 3 represents $n = 50$ and $\eta = 0.15$. Clustering does not seem to help improve the classification results of the XGBoost, LightGBM, and CatBoost methods because if the scores of clusters one and two are averaged, it will give worse results than using the entire data. There is a slight

improvement in the multi-class AdaBoost, where the average score of the two clusters exceeds the results without clustering. For the Prudential Life Insurance Assessment dataset, the main goal of the performance of the two clusters being above the whole has not been achieved.

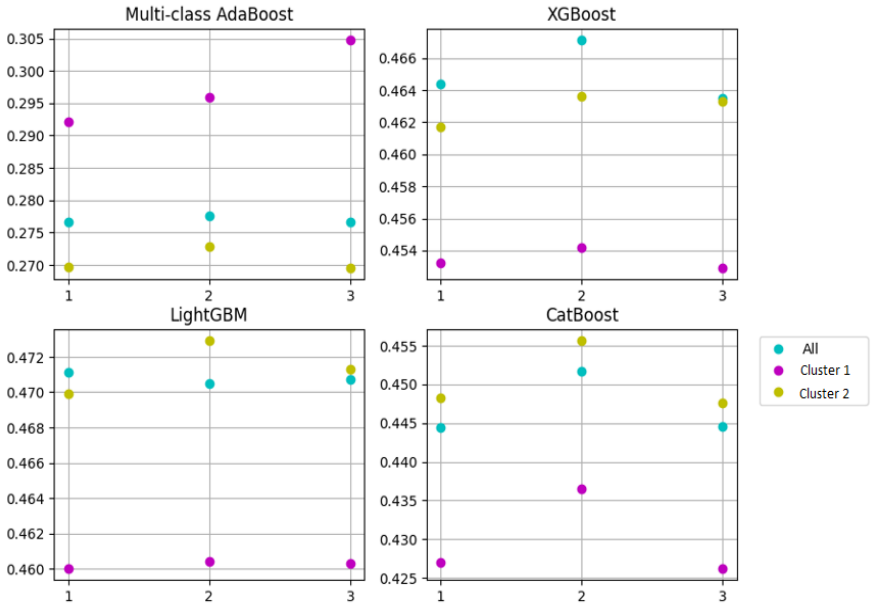

Figure 6. Comparison of boosting performance before and after clustering

In the previous training series, the parameters outside the number of iterations and learning rate still use the default values, so we carry out the optimal hyperparameter search for the XGBoost and LightGBM methods. CatBoost was not studied further because it requires a very long execution time. At the same time, multi-class AdaBoost does have the opportunity to improve the model, but adding depth gradually also increases program time, so it is not an effective method. Table 2 shows the results of hyperparameter optimization of XGBoost and LightGBM. Through 50 optimization iterations with each applied 5-fold cross-validation, the best model for the LightGBM method produces a weighted F_1 score of 0.4730. Meanwhile, the optimal XGBoost model for the risk level classification problem has a weighted F_1 score of 0.4736.

Table 2. List of optimization result hyperparameters

Hyperparameter	LightGBM	XGBoost
learning_rate	0.1067315	0.1125662
n_estimators	171	183
max_depth	13	7
max_leaves	20	200

min_child_samples	75	-
colsample_bytree	0.6618760	0.25
reg_alpha	1	0.5296959
reg_lambda	0.4503880	0.7928206
min_split_gain/gamma	0.5	0
top_rate	0.5227312	-
other_rate	0.4	-

The best model estimator was applied on the test data that had been separated from the beginning. Table 3 shows the most essential features in building the model based on the amount of gain. The features that are in the top five ranks for both XGBoost and LightGBM are Medical_Keyword_15, Wt_1, and BMI_3. This means that these features can divide the response levels well when performing the splitting procedure in building the model. The results on the test data show that XGBoost has an accuracy level of 0.51, while LightGBM is slightly below it at 0.50. Based on the weighted F_1, both boosting methods have similar scores of 0.48. If observations are explicitly made per response level, the highest F_1 score of 0.71 is at risk level 5 for both methods. Other risk levels are in the range of 0.30 to 0.40. This shows that XGBoost and LightGBM cannot detect low to medium-risk levels correctly but can detect high-risk policies well. When choosing only one best-boosting technique for a multi-class classification problem with response inequality and all categorical features like the data in this study, LightGBM is the best option. The algorithm's improvement when selecting features and samples in each iteration produces an effective model.

Table 3. Feature importance according to total gain

XGBoost	***Importance***	**LightGBM**	***Importance***
Medical_Keyword_15	13101.1318	*Wt_1*	24379.367
Wt_1	12934.2646	*Medical_History_4_1*	21080.798
Medical_History_4_2	12292.2998	*Medical_Keyword_15*	17302.895
BMI_3	11791.0117	*BMI_3*	17080.575
Medical_History_23_2	9446.1611	*Wt_2*	15043.828

5. Conclusions

The prospective life insurance policyholder dataset needs help with response level inequality and dimensionality. According to the training data experiment results, the

disparity is better overcome by combining risk levels 2, 3, and 4 to reduce the number of response levels to six. The size of the dimension can be reduced using the chi-squared statistical test. Of the 910 categorical features, only 297 were extracted at a significance level 0.05. Clustering trials on conditions of six response levels and selected features showed no performance improvement. Both the elbow method and the silhouette score suggest grouping into two clusters, but there has been no increase in the weighted F_1 score.

Of the five training conditions in the experimental design, the XGBoost and LightGBM methods have the best classification ability based on weighted F_1 scores that often overlap each other. CatBoost followed it in third place, but its execution time is the longest. AdaBoost has the lowest score, which is far from the other three methods. Hyperparameter search for XGBoost and LightGBM resulted in the best model estimates, with the most essential features such as Medical_Keyword_15, Wt_1, and BMI_3. XGBoost has an accuracy of 0.51, and LightGBM has an accuracy of 0.50, while the weighted F_1 scores of both methods are similar at 0.48. Specific observations at each response level show that the highest risk level tends to be more accurately predicted than other levels. This study will help users interested in the policy risk classification process. The results of the boosting method can provide affirmation if a policy is indeed high risk and must be rejected. Likewise, the use of all categorical features can help during the process of identifying prospective customers if numerical data is challenging to obtain.

References

[1] Boodhun N, Jayabalan M. Risk prediction in life insurance industry using supervised learning algorithms. Complex & Intelligent Systems. 2018;4(2):145-154.

[2] Mayr A, Binder H, Gefeller O, Schmid M. The evolution of boosting algorithms. Methods of Information in Medicine. 2014;53(06):419-427.

[3] Freund Y, Schapire RE. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences. 1997 Aug; 55(1):119–139.

[4] Shalev-Shwartz S, Ben-David S. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.

[5] Schapire RE. The boosting approach to machine learning: An overview. Nonlinear estimation and classification. 2003:149-171.

[6] Hastie T, Rosset S, Zhu J, Zou H. Multi-class adaboost. Statistics and Its Interface. 2009;2(3):349-360.

[7] Galar M, Fernández A, Barrenechea E, Bustince H, Herrera F. An overview of ensemble methods for binary clas-sifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. Pattern Recognition. 2011;44(8):1761-1776.

[8] Chen T, Guestrin C. Xgboost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKKD International Conference on Knowledge Discovery and Data Mining; 2016 Aug 13-17; San Francisco, CA, USA:785-794.

[9] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, ..., Liu TY. Lightgbm: A highly efficient gradient boosting decision tree. The 31st Conference on Neural Information Processing Systems (NIPS); 2017 Des 4-9; Long Beach, California, USA:1-11.

[10] Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. CatBoost: unbiased boosting with categorical features. The 32nd Conference on Neural Information Processing Systems (NIPS); 2018 Des 3-8; Montreal, Canada:1-11.

[11] Grandini M, Bagli E, Visani G. Metrics for multi-class classification: an overview. arXiv preprint. 2020; arXiv:2008.05756.

[12] Sessa J, Syed D. Techniques to deal with missing data. In 2016 5th International Conference on Electronic Devices, Systems, and Applications (ICEDSA); 2016 Des 6-8; Ras Al Khaimah, United Arab Emirates: IEEE;1-4.