Classification of Life Insurance Policy Claims Risk Using Various Boosting Techniques

Calista Irene Sugianto[1*]

[1*] Mathematics, Faculty of Information Technology and Science, Parahyangan Catholic University

Keywords:
Underwriting, Risk Level, Boosting, Classification, Weighted F1, Hyperparameter

Email penulis:
Calistasugianto@gmail.com*

Abstract

The process of evaluating the acceptance of an insurance policy requires accuracy on the part of the underwriter in assessing the risk profile of prospective customers. If it is discovered that a prospective customer has a high risk, the policy application will automatically be rejected. If someone relies on the traditional underwriting process, it will take time and substantial costs. Therefore, this study tried 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 were 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 had the best performance when training, through hyperparameter optimisation and then experimenting on test data, both had a weighted F1 score of 0.48, with LightGBM performing much faster.

Introduction

Based on the 2023 Insurance Barometer Study conducted by LIMRA dan Life Happiness, approximately 39% of prospective customers interested in purchasing life insurance policies in the following year.[2] The number was dominated by Generation Z and millennials. This enthusiasm was supported by data that showed the total premium income of life insurance industries in 2022 rose by 9.8% compared to 2021. Covid-19 emergence back in 2020 was one of the main factors behind the growth of life insurance momentum, as the pandemic claimed many lives across all age groups within a short period of time.[3] If previously only older people purchased policies, now younger and healthier generation realise the importance of protecting themselves from unpredictable risks.

Inside insurance companies, underwriters will analyse the risk profile of each individual in order to prevent fraud application, or known as underwriting process. An underwriter evaluates the age, health condition, occupation, marital status, credit history, and assets of the policy applicant. The collected data will determine whether the applicant is feasible enough for approval and the amount of premium to be paid. If insurers notice that the prospective policyholder has big probability of late payment or is likely to make a claim frequently, the application will be rejected. However, traditional underwriting procedures are often lengthy and costly. On average, formal underwriting flow can take four up to six weeks, depending on how complicated the data obtained (Boodhun & Jayabalan, 2018). The longer the time consumed, the bigger the company's expenses, therefore predictive analysis techniques might become a solution.

Life insurance industry has begun to implement predictive analysis inside operational schemes. For instance, Manulife Canada analyses HIV patient policies using this technique, and it has also been applied for 20 years by the Property and Casualty (P&C) business line for disability claim

[2] https://www.iii.org/fact-statistic/facts-statistics-life-insurance
[3] https://www.bdo.com/insights/industries/insurance/how-covid-19-changed-the-life-insurance-industry

scoring based on recovery periods. Another use case is that predictive analysis can model mortality rate to optimise underwriting efficiency and to prevent adverse selection, a condition where an evaluator does not have enough information and ends up issuing policies to high-risk customers (Boodhun & Jayabalan, 2018). Adverse selection can be avoided by classifying prospective customers according to their risk level using one predictive method, namely machine learning.

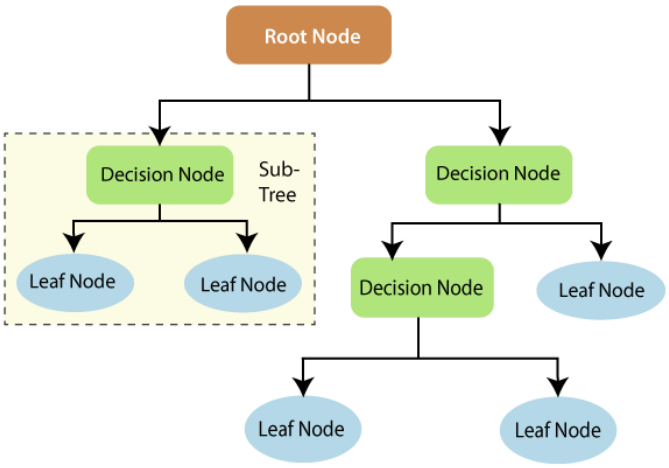
This study discussed the application of boosting in classifying life insurance policy risks into several different levels. Boosting is famous to perform relatively well because of its nature of building model iteratively, correcting the mistakes of previous models. Development of boosting techniques began with adaptive boosting (AdaBoost), which is usually used for classification purposes, then there was statistical boosting for regression purposes, such as the gradient boosting machine (GBM) (Mayr et al., 2014). AdaBoost does weight optimisation, while GBM attempts to minimise the residuals from prediction results in each iteration. Similar studies have proven that boosting has better accuracy compared to sole decision tree, random forest, logistic regression, and other machine learning methods. This study used multi-class AdaBoost and three GBM branch techniques: extreme gradient boosting (XGBoost), light gradient boosting machine (LightGBM), and categorical boosting (CatBoost) for classifying life insurance policy risks. Those four methods were tested in a series of different experiments to find the most well-performed model. The evaluation metric to rate the classification ability of each model was weighted F1. There were three main goals of this study, the first one was to determine the optimal conditions of the dataset (Prudential Life Insurance Assessment 2015) when building boosting classification models. Then, determined whether clustering before training models provided a better performance, as recommended by Boodhun and Jayabalan. And finally, compared the performance of all boosting methods whose parameters had been optimised based on accuracy, weighted F1, and execution time.

Theoretical Foundation and Literature Review

Adaptive Boosting (AdaBoost)

Boosting concept rooted from the idea of developing a weak learner that works slightly better than a random predictor into a strong learner that performs much better than a random variable. Boosting algorithm was first introduced by Schapire dan Freund, with the aim of combining iteration results into one accurate classifier. In the case of binary classification, weak learners can classify with accuracy slightly above 50%, while strong learners have almost perfect accuracy, above 99% (Mayr et al., 2014). The basic concept of boosting is to manipulate training data by weighting each observation in every iteration; higher weight is given to misclassified observations. Consequently, weak learners will give distinct results in each iteration, focusing more on data points that are classified incorrectly. For example, $m = 1, \ldots, M$ are the iterations performed and $\omega^{(m)} = \left(w_1^{(m)}, \ldots, w_n^{(m)}\right)$ are the weight of each of the n observations in iteration m. In final stage, prediction results of M weak learners are combined into one final prediction and then compared with the actual target. The type of commonly used base learners in boosting are linear function and decision tree. A tree consists of nodes (root node, branch node, leaf node), where each node undergoes splitting process, except the leaf node. The metrics for deciding best split can be Gini index, entropy, and information gain (Shalev-Shwartz & Ben-David, 2014).

Figure 1. Illustration of a Decision Tree



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

AdaBoost is the first practical application of boosting methods, commonly used for binary classification with simple decision trees as weak learners. As the name suggests, AdaBoost has adaptive upbring because it automatically adjusts its parameters in each iteration and performs reweighting. By setting equal initial weight for all observations, a decision tree is constructed continuously for classification. If a data point is misclassified, it will obtain a higher weight in upcoming iteration. This step continues until a specified number of iterations is reached, then the final model will be a linear combination of each weak learner result multiplied by amount of say (α). How much contribution one iteration give is represented by the α value (Schapire, 2003).

For instance, given a *dataset* $(x_i, y_i)_{i=i}^{n}$ and number of iterations M. When dealing with AdaBoost binary classification, y_i value is converted into $y_i \in \{-1,1\}$. AdaBoost final model will have the following equation:

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

with $\alpha^{(m)}$ represents amount of say and $T^{(m)}(x)$ as weak learner prediction result for observation x. 'Sign' notation aims to capture the positive or negative sign of the calculation. AdaBoost algorithm uses exponential loss function as an objective function which going to be minimised:

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

where $S^{(m)}(x_i)$ is the sum of linear combination amount of say and weak learner up to iteration m. After expanding and simplifying equation (2), the formula of amount of say can be obtained:

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

where $err^{(m)}$ is the proportion of misclassified observations for an iteration. When dealing with multi-classification job, algorithm Stagewise Additive Modeling using a Multi-class Exponential loss function (SAMME) takes place, with final decision model be (Hastie et al., 2009):

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

Notation argmax will find k value that maximise the sum inside the square bracket, for $k = 1, \ldots, K$.

Gradient Boosting Machine (GBM)

Boosting implementation is not limited to classification problems but can be extended to predict quantitative outputs, as introduced by Friedman in statistical boosting. One elaboration of statistical boosting is the gradient boosting machine (GBM) which has been proven to outperform other machine learning models and works relatively fast. Similar to AdaBoost, GBM also improves model performance by placing greater emphasis on observations that are difficult to predict for each iteration. AdaBoost gives greater weight to misclassified data, while GBM identifies mistakes through large residuals. GBM uses the steepest descent algorithm to decide the best weak learner decision tree that minimises the objective function (Mayr et al., 2014). Given a dataset $(\mathrm{x_i}, \mathrm{y_i})_{\mathrm{i=i}}^{\mathrm{n}}$ for binary classification {0,1} and the number of weak learner iterations is M. Suppose the probability of the i-th observation being in class 1 is $\mathrm{f_i}$. GBM searches for the initial prediction value by minimising a loss function, which is the sum of the negative log-likelihood of all observations. The loss function can be expressed as $\mathrm{f_i}$ or log(odds)i or $\mathrm{O_i}$, namely:

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

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

By calculating the derivative of equation (6) upon log(odds), the initial prediction for all observation will be the average of $\mathrm{y_i}$. In GBM, there is a term called output value (ψ) for each leaf node in an iteration of a weak learner. Output value can be calculated by minimising the following objective function:

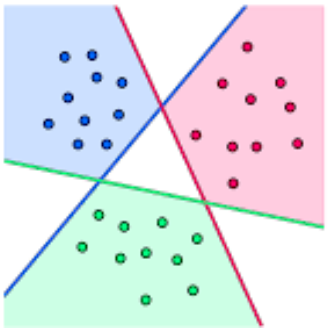
$$\mathcal{L}^{(m)} = L\left(y_i, f_i^{(m-1)} + \psi_j^{(m)}\right), \qquad (7)$$

where $\psi_j^{(m)}$ is the *output value of terminal* $\mathrm{j = 1, \ldots, J_m}$ at m-th iteration. Through second order taylor-series expansion on equation (7), then differentiate it with respect to ψ, obtained:

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

with $\mathrm{e_i^{(m)} = y_i - f_i^{(m-1)}}$. When dealing with multi-class problems, GBM applies the One-vs-All (OVA) approach as illustrated in Figure 2. As an example, assume the response variable has three classes: unmarried, married, and divorced. In each iteration, GBM builds three binary classification models, namely (unmarried vs married, divorced), (married vs unmarried, divorced), and (divorced vs unmarried, married). The final class prediction will be based on the model with higher confidence level. For instance, the first model is 60% confident that the observation is unmarried, the second model is 75% sure that the observation is already married, and the third model is only 50% confident. Then, it can be concluded that the observation is most likely to be married (Galar et al., 2011). GBM, founded 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, light gradient boosting machine (LightGBM) was designed to be extremely fast and efficient. Then, categorical boosting (CatBoost) was introduced in 2018, specifically to assist categorical feature processing.

Figure 2. One-vs-All Scheme Illustration



Source:
https://jermwatt.github.io/machine_learning_refined/notes/7_Linear_multiclass_classification/7_2_OvA.html

XGBoost, an extension of GBM was proudly introduced by Tianqi Chen dan Carlos Guestrin. It has relatively higher processing speed compared to traditional GBM due to enhancement in the split finding algorithm for finding the best split. This algorithm consists of basic exact greedy algorithm, approximate algorithm, and sparsity-aware split finding algorithm to automatically handle missing data (Chen & Guestrin, 2016). XGBoost's objective function is similar to equation (7), except there is an additional component:

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

with γ and λ as regularization parameters. As a result, there is an additional λ in the denominator of the output value formula. When deciding the splitting node, XGBoost searches for a feature that gives the biggest loss reduction:

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

where g_i and h_i represent the first and second derivative of the loss function.

LightGBM was developed by Microsoft in order to obtain models quickly without having to involve all observations nor features. LightGBM takes XGBoost as its baseline, then adds Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) applications (Ke et al., 2017). LightGBM's decision tree selects splitting feature s at node d based on the largest varance value after splitting:

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

where A is the set of (a.100)% observations with large absolute residual values and B is the set of (b.100)% observations with small absolute residual values.

CatBoost was introduced by Prokhorenkova et al. to address prediction shift problems by applying the ordering principle technique. CatBoost uses ordered boosting as a modification of standard GBM when building weak learners and uses a new algorithm in the initial processing of categorical features. CatBoost handles categorical features more effectively by substituting a numerical feature equivalent to the target statistic, known as ordered target encoding (Prokhorenkova et al., 2018). When selecting splitting nodes in decision trees, CatBoost selects feature that produces greatest cosine similarity value:

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

A_i is the residue of i-th observation in the previous iteration and B is the output value of the leaf node before i-th observation takes place.

Weighted F1

In this study, model performance is measured by weighted F1 score, considering the imbalance of response variable. Weighted F1 formula is:

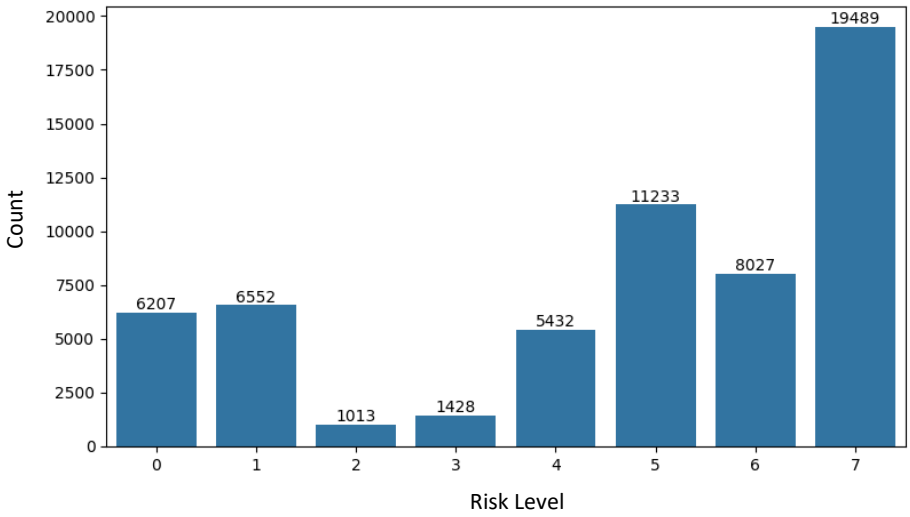
$$Weighted\ F_1 = \sum_{k=1}^{K} w_k \cdot F_{1k},$$

where K denotes the number of response classes, w_k as the weight of k-th class, and F_{1k} denotes weighted F1 score of k-th class (Grandini et al., 2020).

Research Method

The dataset used in this study was attribute data on prospective life insurance policyholders published by Prudential Life Insurance Assessment for the Kaggle competition in November 2015.[4] Prudential, one of the biggest insurance companies in Indonesia, realised how crucial predictive model roles in creating an effective policy evaluation process. Therefore, they invite people interested in data analytics to participate. The data is most likely dummy due to personal data protection rules, but it still portrays the reality of the insurance field. The dataset consists of 59,381 observations and 127 variables (126 features and 1 response). In total, there are 13 continuous features, 5 discrete features, 60 categorical features, and 48 dummy. The response or targeted variable has eight different risk levels.

Figure 3. Barplot of Rensponse Variable with 8 Levels



From Figure 3, there is a significant imbalance in response proportions with risk level 7 having the most observations and level 2 having the fewest. Those eight levels are ordinal, where 0 means the prospective customers has low risk and 7 means extremely high risk. Most categorical features consist of two to three different categories, only Medical_History_2 has many classes. Unfortunately, there is no explanation about the difference between each class in one categorical feature. Missing data were handled before moving on to the training phase and there were thirteen

[4] https://www.kaggle.com/competitions/prudential-life-insurance-assessment/data

columns with NaN identified. Columns with more than 20% missing data were considered unreliable and suggested to be taken out. However, before deleting nine out of thirteen columns with more than 20% missing rows, it was necessary to analyse the heat map. Based on the analysis, Insurance_History_5 and Family_Hist_4 features were kept, making the seven remaining features deleted from the dataset. To fill in missing cells, this study used imputation techniques: continuous numerical features were filled with average value and discrete numerical features were filled with the median value (Sessa & Syed, 2016).

In this study, all numerical features were converted into categorical features in consideration of the dimension reduction technique used. The conversion was performed utilizing binning into three categories per feature. Binning is a procedure of dividing the range of each category equally without making the number of observations in each category the same. After conversion, all features underwent one-hot encoding to avoid ordinality issues and to facilitate machine learning methods that can only accept numeric value. After encoding, the number of features increased from 119 to 910. The proportion of training and testing data for this study were set at 80% and 20%. Model training was conducted in five different conditions and the optimal condition underwent clustering. The boosting method with the best weighted F1 score was tested on 20% test data. There was hyperparameter optimisation in boosting methods chosen for the testing phase. During training, only two hyperparameter were not Python's default settings, namely n_estimators (n) and learning_rate (η). In the appendix part, there is a flowchart showing the overall flow of this study. All coding was done in Google Collaboratory as a Python-based Cloud platform.

Table1. List of Instances and Library

No.	Description	Instances	Library
1	One-hot encoding	OneHotEncoder	sklearn.preprocessing
2	Feature selection	chi2	sklearn.feature_selection
3	Random oversampling	RandomOverSampler	imblearn.over_sampling
4	Random undersampling	RandomUnderSampler	imblearn.under_sampling
5	Clustering	KModes	kmodes.kmodes
6	Fold division	RepeatedStratifiedKFold	sklearn.model_selection
7	Fold evaluation	cross_val_score	sklearn.model_selection
8	AdaBoost method	AdaBoostClassifier	sklearn.ensemble
9	XGBoost method	XGBClassifier	xgboost
10	LightGBM method	LGBMClassifier	lightgbm
11	CatBoost method	CatBoostClassifier	catboost
12	Hyperparameter tuning	BayesSearchCV	skopt

Result and Discussion

The performance of four boosting methods during training was observed under five distinct conditions. These variations were conducted to resolve response level imbalance and

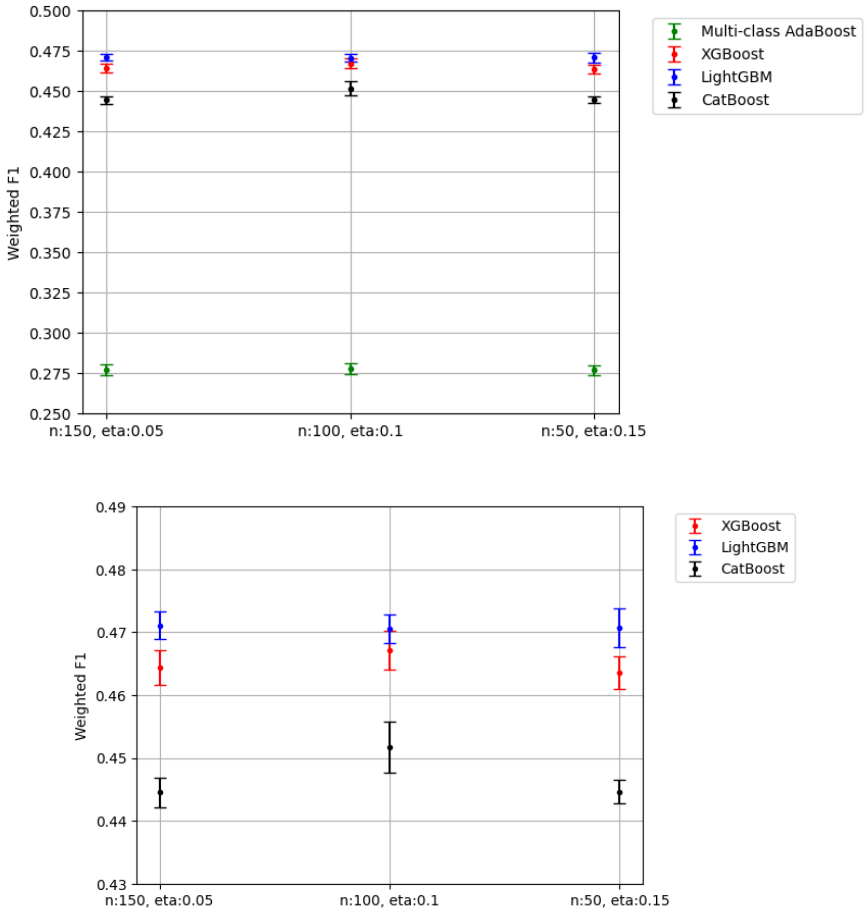
dimensionality issues. Imbalance was addressed through random sampling and combining risk level 2, 3, and 4. Meanwhile, dimensionality issue was addressed through chi-squared feature selection. The five conditions during training consisted of: eight risk levels without modification, application of random sampling without feature selection, six risk levels without feature selection, eight risk levels plus chi-squared feature selection, and lastly six risk levels with chi-squared feature selection. All conditions went run on three combinations of n and η, with each applied with 5-fold cross-validation,

Table 2. Comparison of Boosting Performance on Five Training Condition

Condition	n and η	Mean weighted F1 (SD weighted F1)			
		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 weighted F1 score observations and program running time in each condition, it was found that addressing imbalance issues with combining some minority risk levels improved multi-classification performance (+0,01) in AdaBoost, CatBoost, XGBoost, dan LightGBM. Random sampling was not an optimal solution due to its incapability to perform fast and effectively. Furthermore, implementing chi-squared feature selection significantly reduced running time and extracted important features, although weighted F1 score only slightly increased. As a result, the fifth condition of combining several response levels and applying feature selection was chosen for clustering trial. A total of 297 features were selected from the previous 910, with the BMI_3, Wt_2, and Medical_Keyword_15 having the highest importance. LightGBM performed best in all five conditions and required the least time to run. The second-best method was XGBoost, whose score deviations often overlapped with LightGBM. Followed by CatBoost and multi-class AdaBoost in third and fourth positions. CatBoost occupied the longest execution time, most likely because of the existence of an ordering principle algorithm. Figure 4 illustrates a visual comparison of the weighted F1 score under optimal conditions.

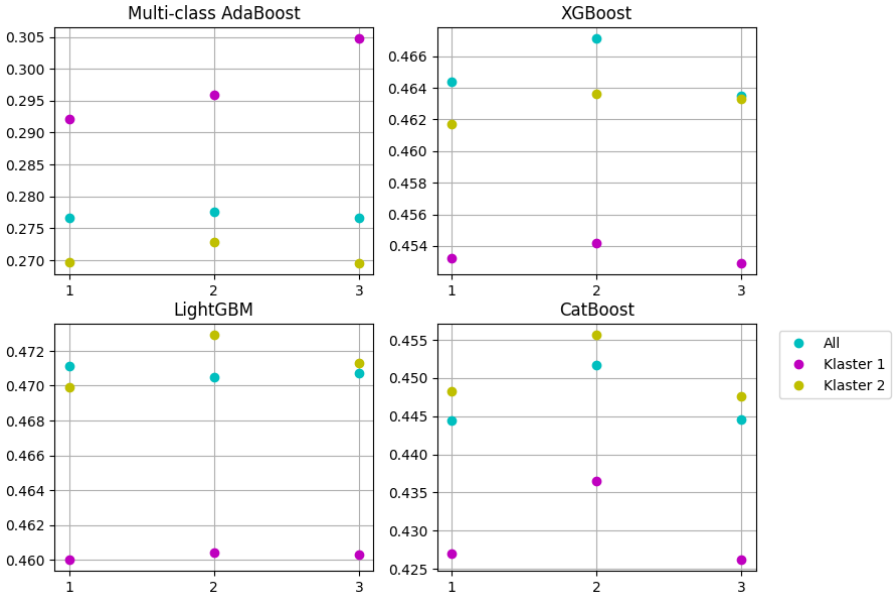
Figure 4. Comparison of Weighted F1 Scores with Six Response Levels and Feature Selection



In accordance with the recommendation (Boodhun & Jayabalan, 2018) to use clustering technique, this study implemented k-modes clustering in the optimal condition: six risk levels and 297 categorical features. Clustering aims to group observations with similar feature characteristics so the model can learn easily from each cluster. Based on elbow method and silhouette score, only two clusters were required in this study. Through the k-modes algorithm, which was repeated five times with random centre selection, the distribution of response levels in the two clusters was fairly

balanced. Figure 5 shows the differences between weighted F1 score when boosting applied to all observations, first cluster, and second cluster. Row value of 1, 2, dan 3 denotes number of iterations and learning rate combinations. Note: 1 represents n=150 and η=0.05, 2 represents n=100 and η=0.1, while 3 represents n=50 and η=0.15. Clustering did not succeed to improve classification results in XGBoost, LightGBM, and CatBoost methods because the average score of the two clusters was lower than using all data. There was a slight improvement in multi-class AdaBoost, where the average of two clusters exceeded overall. It was decided that for the Prudential Life Insurance Assessment dataset, clustering was unable to enhance model performance.

Figure 5. Comparison of Boosting Performance Before and After Clustering



In the previous training, parameters other than number of iterations and learning rate used default values, so the next step was implementing hyperparameter optimisation for XGBoost and LightGBM. AdaBoost and CatBoost were not further investigated due to its significant low score and long running time respectively. Table 3 provided the result of hyperparameter optimisation from XGBoost dan LightGBM. Through 50 repetitions and each applied with 5-fold cross validation, the best model using LightGBM method produces a weighted F1 score of 0.4730. The best XGBoost model for risk level classification produced a weighted F1 score of 0.4736.

Table 3. List of Optimised 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 was tested on the remaining 20% data that was separated earlier before training. Table 4 showed five most important features when building a final classification model based on the total gain. Both XGBoost and LightGBM had Medical_Keyword_15, Wt_1, and BMI_3 in the top five list. This means that those features divided the response levels well. Testing results on 20% data showed that XGBoost had 0.51 accuracy, while LightGBM was slightly below, at 0.50. According to the weighted F1 score, both models had the same score of 0.48. If analysis were made specifically per level, the highest F1 score was at risk level 5 for two models. The other risk levels obtained a weighted F1 score between 0.30 and 0.40. This indicated that XGBoost and LightGBM were not yet able to correctly classify low to medium risk levels, but ablee to detect high-risk policies well. If this study must choose only one boosting method for multi-classification problems specifically for this data, LightGBM appeared to be the best option.

Table 4. Feature Importance

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

Conclusion and Implications

This study investigated attributes of prospective life insurance policyholders and tried to classify the risk response by boosting methods. The dataset has imbalance and dimensionality issues. According to the training results, the imbalance problem was better addressed by combining risk levels 2, 3, and 4 so the number of levels was reduced to six. Then, the problem of dimensionality could be addressed with a chi-squared statistical test. Based on the feature selection, out of 910 categorical features, only 297 features were considered significant at a significance level of 0.05. Clustering suggestion from a past paper was applied to the best condition combination. Both the elbow method and silhouette score advised grouping the data into two clusters, however there was no improvement in the weighted F1 score.

From four boosting methods, XGBoost and LightGBM had the best multi-classification capabilities based on weighted F1 scores, which often overlapped with each other. Catboost followed in third place but the remaining time was extremely long. AdaBoost had the lowest score, far behind the other three models. Hyperparameter search for XGBoost and LightGBM produced the best model estimation, with the most important features being Medical_Keywords_15, Wt_1,

and BMI_3. When tested into the holdout set, both models had the similar weighted F1 score of 0.48. Specific analysis at each response level showed that highest risk levels tended to be more accurately predicted than other lower levels. Based on the results, hopefully this study can assist users in the policy risk classification process to decide whether one application should be approved or rejected.

References

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

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

Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding Machine Learning: From Theory to Algorithms. Cambridge university press.

Schapire, R. E. (2003). The boosting approach to machine learning: An overview. Nonlinear estimation and classification, 149-171.

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

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

Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKKD International Conference on Knowledge Discovery and Data Mining (pp. 785-794).

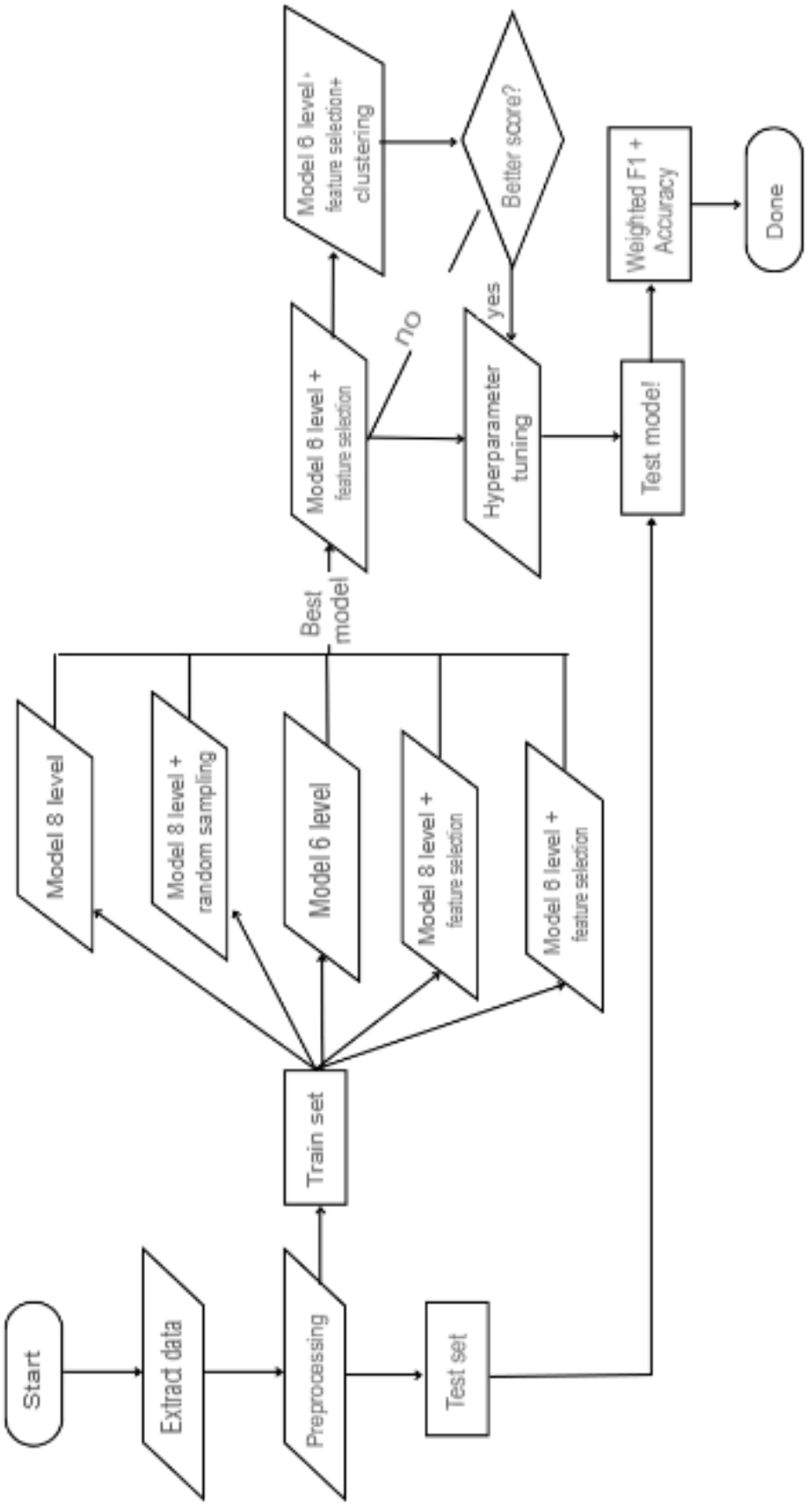
Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. 31st Conference on Neural Information Processing Systems (NIPS) (pp. 1-11).

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). CatBoost: unbiased boosting with categorical features. 32nd Conference on Neural Information Processing Systems (NIPS) (pp. 1-11).

Grandini, M., Bagli, E., & Visani, G. (2020). Metrics for multi-class classification: an overview. arXiv preprint arXiv:2008.05756.

Sessa, J., & Syed, D. (2016, December). Techniques to deal with missing data. In 2016 5th International Conference on Electronic Devices, Systems, and Applications (ICEDSA) (pp. 1-4). IEEE.

Appendix


Start
Extract data
Preprocessing
Test set
Train set
Model 8 level
Model 8 level + random sampling
Model 6 level
Model 8 level + feature selection
Model 6 level + feature selection
Best model
Model 6 level + feature selection
no
Model 6 level + feature selection+ clustering
Better score?
yes
Hyperparameter tuning
Test model
Weighted F1 + Accuracy
Done