2019 Cathay Big Data Competition Analysis Document

I. Project Introduction

In the insurance industry, customer behavior and decision-making processes for purchasing policies are complex and unpredictable. For sales professionals, precise marketing, maintaining good customer relationships, and understanding customers' insurance needs are crucial. In this project, we aim to utilize customers' historical data to predict whether existing customers will purchase critical illness insurance policies within a specific timeframe, thus creating a predictive purchasing model. This model will help identify customers with higher insurance needs and provide sales professionals with a method to target potential customers with such needs, aiming to reduce overall costs and maximize profits.

II. EDA, Exploratory Data Analysis

2.1 Dataset Description

The official dataset provided includes two types: Train and Test, as shown in Table 1. In both datasets, the "CUS ID" field serves as the index value (index), and therefore it is not included in the column calculations. However, in the Test dataset, the field "Y1" is missing because it is the target field that this project aims to predict.

Table 1

	資料筆數	欄位數
Train 資料集	100000筆	131
Test 資料集	150000筆	130

2.2 The Y1 field

From the cumulative plot of the Y1 field, it is evident that the distribution of whether customers have purchased critical illness insurance is highly imbalanced. The Y-value represents customers who have purchased the insurance, totaling 2000 data points, while the N-value represents customers who have not purchased the insurance, totaling 98,000 data points, as shown in Figure 1.

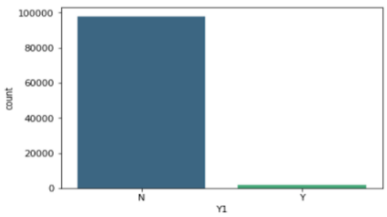
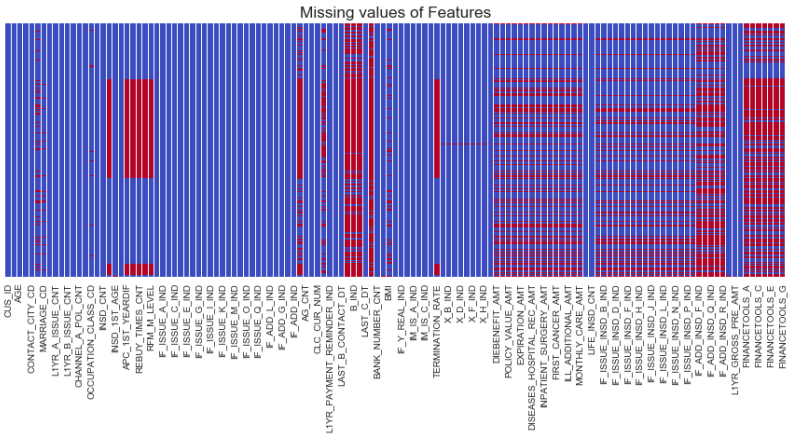


Figure 1: Distribution of N and Y values in the Y1 field

2.3 Missing Values

From the missing value patterns, the missing value patterns in the Train and Test datasets were analyzed using a missing value heatmap, as shown in Figure 2. The X-axis represents the columns, while the Y-axis represents customer IDs. The data cells with values are represented in blue, while the data cells without values are represented in red.

Figure 2: Missing Value Heatmap



Based on the heatmap analysis, the following observations can be made regarding the missing values in the Train dataset: (1) There are a total of 73 columns with missing values. (2) Nearly half of all columns have missing values in almost half of the 5,783 samples. (3) Among the 131 features, 17 columns have missing values that account for more than half of the values in those columns. The Test dataset also has 73 columns with missing values, and the columns with missing values in the Test dataset are the same as those in the Train dataset. Additionally, it can be observed that the missing values in the heatmap exhibit parallel distribution in certain columns. This means that for certain customers, they either have missing values in specific columns or no missing values in those same columns simultaneously.

III. Data Cleaning & Data Processing

3.1 Variable Transformation

To integrate the observation of the value range across the entire dataset and align with the selected machine learning model's requirements, the following rules have been planned for transforming categorical variables into numerical variables:

(1) "Low", "Medium", "Medium-High", "High" will be transformed into 1, 2, 3, 4 respectively.

(2) "Low", "Medium", "High" will be transformed into 1, 2, 3 respectively.

(3) "N" and "Y" will be transformed into 0 and 1 respectively (including the label).

(4) "F" and "M" will be transformed into 0 and 1 respectively.

(5) "A1" will be transformed into 1, "A2" into 2, "B1" into 3, "B2" into 4, "C1" into 5, "C2" into 6, "D" into 7, "E" into 8.

(6) "A" will be transformed into 1, "B" into 2, "C" into 3, "D" into 4, "E" into 5, "F" into 6, "G" into 7, "H" into 8.

At this stage, the variable "BMI value" will be discretized into whether it falls within the range [0, 0.2) or not, represented by 1 for "yes" and 0 for "no". By referring to the kernel density plot (refer to Figure 3), we can observe the distribution of the number of samples that purchase critical illness insurance and those that do not purchase it across different BMI values. From the plot, it is evident that when the BMI value is within the range of 0 to 0.2, the number of samples not purchasing critical illness insurance is significantly higher than the number of samples purchasing it. On the other hand, when the BMI value is not within the range of 0 to 0.2, the number of samples not purchasing critical illness insurance is noticeably lower than or approximately equal to the number of samples purchasing it. Therefore, it can be inferred that when a customer's BMI value falls within the range of 0 to 0.2, it is highly likely that they will not purchase critical illness insurance. Based on this observation, it is concluded that whether the BMI falls within the [0, 0.2) range is an important feature in explaining Y1 (the purchase of critical illness insurance), and thus the BMI value will be discretized accordingly.

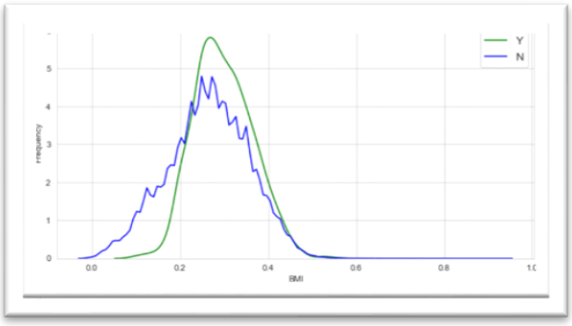


Figure 3: Kernel Density Plots of BMI for Y1 as N or Y

Finally, a new column was added to appropriately describe the characteristic of the total annual coverage amount for customers. The new column is called "TOTAL AMT - Total Annual Coverage," and it is calculated by summing up the relevant columns related to the annual coverage amount (a total of 15 columns). After creating the new column, the original 15 columns are deleted.

3.2 Missing Value Imputation
3.2.1 Missing Value Imputation Methods
A. Domain-specific Imputation

B. Regression Imputation (two kinds of methods)
Method 1:
Use the columns in the dataset that do not have missing values as the independent variables and the column to be imputed as the dependent variable. Use the feature_importances_ function in the ExtraTreesRegressor model to calculate the importance scores of each independent variable for the dependent variable. Sort the variables in descending order and select the top 10 important variables as the explanatory variables. Divide these 10 variables into 10 groups: the first important variable, the first + second important variables, the first + second + third important variables, and so on. Use these 10 feature combinations as the explanatory variables and the column to be imputed as the dependent variable. Perform calculations using the ExtraTreesRegressor model, and for each feature combination, measure the accuracy using a specific metric. If the column to be imputed is a categorical variable, use the F1 score as the accuracy metric. If it is a numerical variable, use MSE and RMSE as the accuracy metrics. Finally, select the feature combination with the best accuracy metric and use the ExtraTreesRegressor model to predict and impute the missing values in the column to be imputed.

If the F1 score is below 0.7 for each feature combination or if the MSE and RMSE are not ideal, discard this method.

Method 2:
This method is a more cautious approach to reinforce the first approach. If the accuracy metric results from the first approach are not satisfactory, for the sake of rigor, the second approach will be considered for imputing missing values. First, the columns without missing values are used as independent variables, and the column to be imputed is set as the dependent variable. The ExtraTreesClassifier model is employed with 10 different seeds, and the model is run 10 times. Each time, the feature_importances_ function is used to calculate the importance scores of each independent variable. The top 10 important variables (independent variables) are listed for each run. The frequency of occurrence of all the independent variables (columns without missing values) in the top 10 important variables across the 10 runs is then calculated. Finally, the variables (independent variables) with the highest frequencies are selected, and the ExtraTreesClassifier model is used to predict and impute the missing values in the column to be imputed.

C. Filling with Mode and Mean
Fill the missing values of the target column by using either the mode or the mean of that column.

3.2.2 Execution Result
According to the exploratory data analysis (EDA) in sections 2-3, the missing values in the columns "APC_1ST_AGE" (Age of the first policyholder), "REBUY_TIMES_CNT" (Number of repeat purchases), "RFM_M_LEVEL" (Number of previous policies owned), "RFM_R" (Time interval since the last policyholder's identity was insured), and "LEVEL" (Relationship level) (refer to Table 2) all indicate that these samples have never served as policyholders. However, the meaning of these missing values does not belong to any specific category in these five columns. Therefore, these missing values will be replaced with a new category: 0.

REBUY_TIMES_CNT (再購次數)	資料筆數
1	27778
2	12267
3	6255
4	10418
NaN	43282

APC_1ST_AGE (首次擔任要保人年齡)	資料筆數
NaN	3620
3	1388
1	791
4	774
2	541

LEVEL (往來關係等級)	資料筆數
NaN	43305
5	28415
1	13940
4	6490
3	4041
2	3809

RFM_M_LEVEL (曾投保主約件數)	資料筆數
3	20799
5	11832
7	7296
8	7753
9	4865
10	4173
NaN	43282

RFM_R (上次要保人身份投保距今間隔時間)	資料筆數
1	17506
2	12291
3	14616
4	12293
NaN	43294

Table 2

In the exploratory data analysis (EDA) of sections 2-3, the missing values in the column "APC_1ST_YEARDIF" (Time interval since becoming the first policyholder) represent cases where the individual has never served as a policyholder. The meaning of these missing values does not correspond to any numerical value in the column. However, since "APC_1ST_YEARDIF" is a numeric variable and cannot have missing values, the column will be discretized and transformed into a nominal scale (refer to Table 3). The missing values will be replaced with a new category.

Table 3

APC_1ST_YEARDIF (首次成為要保人距今間隔時間)	
原本型態	離散化型態
NaN	0
[0,25]	1
(25,50]	2
(50,75]	3
(75,100]	4

Table 4

TERMINATION_RATE (曾解約保單張數佔曾投保保單張數佔率)	
count	56718
mean	12.09
std	27.66
min	0
25%	0
50%	0
75%	0
max	100

Continuing from the exploratory data analysis (EDA) in sections 2-3, the missing values in the column "ATION_RATE" (Cancellation rate as a percentage of the total number of policies) represent cases where the individual has never served as a policyholder. In this state, the missing values do not correspond to any specific value in the column. Since "ATION_RATE" is a numeric variable and cannot have missing values, the column will be discretized, and the missing values will be replaced with a new category. From the descriptive statistics in Table 4, it is noted that most values are close to 0. Therefore, the column will be transformed into two categories: "Is 0" and "Not 0". The missing values will be treated as a separate category.

The column "OCCUPATION_CLASS_CD" (Occupation class of customers) is

filled using regression method one. Firstly, 15 important explanatory variables are selected, and all possible combinations of these variables are generated (15 combinations in total). Each combination is used as the set of explanatory variables to predict the "OCCUPATION_CLASS_CD" values. The F1 scores for these predictions range from approximately 0.69 to 0.71. Finally, one combination is chosen, and the following fields are used as explanatory variables to predict and fill the missing values: CONTACT_CITY_CD, TOOL_VISIT_1YEAR_CNT, LIFE_INSD_CNT, CUST_9_SEGMENTS_CD, AGE, L1YR_GROSS_PRE_AMT, CHARGE_CITY_CD, CHANNEL_A_POL_CNT, AG_CNT, and AG_NOW_CNT.

The column "ANNUAL_INCOME_AMT" (Annual income) is filled using a combination of scheme one and scheme two in the regression method. Firstly, in scheme one, the top 25 important fields are selected, and all possible combinations of these fields (25 combinations in total) are generated. The mean squared error (MSE) as the accuracy metric for each combination ranges mostly between 1.9 and 2.3, making it difficult to determine clear superiority. To further refine the approach, scheme two is employed with 50 different seeds for 50 model iterations. Among these iterations, the fields "L1YR_GROSS_PRE_AMT" (Gross premium paid in the past year), "AG_CNT" (Number of agents handled in the past), "AGE" (Age), "LIFE_INSD_CNT" (Number of valid life insurance contracts), and "CUST_9_SEGMENTS_CD" (Nine major customer segments) appear in the top 10 features in approximately 50 to 49 iterations. Therefore, these five fields are used as explanatory variables to predict and fill the missing values.

The column "BMI value," following the variable transformation discussed in section 3.1, is discretized into two categories based on whether it falls within the range [0, 0.2). A value of 1 is assigned to indicate that the BMI value is within this range, while a value of 0 represents that the BMI value is outside this range. The missing values are classified as "No" in this category.

For the "Current annual coverage amount-related fields" (total of 15 columns) in the dataset, regression method scheme two is employed to fill in the missing values. Firstly, 10 different seeds are set, and 10 iterations are performed. Among the iterations, the following nine columns: "Age," "IF_Y_REAL_IND" (whether to purchase Y insurance), "TOOL_VISIT_1YEAR_CNT" (number of visits by salesperson using management tools in the past year),

"IF_2ND_GEN_IND" (whether the policyholder is a second-generation), "IF_S_REAL_IND" (whether to purchase S insurance), "IF_ADD_Q_IND" (current valid category of Q life insurance policy), "APC_CNT" (number of insured persons corresponding to the policy), "LIFE_INSD_CNT" (current effective number of insured contracts), and "CONTACT_CITY_CD" (contact address - city), all appear 10 times among the top 10 features. Therefore, these nine columns are chosen as explanatory variables to predict and fill in the missing values.

The missing values in the "Current valid category of life insurance policy_AR (primary contract)" (a total of 17 columns) can be partially inferred from the "LIFE_INSD_CNT" column. If the value in that column is 0, it indicates that the sample did not have a primary contract in that year. Therefore, it is expected that all 17 columns in the "Current valid category of life insurance policy_AR (primary contract)" should be 0 (no). However, when examining the values of these 17 columns corresponding to a "LIFE_INSD_CNT" value of 0, it can be observed that almost all missing values occur when "LIFE_INSD_CNT" is 0. Therefore, the missing values in these 17 columns will be filled with 0 (no).

From the Figure 4, it can be noticed that when the "OCCUPATION CLASS CD" (Customer Occupation Class Category) is 1, the number of females is significantly higher than males. Therefore, it can be inferred that when a sample has a value of 1 in the "OCCUPATION CLASS CD" column, it is likely to be a female. To fill in the missing values in the "Gender" column, we can map them to the "OCCUPATION CLASS CD" column. If the missing value in the "Gender" column corresponds to a value of 1 in the "OCCUPATION CLASS CD" column, the missing value in the "Gender" column is filled with 1 (female). Otherwise, it is filled with 0 (male).

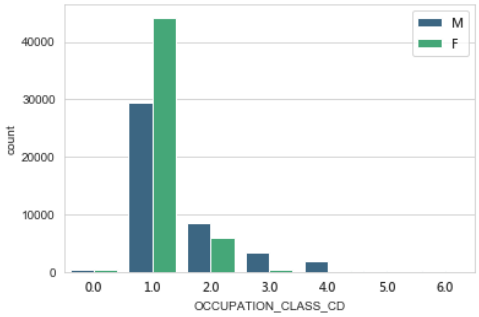


Figure 4: Distribution of Male (M) and Female (F) Counts for the OCCUPATION CLASS CD Field.

From the "L11YR C CNT (Number of Service Applications via Channel C in the Last 11 Years)" field, it can be inferred that if there is no service application via Channel C in the last three years, it means that there was no service application via Channel C in the last year. The missing values in the "L11YR C CNT" field can be associated with the "LAST C DT (Date of Last Service Application via Channel C)" field. If a sample has missing values in the "L11YR C CNT" field and the corresponding value in the "LAST C DT" field is 0, the missing value should be 0 as well. Therefore, it will be filled with 0.

For other columns with missing values, categorical columns and numerical columns will be filled with the mode and mean, respectively. Finally, for the remaining columns with a small number of missing values, the samples with missing values in those columns will be deleted directly.

IV. Model Construction

4.1 Further Processing

Multiple-category nominal variables such as CHARGE_CITY_CD, CONTACT_CITY_CD, MARRIAGE_CD, CUST_9_SEGMENTS_CD, APC_1ST_AGE, REBUY_TIMES_CNT, RFM_M_LEVEL, APC_1ST_YEARDIF, TERMINATION_RATE, RFM_R, and LEVEL will be processed using One-Hot Encoding. If there are only two categories in a nominal variable, they have already been converted to the 0-1 category during the data cleaning process.

4.2 Model Selection

Due to the time constraints and hardware limitations during the competition, our team found it challenging to perform extensive hyperparameter tuning for multiple models. Additionally, considering the limited number of submissions allowed, we have chosen to adopt a strategy of selecting the best-performing models. Therefore, we will focus on enhancing and testing only the two most promising models for submission on the competition website.

Model	AUC
Extreme Gradient Boosting (XGBoost)	0.8156
AdaBoostClassifier	0.8050
Random Forest	0.7979
Logistic Regression	0.7970
MLP	0.6635
Decision Tree	0.5264

Table 5

We ran the cleaned data through six different models: XGBoost, AdaBoost, Random Forest, DecisionTree, Logistic Regression, and MLP. The goal was to select the two models with the highest AUC for further enhancement and testing. These models were run with default parameters without any tuning. From Table 5, it can be seen that XGBoost and AdaBoost achieved the top two AUC scores. However, since the top three models are all tree-based models with similar underlying principles, we aimed to choose one tree-based model and one non-tree-based model to diversify the subsequent testing and increase the potential for score improvement. Additionally, the Logistic Regression model ranked fourth, and its AUC score was not significantly lower. Furthermore, this model has room for optimization, such as addressing the impact of imbalanced data by using techniques like SMOTE sampling. Additionally, tree-based models inherently have a tendency to select important features during computation, while Logistic Regression does not. This can be improved by selecting features based on their importance using techniques like Feature Importance. Therefore, in the end, we chose the XGBoost model with the highest AUC and the non-tree-based Logistic Regression model for further enhancement and testing.

4.3 Modeling

By using the train_test_split function, the Train dataset was split into 67% training data (Training Data) and 33% validation data (Validation Data). Subsequently, the models were trained using the 67% training data and their performance was evaluated using the 33% validation data. The target variable for the training data is Y1 (whether to purchase critical illness insurance), while the remaining columns in the dataset are used as predictor variables. After training the models, they were used to predict the target variable (Y1: whether to purchase critical illness insurance) for the Test dataset. The predicted values were then uploaded to the competition platform to obtain an initial score. For

this process, two models selected from the Model Selection section in section 4.2 were used: XGBoost and Logistic Regression.

For the XGBoost model, it was run in five different ways, and after each training, the model's predictions were uploaded to the competition platform, resulting in an initial score:

1. Default Parameters: The model was trained without any preprocessing, and the model parameters were set to their default values.
2. Parameter Tuning: The model's parameters were adjusted to optimize its performance.
3. Parameter Tuning and Feature Selection: In addition to adjusting the parameters, important features were selected to train the model.
4. Enhanced Data Processing: New data processing techniques were applied to improve the model's performance.
5. Enhanced Data Processing with Parameter Tuning: Both new data processing techniques and parameter tuning were applied to further improve the model's performance.

Similarly, for the Logistic Regression model, it was run in five different ways, and after each training, the model's predictions were uploaded to the competition platform, resulting in an initial score:

1. Default Parameters: The model was trained without any preprocessing, and the model parameters were set to their default values. The predictions were directly uploaded.
2. One-Hot Encoding and Parameter Tuning: The categorical variables were encoded using one-hot encoding, and the model's parameters were adjusted for better performance.
3. One-Hot Encoding, Parameter Tuning, and Feature Selection: In addition to one-hot encoding and parameter tuning, important features were selected to train the model.
4. One-Hot Encoding, Parameter Tuning, and SMOTE Sampling: In addition to one-hot encoding and parameter tuning, the dataset was balanced using the SMOTE sampling technique.
5. Enhanced Data Processing: New data processing techniques were applied to improve the model's performance.

After evaluating the initial scores for each model and approach, the best-performing models can be selected for further enhancement and testing.

For XGBoost, the first run was conducted without any preprocessing, and the model was directly uploaded. The validation results using 33% of the training dataset showed an AUC value of 0.815. The model's uploaded score on the platform was 0.8469.

For XGBoost, the second run involved adjusting the model parameters. The parameters were adjusted as follows:

1. Learning Rate: 0.05, 0.1, 0.15, 0.2, 0.25
2. max_depth: 3, 5, 7, 9, 12
3. min_child_weight: 1, 3
4. gamma: 0.0, 0.1, 0.2
5. colsample_bytree: 0.3, 0.5, 0.7, 1

The optimal parameter combination was found to be colsample_bytree = 1, gamma = 0, learning_rate = 0.1, max_depth = 3, and min_child_weight = 1. Running the model with these optimal parameters on the validation set (33% of the training dataset) resulted in an AUC value of 0.817. The uploaded score of the model on the platform was 0.8407.

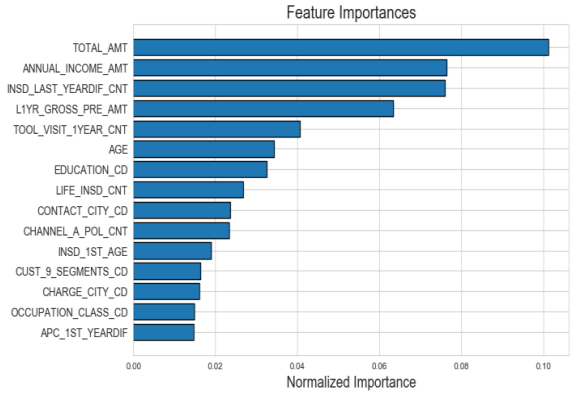


Figure 5

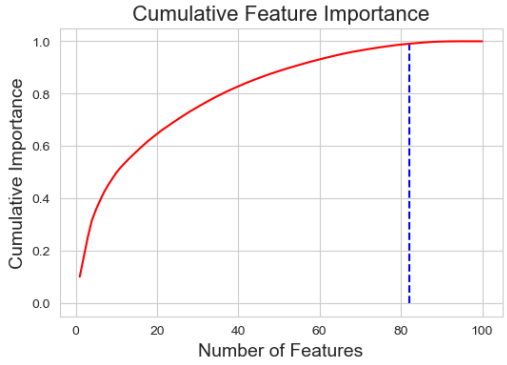


Figure 6

For XGBoost, the third run involved selecting important features and adjusting the model parameters. The feature selection process was performed using the FeatureSelector package, which removed features with a feature importance of 0 or that exhibited collinearity. Specifically, the features "IF_ISSUE_INSD_H_IND," "IF_ISSUE_M_IND," and "IF_ISSUE_E_IND" had a feature importance of 0 and were removed. Additionally, "IF_ISSUE_INSD_H_IND" and "IF_ISSUE_H_IND" exhibited collinearity, and one of them was removed. Based on Figure 5, which shows the top 15 important features for the "Y1" column, and Figure 6, which demonstrates that around 80 features already capture almost all the importance for the "Y1" column, the feature selection process was conducted. Next, the model parameters were adjusted using the parameter combination from Method 2, and the model was run with the optimal parameters. The validation results on the 33% of the training dataset showed an AUC value of 0.818, and the uploaded score of the model on the platform was 0.844.

For XGBoost, the fifth run involved adding a new data processing method and adjusting the model parameters. The new data processing method was the same as the one used in the fourth run. Then, the parameters were adjusted using the parameter combination from the second run, and the model was run with the optimal parameters. After adding the new data processing method and adjusting the parameters, the validation results on the 33% of the training dataset showed an AUC value of 0.8217. The uploaded score of the model on the platform was 0.8427.

For Logistic Regression, in the first run, the model was run without any preprocessing and directly uploaded to the platform. The validation results on the 33% of the training dataset showed an AUC value of 0.796. The uploaded score of the model on the platform was 0.8321.

For Logistic Regression, in the second run, the model was run with one-hot encoding and adjusted model parameters. The adjusted parameters were as follows: 1. Classifier penalty options were l1 and l2, 2. Classifier C values were 0.01, 0.1, 1, and 10. The optimal parameter combination was a classifier penalty of l1 and a classifier C of 1. After running the model with the optimal parameter combination, the validation results on the 33% of the training dataset showed an AUC value of 0.801. The uploaded score of the model on

the platform was 0.8329.

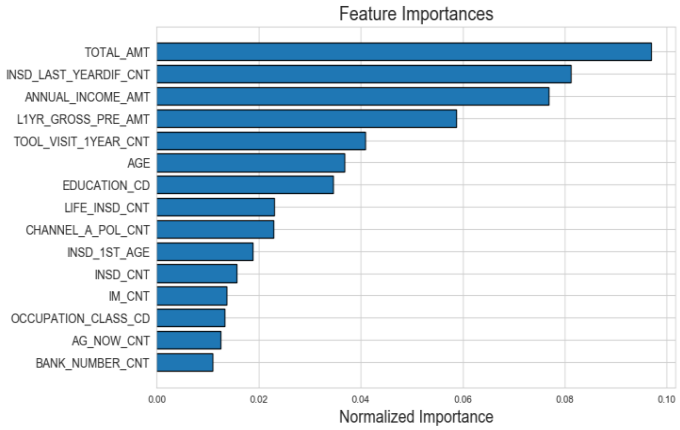


Figure 7

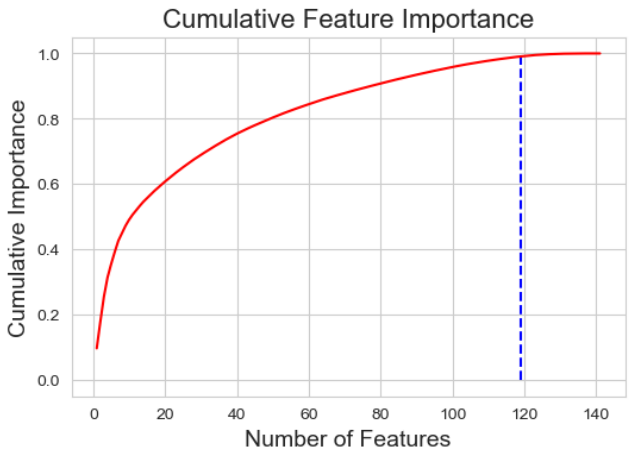


Figure 8

For Logistic Regression, in the third run, the model was run with one-hot encoding, adjusted model parameters, and feature selection. Since more one-hot encoding was performed compared to the third method of XGBoost, the feature selection results would be different. Feature selection was performed using the FeatureSelector package, where features with a feature importance of 0 and features with collinearity were removed. The features IF_ISSUE_INSD_H_IND, IF_ISSUE_M_IND, IF_ISSUE_INSD_E_IND, and IF_ISSUE_E_IND all had a feature importance of 0 and were removed. IF_ISSUE_INSD_H_IND and IF_ISSUE_H_IND had collinearity, and one of them was removed. From Figure 7, the top 15 important features for the Y1 column among all features can be observed, and Figure 8 shows that when the number of features reaches around 120, it almost includes all the importance

for the Y1 column. The model parameters were adjusted using the parameter combination from the second method, and the model was run with the optimal parameter combination. After performing one-hot encoding, removing unimportant features, and using the best parameter combination, the validation results on the 33% of the training dataset showed an AUC value of 0.801. The uploaded score of the model on the platform was 0.8329.

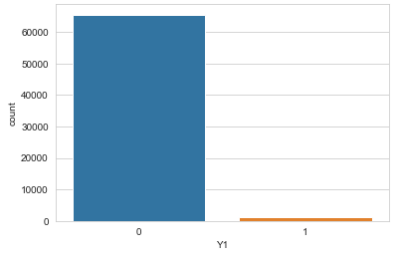


Figure 9

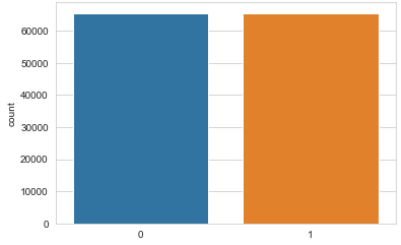


Figure 10

For Logistic Regression, in the fourth run, the model was run with one-hot encoding, SMOTE (Synthetic Minority Over-sampling Technique), and adjusted model parameters. Due to the highly imbalanced distribution of the target variable Y1 in the dataset, as shown in Figure 9, the Logistic Regression model is susceptible to the impact of this class imbalance. To address this issue, the SMOTE technique was applied to the training data. SMOTE creates a new set of data by randomly sampling and generating synthetic samples for the minority class (class Y) with replacement, as shown in Figure 10. This balances the distribution of the target variable Y1 between class N and class Y in the new data. The model parameters were adjusted using the parameter combination from the second method, and the model was run with the optimal parameter combination. After performing one-hot encoding, using the best parameter combination, and applying SMOTE, the validation results on the 33% of the training dataset showed an AUC value of 0.791. The uploaded score of the model on the platform was 0.8153.

For Logistic Regression, in the fifth run, a new data processing method was introduced. This new method did not aggregate the 15 columns of the AMT series, and missing values in each column were imputed using the median. Additionally, the AGE and IF_2ND_GEN_IND columns were combined to create a new column with three categories: "Young and 2nd Generation Policyholder," "Middle-aged and Non-2nd Generation Policyholder," and "Middle-aged and 2nd Generation Policyholder." After implementing this new

data processing method, the validation results on 33% of the training dataset showed an AUC value of 0.8030. The uploaded score of the model on the platform was 0.8298.

V. Result & Discussion

According to the final statistics, the XGBoost model performed best on the validation dataset when using the new data cleaning method and tuning the parameters. However, the actual uploaded scores showed that the XGBoost model performed best when no preprocessing was applied. Nevertheless, the platform will ultimately use a new dataset to test the models and determine the scores for the Private Leaderboard. Therefore, it is still uncertain which model will ultimately prevail. Additionally, Logistic Regression performed worse than any configuration of XGBoost, regardless of the method applied for preprocessing.

	Validation Data (AUC)	上傳成績 (AUC)
XGBoost，未處理	0.815	0.8469
XGBoost，調參	0.817	0.8407
XGBoost，調參 + 選擇特徵	0.818	0.8441
XGBoost，新的資料清理方式	0.8143	0.8440
XGBoost，新的資料清理方式+調參	0.822	0.8427
Logistic Regression，未處理	0.796	0.8321
Logistic Regression，調參 + One-hot	0.801	0.8329
Logistic Regression，調參 + One-hot + 選擇特徵	0.801	0.8329
Logistic Regression，調參 + One-hot + 重複抽樣	0.791	0.8153
Logistic Regression，新的資料清理方式	0.8030	0.8298

Table 6

VI. Conclusion

The goal of this project is to evaluate the likelihood of a specified customer purchasing critical illness insurance policy within the next three months based on their historical insurance data. Through exploratory data analysis, we examined the characteristics of the target variable, distribution of missing

values, and data features. We delved into suitable data cleaning and preprocessing approaches for categorical data and missing values in an insurance context.

We integrated six machine learning models and adjusted their parameters to obtain the best model solution for the current stage. The experimental results showed that XGBoost and AdaBoost methods performed the best. For the competition strategy, we selected XGBoost and Logistic Regression as the base models for improvement and competition.

As for future recommendations, due to time constraints, the models used for comparison in Section 4.2 (Model Selection) were evaluated without parameter tuning, which means that the performance of these models in Table 6 does not reflect their optimal performance. If each model had been compared based on its best performance, it could have potentially missed the opportunity to identify more promising models for testing and improvement. Therefore, future improvements in Section 4.2 (Model Selection) could lead to the training of more performant models.