1 Churn Prediction

1.1 Introduction

For this project, we developed a classification model to predict whether customers of a Belgian bank-insurer, with retail customers above 18 years old, will churn or not. The datasets utilized included information on customers for 3 consecutive months. The approach followed involved data pre-processing (cleaning, feature selection and engineering), model creation and model tuning.

1.2 Data

The provided datasets contain 63,697 anonymized entries on retail customers of a Belgian bank-insurer. The datasets contained the same features, for months T-2, T-1, and T, where month T also had the target variable, indicating whether an individual churned or not. The features provided included background personal and demographic information as well as bank-activity information. We were also provided with 3 testing datasets in order to create predictions. Finally, we apply the same steps mentioned for feature engineering in order to prepare the testing data for future use.

1.3 Data pre-processing

When pre-processing the data, three major steps were carried out, data cleaning, feature engineering and feature selection.

1.3.1 Data cleaning

The first step in cleaning the data was the identification of categorical and continuous features, this is necessary as they may be dealt with different when data is missing. After identifying the type of variables, we deal with missing data, with the overview of this missing data being seen in table 1.

Variable	missing mth 1	% mth 1	missing mth 2	% mth 2	missing mth 3	% mth 3
cust since all	234	0.37%	234	0.37%	234	0.37%
cust since bank	249	0.39%	249	0.39%	249	0.39%
cust occupation code	2002	3.14 %	2002	3.14%	2002	3.14%
cust relationship	14899	23.39%	14476	22.73%	14456	22.69%
cust children	23364	36.68%	23065	36.21%	23056	36.20%
cust education	47125	73.98%	47125	73.98%	47125	73.98%

Table 1: Missing churn data

Four out of the six variables have missing data for the same observations, meanwhile two of them, *cust relationship* and *cust children*, have data missing for different observations. Given that these two variables are very unlikely to change within a 3 month period, we replace any missing values in one month with data from another month if it is available. Additionally, we drop all observations with missing data for the *cust since all* and *cust since bank* variables since they account for a very small part of the data. This leaves us with four variables with missing data, where the *cust relationship* and *cust children* have 13823 and 22305 values missing respectively. Furthermore, we drop the *cust education* variable as the number of missing values is too large to consider any solution for.

We then proceed to look at each variable with missing data, by looking at their variability and potential effect on the target variable. The customer occupation code has 95% of its values for one category out of nine, with a relatively even distribution of churners across categories, therefore this variable is also dropped, and its missing values are not dealt with. Finally, the *cust relationship* and *cust children* variables are looked at, where the latter variable is transformed into a binary, yes/no variable for the sake of consistency (some of the values respond to a question yes/no, and some specify the age group of the children), while the former is already a binary variable. A look at the effect of the two categories relative to the target variable shows that

2.3% customers without kids churn, while 4.1% of customers with kids churn, and 2.6% of single customers churn, and 3.1% of customers in a relationship churn. These differences seem meaningful enough to warrant keeping the variables, therefore in order to retain them, an extra category for missing data will be added.

1.3.2 Feature engineering

In this subsection we create a *birth year* (which is utilized to create an *age variable*), *year since all year since bank* (which are utilized to create a *client since* variable). Additionally, we create variables for the difference between each feature at T and T-1/T-2, where categorical variables simply have a yes/no for a change, and continuous variables have the value by which it has changed. Finally, we remove observations with data that is inconsistent, this is done by identifying observations where a customer has a child but is less than 21 years old (since it is extremely unlikely for that to actually be the case).

1.3.3 Feature selection

In this subsection we start off by plotting stacked absolute value histograms and 100% stacked histograms for both the categorical and continuous variables, with a hue indicating the number of people who churned. Using these graphs we identify variables that clearly have no relationship with the target variable (whether its by being heavily concentrated on one value/category, or the distribution of churners being very even across values). This gives us all the continuous variables that we will retain, as well as categorical variables that we further investigated.

In order to finalize which categorical variables will be used, we carry out pearson chi-square tests at a 1% significance level to see if the target variable is dependent on the variable or not. This leaves us with twenty categorical variables and nine continuous variables.

1.4 Model creation and evaluation

In this section we carry out an initial model fitting process followed by some improvements to deal with the imbalanced target variable, and hyper-parameter tuning to create our final churn model.

1.4.1 Initial model creation and evaluation

We start off by defining a list of the models that we want to fit, evaluate, and compare their performance with each other. The classification models we define are: 'KNeighbors', 'RandomForest', 'AdaBoost','LogisticRegression', 'GradientBoost', 'XGB', 'BalancesRandomForest', 'BalancesBagging', 'RUSBoost'.

These models are fit onto the training data, and their confusion matrices are calculated as well as a 5-fold CV with AUC being the scoring parameter monitored. The purpose of the 5-fold CV is to see whether the model is over-fitting or not. However, the main metric used to evaluate the performance of our models is precision since we are interested in capturing as many true positives as possible, and in minimizing the number of false positives since there's only a limited amount of time and resources to investigate risky customers. The performance of the models can be seen in table 2, which shows that while the models seem to not be over-fitting (similar AUC and 5-CV AUC), they are performing badly at catching true positives (looking at precision and recall). The classifiers that use an under sampling technique for re-balancing the classes in the training data perform better than most others, while XGB has the best performance without any resampling. In order to attempt to improve model performance in terms of precision, we add miss-classification costs and try over/under sampling techniques for churned targets.

1.4.2 Model tuning

After initially building the models, we want to tune our models to optimize performance. The first way we seek to do this is to consider adding miss-classification costs, as well as data over sampling techniques

Model	AUC	5-CV AUC	Accuracy	Recall	Precision
GradientBoost	0.738	0.753	0.969	0.000	0.000
BalancedRandomForestClassifier	0.735	0.751	0.688	0.673	0.064
RUSBoostClassifier	0.729	0.727	0.722	0.584	0.063
AdaBoostClassifier	0.728	0.741	0.969	0.000	0.000
BalancedBaggingClassifier	0.692	0.714	0.802	0.441	0.070
RandomForest	0.684	0.720	0.969	0.000	0.000
XGBClassifier	0.681	0.702	0.968	0.003	0.077
LogisticsRegression	0.611	0.626	0.969	0.000	0.000
KNeighbors	0.554	0.579	0.969	0.000	0.000

Table 2: Initial models performance

and under sampling techniques.

Starting off with miss-classification costs, we use an inverse class distribution, resulting in a weight of 0.516 for non churned samples and 16.301 for churned samples. In this case more models have a non-zero precision, with XGB being once again the best performing model with a precision of 0.072. Since it seems to consistently be the best performing model, we continued to utilize only XGB when trying out data sampling techniques.

We then also try to over sample our minority class with the following ratios of minority to majority class: 0.1,0.15,0.2,0.5,0.7,1, using the SMOTENC function. Similarly these same ratios are considered when we apply under sampling using both the NearMiss function and the RandomUnderSampler function. The best performing model ends up being for a ratio of 0.1 using the RandomUnderSampler function, with a precision of 0.110.

Therefore, the best strategy seems to be to use a random under sampler with a ratio of 10% for the number of samples in the minority class over the number of samples in the majority class. The next step to further tune the model has two steps, firstly considering PCA, as it may improve the rate of training by simplifying the required structure to represent the data, and secondly applying hyper-parameter tuning on the learning rate, maximum depth, and minimum child weight. Additionally, the sub-sampling variable for creating trees will be tuned as it has been shown to reduce over-fitting. The results of tuning show that PCA does indeed improve precision, using 20 PCA components. Additionally, the optimal hyperparameters are a learning rate of 0.05, maximum depth of 7, minimum child weight of 25 and finally a tree sub-sampling rate of 0.6, resulting in an overall precision of 0.2 on the training data.

Finally, this XGB model is utilized on the testing data to create the final predictions that were posted on the leader board. We got 37 true positives while ranked 9th in the original leader board. In the final leader board, we scored 36 while ranked 5th. On one hand, the model performed similarly before and after the reveal of the hidden test set. On the other hand, the low precision score obtained in the training data is close to the ratio of true hits over number of churners (0.14). Hence, we will need to carry out additional feature engineering in order to improve the performance of the model.