Crescent Bank Competition

Should we call them?

We wanted to thank you for organizing this competition, we did learn a lot in the process and we very much enjoyed trying to solve the provided business question. (if you got board, please jump to the last two paragraphs.

As defined by the organizer the objective of this project was first to find rules to identify those customers who went delinquent and then determine if they are going to pay back their debt in less than 30 days or not. To be able to come up with a credible answer, we need to frame the question properly and build representative based on our definition drive the target and independent variables from the initial dataset.
To find our target customers, we used the UTD_DPD dataset. At first, we detected those customers who had a DPD value of 1 (this means the first day they became delinquent), then we observed them for 30 days to check whether they paid back their debt within 30 days or not. As this is an important step we are going to further elaborate some of our considerations:

- Some customers have failed to pay on time more than once, but since we need to take into account the IID assumption for training tree-based classifiers, we are going to only pick them once. Although one might argue that they are more dangerous and we should count them twice but we believe the complications will hinder model accuracy.
- We are also not going to count the number of times a customer was late as this would be a clear leakage.
- To be agnostic to the order of the occurrence of delinquencies for the customers who had not paid on time more than once, we are going to pick the occurrence randomly
- The other important issue that we had was that not all the customers have been tracked for 90 days, hence they did not have the same chance of being present in the target, however since by removing those that we have observed for a limited time, we would miss a considerable portion of data and it could be the case in many instances in reality, we did not remove these observations.

Using the mentioned methodology, we dropped those customers who never became delinquent, then allocated a value of 0 for those who became delinquent and 1 for those who did not pay back their depth in 30 days of becoming delinquent.

After defining the proper target value for observations, we merged our dataset with the UTD_DATA_1 dataset to be able to predict customers' behavior based on their specifications and previous transactions. We only need one record for each customer and the key point at this stage was to capture their characteristics on day 1 of becoming delinquent since that is the time we needed to decide whether call them or not.

Then we went through correlations and sources of data leakage inside our dataset and removed the sources of leakage from our data in an extremely conservative manner by running several initial models.

For the encoding part, we used One Hot Encoder for columns with less than 20 unique values and WOE encoders for columns with greater than 20 unique values in our dataset.

In terms of dealing with missing values, we used MICE imputer to tackle missing values to preserve the distribution of data as much as possible.

Then we did feature engineering for a subset of data to find hidden patterns that could help us increase accuracy. As we had limited computation capacity and to avoid overfitting, we selected only a few columns for this transformation. To decide which columns to pick for implementing feature engineering, we used Light GBM to observe feature importance and to choose candidate columns for feature engineering.

At this point, our data was ready and we started to train and tune models on it.

We decided to select our models from gradient boosting methods as they are proven to be extremely powerful in prediction. We trained three boosting models with their default set of hyperparameters as the base to compare our results with (LightGBM, XGBoost, and CatBoost). Then we used Bayesian hyperparameter optimization method to tune our models and contrary to our expectation, Bayesian hyper tunning did not improve our results much over default classifiers. We also ran a series of grid searches.

In the end, we tried stacking all those models as ensemble methods have shown a great performance. As a result, we managed to get a roc_auc score of 0.8630 from test data.

There are some notes to test our model on OOT data:

- First, datasets should be loaded. We assumed that they have the same name as the original data and are stored in a folder named ‘test’.
- All other objects that we have made (pickle files) should be present in the folder in which our notebook is going to be executed.
- We have used the following libraries that should be installed, for python 3.10:
    - Pandas, numpy, sklearn, category_encoders, joblib, dill, miceforest, autofeat, openpyxl, catboost, xgboost, lightgbm.
- miceforest library requires visual studio and C++ compiler to be installed on windows.

After installing the necessary libraries, you only need to run the final cell.

Now that we are finished with the project, we are questioning whether it was wise to use so many libraries? Hopefully, it does not lead to any compatibility issues.

Thanks Again
Best regards
Ali and Rouzbeh