Exit Report of Project for Customer

Customers: German Credit Company, Portuguese Banking Company

Team Members: Tomer, Esther, Eli, Ziv

Overview

To improve the baseline model, our approach was to focus on low performing subsets in the dataset. The process involved identifying the problem subsets, oversampling from those areas in a new dataset, and then retraining the model on the new dataset. With the model now having more emphasis on the problem areas, we were able to improve performance in those areas.

Business Domain

- German Credit Company, consumer credit risk assessor and loan provider
- Portuguese Banking Company, marketing and selling banking products

Business Problem

Both companies are looking to improve their classification models in the lower performing important sub demographics.

German Credit Company
The goal is to build a classification model that labels customers as good or bad credit risks. Misclassifying a customer as a good credit risk when they are in fact a bad credit risk (false positive), will be more harmful to the company's bottom line then the opposite case (false, negative). Due to that and the fact that the provided training set is unbalanced, recall is the appropriate accuracy measure to improve in order to most positively affect revenue.

Portuguese Banking Company
The goal is to build a classification model that predicts if leads will subscribe to a term deposit or not. Misclassifying a lead as subscribing when they in fact won't (false positive) will waste marketing resources and reduce the conversion rate for this product. The opposite case however will result in not pursuing a lead that would convert, missing the opportunity for increasing revenues. Both scenarios want to be avoided and due to the fact that the dataset is imbalanced, F1 score will be a more appropriate accuracy measure to use.

Data Processing

Original Schema

	Numeric Features	Categorical Features	# of Samples	Minority Class %	Baseline F1 Score
German	7	13	1000	30%	47.06%
Portuguese	7	9	45211	11.7%	51.97%

Our data did not require any preprocessing since we received clean data without any missing values. For processing our data, the first step was splitting into 3 sets for training, validation, and testing (70%, 15%, 15%). For each set we hold 2 versions, one raw with categorical columns for our synthetic data creation attempts, and the other with one-hot encoded columns for using with our xgboost model and decision tree model for identifying problem slices. For the German dataset, the 13 categorical features become 55 one-hot encoded features. For the Portuguese dataset, the 9 categorical features become 45 one-hot encoded features.

Modeling, Validation

We first establish a baseline by fitting the model to our training set and observing its performance on the validation set. This baseline will be updated with each iteration and be used to help us identify how to improve the model.

The next step is to identify problem slices in our validation set with our decision tree model. Instead of having our decision tree use labels for classification, we will have it observe which samples were classified correctly or not in the validation set. This is because we want our tree node purity to be based on the accuracy of its classification and not the class itself. We limit our tree depth to 2 to keep slices explainable and of a significant enough size. Once our tree has created its 4 leaf nodes, we will check to see which of these nodes make up at least 5% of the misclassified data in the validation set. The nodes that pass this test will be considered slices with minimal support. We will then choose the slice with the lowest F1 score in the validation set as our problem slice as long as its F1 score is at least 2% lower than the baseline F1 score for statistical significance.

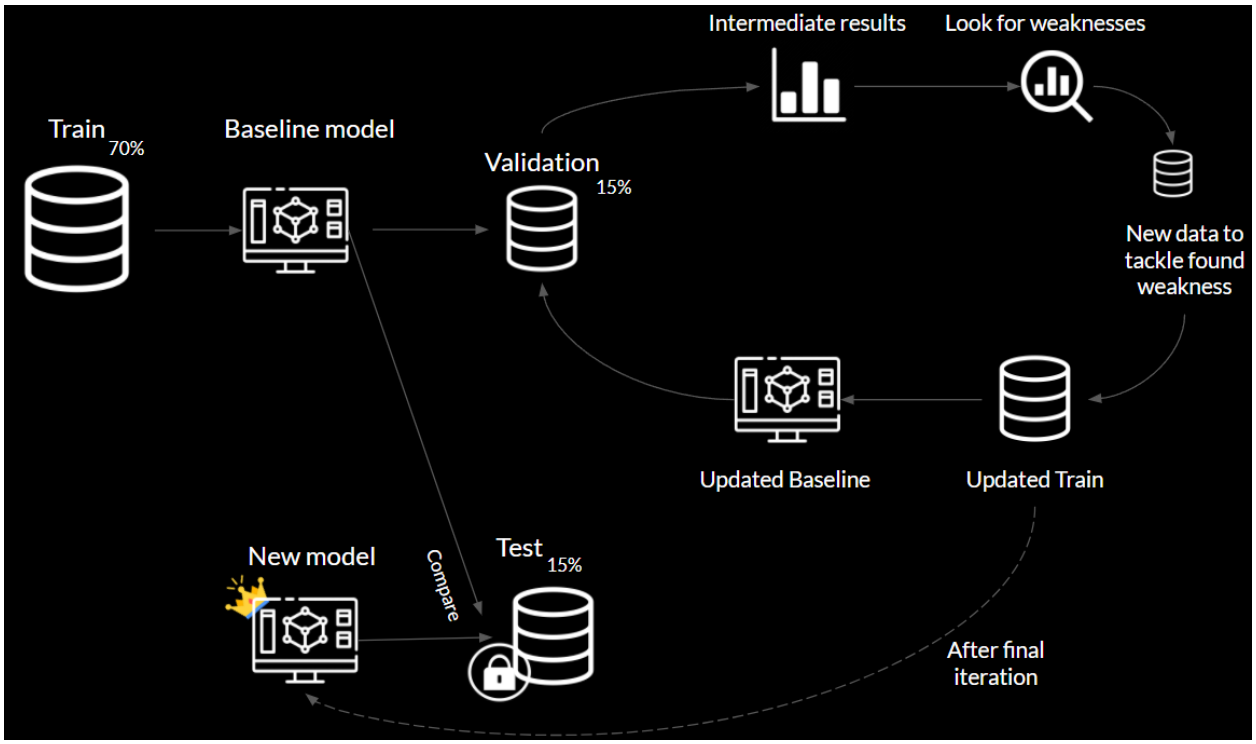
With our problem slice identified, we then randomly resample data from our training set that matches the features of our problem slice. Since the dataset has a heavy imbalance, we only resample from our minority class. One of the model parameters is to choose what percent we want to resample based on the size of the problem slice, in the minority class, in the training set, which we set to 50%. This resampled data is added to the original training set to create an augmented training set.

The augmented training set is then used to retrain the model. We do this for a set number of iterations, each time identifying a new problem slice and adding more resampled data to our

training set. Since we are updating our model based on performance in the validation set, it is important to keep a segment of untouched data, our test set, in order to prevent data leakage.

Solution Architecture

Implemented Architecture



As mentioned in the previous section, our model goes through several iterations, each time finding a new problem slice via a decision tree and updating the model accordingly. After the final iteration, we take the original baseline model, each updated baseline model, and our final model and compare their F1 scores on the test data. Since in each iteration the dataset is updated based on the validation set performance, depending on how many iterations we do, we may at one point overfit to the validation set. This is why we test the model of every iteration on the test set, because the last model may not be the best model due to overfitting. While we experimented with other architectures, this solution provided the largest improvement in our model.

Besides the fact that our model goes through several iterations, the other architectures we experimented with differed in two key areas:

Look for weakness

Instead of having a decision tree create bivariate slices we also explored creating univariate slices for each feature depending on the type of feature. For numeric features, we tried identifying high density regions (HDR). There are a few different approaches to this we explored but the general idea is to find an interval for the numeric feature where performance is weakest in the validation set. Ultimately this method is computationally heavy for searching the

distribution of each numeric feature which is one of the main reasons we prefer the decision tree. For categorical features we simply made a slice for each feature value. Just like in the decision tree model, we only considered univariate slices problematic if they passed our test for minimal support (at least 5% of misclassified validation data)and statistical significance (at least 2% lower F1 score than entire validation set). This process can create a lot of noise since many slices may be considered problematic. Deciding which ones are then the most problematic is a difficult task. If we use all the problematic slices in updating the dataset, there may be too much data oversampled, some of which may be overlapping in different problem slices. For these reasons , the iterative decision tree process was more efficient.

New data to tackle found weakness

For oversampling we chose to randomly resample existing data but there are also different synthetic data generation methods that we tried out. First we tried ADASYN, a variation of SMOTE that cannot use categorical features. Instead of using one-hot encoding, out of fear that we could get multiple values for a feature in the synthetic data (for example, a sample that has their job as unemployed and retired), we chose to input a factorized version of our dataset. This randomly assigned numeric values to our categorical values which in the end also proved inaccurate since ADASYN uses k nearest neighbors to generate data. It was also difficult to control how many samples would be generated since the library we used generates the amount of samples required to balance the classes. We also tried CTGAN for synthesizing data, which seemed more promising since it can handle categorical data. However, while this method seemed more accurate, neither of the synthetic data approaches improved our model. After further investigating the distributions of the generated synthetic data compared to the training set, there were noticeable discrepancies that we assumed ended up confusing the model more than helping it.

Customer Benefit

For our German and Portuguese customers, our updated model increased the baseline F1 score by 9.8 and 6.7 percent respectively. For the German credit company, this translates to increased revenue, by avoiding customers likely to default on their loans. For the Portuguese bank, this translates to an increased conversion rate for their marketing efforts which also increases their revenue.

Learnings

Project Execution

Since we were not sure what the best way to execute this project would be, we tried several different methods. This proved beneficial because we saw what worked best in each of our approaches and combined those things together to make the best performing model for our clients. The resampling strategy for oversampling, decision trees for identifying slices, and the minimal support and statistical significance measures we used all came from our 3 different

initial approaches. For the Portuguese bank specifically, our final approach F1 score was roughly 5% better than the initial approach we tried.

Data science / Engineering

When it comes to finding problem slices, you can get lost in the feature space. There are several approaches we tried, even brute force by checking every single categorical feature value as a potential problematic slice. Deciding how multivariate to allow problem slices to be, identifying problematic regions in numeric features, and how to judge the importance of a potential problematic slice, are all parameters that we can play around with more and even explore with different measures. Keeping it simple with a decision tree proved to be the most efficient strategy and yielding the best results.

Domain

In the financial industry, many classification models are trained to work with extremely unbalanced datasets. For this reason, assessing the performance of a model should use accuracy measures that better describe the model performance. For example, if a false positive is more undesired, recall will be the best measure to improve upon.

What's unique about this project, specific challenges

For this project, we explored a topic with little research and no publicly available code to reference. This is why we tried out a few different approaches for our implementation.

The first large challenge we faced was figuring out our method for finding high density regions for numeric features. At first we tried an iterative approach that would bound a range in the distribution and iteratively shrink the range by some value to see its effect on accuracy. If the accuracy increases, the excluded regions have a lower accuracy then the remaining window. This method did not prove to be effective for our project since we did not identify regions with significant changes in accuracy. This could be due to the many hyper-parameters such as the shrink size, number of iterations, and shrink direction. While not efficient for our experiment, this method could be much more efficient for large datasets where searching the distribution space can be computationally heavy. We also tried a KDE approach for finding a HDR for numeric features. This also proved inefficient since we were looking for one range for each feature and KDE produces multiple regions in a multimodal distribution. The best method we found was using Q-cut, which searches for the smallest region that holds 80% of the data then splits it into 4 regions that hold an equal amount of samples. We then chose the lowest performing region as our HDR. While this proved most effective in the case of finding numeric problem slices, we chose the decision tree method for creating our hybrid slices instead.

Synthesizing data also proved to be a challenging task as mentioned in our solution architecture. By looking at jobs for people between the ages of 51-59 in our training data compared to our synthetic data attempts, we were able to observe irregular values and differences in the distributions. In the future we will try more synthesizing approaches and will continue to do sanity checks to make sure our data makes sense.

Links

Git Repository

- https://github.com/sababaganoosh/ML-Ops-Final

Resources

- https://towardsdatascience.com/how-to-find-weaknesses-in-your-machine-learning-models-ae8bd18880a3A
- https://arxiv.org/pdf/1807.06068.pdf
- https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/
- https://arxiv.org/pdf/2108.05620.pdf

Next Steps

Now that we have created a proof of concept model, there are several areas to explore for further improving the model. First and foremost, adjusting hyperparameters such as depth of trees, number of iterations, or oversampling ratio may lead to a better performing model. Another area we would like to further explore is synthetic data creation. While the methods we initially tried did not improve the model, exploring other python libraries and parameters may lead to more accurate synthetic data that in turn improves model performance. For this project, our model improved results for customers who both had unbalanced datasets. We would like to further explore the application of our model, and see if it is able to improve performance on balanced datasets as well. Identifying and improving weak performing data subsets in a classification model has so many use cases, and we are excited to see how our implementation can be improved to better suit more of these applications.