Overview

You are provided a dataset indicating whether a customer has made a transaction based on anonymized numeric feature variables. Your task is to train a model that will be able to predict whether a customer will make a transaction, given the corresponding feature variables of that customer.

Description

The dataset provided to you contains information about customers, and the target feature is whether they made a transaction or not. The target features are anonymized - meaning you do not know what the features represent. The only information provided to you is that they are all numeric.

The training data provided to you has a severe class imbalance - 92% of target values are 0, while only 8% of target values are 1. This, however, does not mean that the testing data will be split in a similar ratio. Ensure you handle this class imbalance carefully.

In this competition, you are expected to build predictive models using traditional ML algorithms such as linear and logistic regression, SVMs, decision trees, random forests, kNN, bagging and boosting methods, clustering techniques like k-means, and other classical approaches. You are encouraged to try out various techniques of feature engineering.

My Solution

Because we had to predict a binary output (0 or 1), I used logistic regression as my model in this assignment.

I used the imported Pipeline to implement it cleanly, and that contained StandardScaler; for scaling all the features before applying functions on them.

As for parameters within logistic regression, I wanted special emphasis on preventing overfitting. So I initially tried l1 and l2 penalties, but I wasn't fully satisfied with the result they were giving me, so I settled on elasticnet which let me use both the penalties in a ratio which I decided by hit and trial. I chose my C to be 0.01 (a small value to heavily penalize overfitting).

Also since the dataset was heavily imbalanced in a 92-8 ratio, I used class_weight='balanced' to account for that. I had initially tried using smote but that further harmed my score, so I didn't go for it.

solver='saga' was just something chatgpt suggested for large tabular datasets, and so I included it.

I used StratifiedKFold for utilizing all the training data, and settled on 5 folds as anything above that was not really helping much.

Since the output of logistic regression heavily depends on the threshold it uses, I included a loop running through all the folds testing each threshold within a specified range, seeing which gives the best f1 score for the validation data of that fold. I then took the median of those thresholds as my final threshold. (One thing I noticed was that the lower I set the threshold upper limit, the better my f1 score was)

Finally I fit the data and predicted probabilities, converted them to class based on the threshold and converted to the submission file.