CONVOLVE 2.0 EPOCH-2

LIABILITY ACCOUNT MONITORING FOR MONEY MULES

April 20, 2024

Team: LAMe
Indian Institute of Technology Madras

Leshya A

Amizhthni PRK

Mohitabinav M

Contents

1 Introduction

1.1 Money Mules

Money mules are used to hide the origin of money gained through illicit activity. They are an instrument for money laundering used in the placement and layering of the money laundering process. The money mule's role is to transfer the money, thus helping hide the origin, which usually earns them compensation.

1.2 Goal

The goal is to develop predictive models using different supervised machine learning algorithms. The algorithms used for the model development are logistic regression, decision tree, and three ensemble methods. These models will then be evaluated based on several classification metrics. This process will ensure the selection of the most appropriate final model

2 Problem Statement

Given a data set containing various columns describing the nature of an account and whether it is a mule, create a ML model that learns on this data and predicts the probability that an account is a mule on a validation dataset.

2.1 Description of the dataset

The given dataset has 100000 rows and 178 columns describing account details like date of joining, demographics, transaction detalis . The demographics are marked with "demog_" , the transactions are marked with "txn_" and other details are marked as "other_". There is also a "target" column that describes whether the give account is a mule.
The validation data has the same set of input columns except the "target" column and has 50000 rows.

2.2 Analysis and pre-processing of Dataset

The given dev_data has 2000 mules which is 2% of the total data, which makes it highly class imbalanced. In addition to the high number of columns and extreme class imbalance the data also had a lot of NaN values which needed to be dealt with appropriately.
11 columns with zero variance i.e only one unique value were identified and dropped.
Columns with a very high number of NaN value for all mules were dropped.

On preliminary analysis there were 24 columns that had non-numerical data. Their descriptions and the way in which they were handled are as follows :

- Yes / No : Converted to 1 / 0.
- True / False : Converted to 1 / 0.
- High / Medium / Low : Converted to 3 / 2 / 1.
- Income and City Tier : Label Encoded in a sensible order.
- Mostly Numerical Columns : Changed the non - numerical part to a sensible number.
- Purely Categorical Columns : One-Hot encoded. Example : Email_Domain

3 Model Selection

All the models mentioned below are ensemble models except Decision Tree. Model ensembles refer to the technique of combining the predictions of multiple individual models to create a more robust and accurate overall model.
These methods may require more time and present challenges in terms of interpretability compared to other models. Nevertheless, they effectively reduce variance and enhance the robustness of predictions Unlike decision trees, Ensemble models are more suited for **imbalanced datasets**.
The following models were chosen based on the dataset.

3.1 Decision Tree

- This model is expected to not work satisfactorily since the dataset was not well balanced. And yet it was crucial to see the working of decision trees since they form the basis for many ensemble methods.

3.2 Random Forest

- For the Random Forest model, the boosting process improves the model by changing the weight with each iteration, adding new models with increased weights for misclassified instances and decreased weights for correctly classified instances.

3.3 Catboost

- CatBoost is particularly known for its efficient handling of categorical features.
- It can work with categorical data directly without the need for manual encoding

- CatBoost includes a robust algorithm that makes it resilient to noisy data and outliers.
- While CatBoost shares the fundamental principles of gradient boosting with other implementations like XGBoost and LightGBM, its focus on categorical feature handling and other features make it a distinct and valuable choice, especially in scenarios where categorical features play a significant role in the dataset.
- The main difference is Catboost grows with symmetric Trees unlike the other models mentioned below.

3.4 LightGBM

- LightGBM is designed for efficiency, using a histogram-based approach for tree building and offering faster training times, particularly on large datasets.
- While LightGBM can handle categorical features, it typically requires one-hot encoding for them
- The leaf-wise growth strategy in LightGBM may make the resulting models **less interpretable** compared to level-wise growth, as seen in XGboosting.

3.5 XGBoost

- XGBoost has built-in capabilities to handle missing data. It can automatically learn how to deal with missing values during the training process, reducing the need for preprocessing steps to impute missing values.
- XGBoost uses **depth-wise tree growth** combined with **pruning** to control the complexity of trees and mitigate overfitting.
- XGBoost provides a **feature importance score**, indicating the contribution of each feature to the model's predictions. This can be valuable for feature selection and understanding the impact of different variables on the model's output.
- XGBoost can handle large datasets efficiently. Its **parallelization** and **optimization** techniques make it **scalable** to handle big data scenarios.
- We also note that the XGBoost method works slightly better than the LightGBM method.

4 Approach

From the models explored above, for our final approach, we chose the best performing out of the above - XGBoost. After model selection, we now focus on improving the robustness of the model.

4.1 Choosing performance metric

Since we have an imbalanced dataset, using simple accuracy score to evaluate the model will be misleading. So we resort to a metric that places equal weightage to the prediction accuracy of each class.

- F1 Score : **F1 score** is ideal here as it tells us how effectively a model can identify positive cases while minimizing false positives and false negatives. F1 score is calculate as harmonic mean of **precision** and **recall**. Recall= TP / (TP+FN) and Precision = TP / (TP + FP).
- Balanced Accuracy Score : **Balanced Accuracy Score** is also used, which is an mean of the **sensitivity** and **specificity**. Where, Specificity = TN / (TN + FP) and Sensitivity = TP / (TP + FN). TP is True Positive, TN is true Negative, FP is false positive, FN is True Negative.
- Confusion Matrix : A **confusion matrix** which displays TP, TN, FP, FN in a matrix form was consistently used by us to avoid any chance of misinterpretation.

4.2 Hyperparameters Tuning

We selected hyperparameters based on the purpose and their performance.

- max_depth = 6 [We took 6 levels as we have a large dataset with a lot of features and a larger tree is required. Going any higher might lead to overfitting.]
- subsample = 0.8 [Again, as this is a large dataset, it's better to cut off few data points to avoid overfitting and reduce the effect of noisy data]
- scale_pos_weight = 49 [As a rule of thumb, this parameter is taken as the number of negative instances / number of positive instances]

4.3 Data Manipulation

As we have very less instances of the positive class, compared to the negative class, the model can tend to be biased towards the negative class and yield incorrect results. There is also relatively very little information about the positive class to train the model on. Manipulating the data, to get more information out of it, could improve the model. So we do sampling -

- Random Undersampling: Undersampling cuts off a lot of useful information about the negative class, yielding highly inaccurate results.
- Random oversampling: duplicates the minority class multifold. Although this improves the results, it is only to a minimal extent, as it is the same old information with no change in form.

- SMOTE Oversampling: Synthetic Minority Oversampling Technique or SMOTE is a technique where new instances are synthesized from the existing data. It basically looks into minority class instances and use k nearest neighbor to select a random nearest neighbor, and creates a synthetic instance randomly in feature space. This way, we generate new information from the little we have. We observed that using this method gives better results on all the models explored in the above section.

5 Training and Results

5.1 Overview

The test dataset has 400 mules and 19600 normal accounts to make a total of 20000 points. FP stands for false postives and FN stands for false negatives.

5.2 Decision Tree

- F1 score = 0.8500619578686495
- BAS score = 0.9271173469387755
- FP = 64
- FN = 57

5.3 Random Forest

- F1 score = 0.9093167701863354
- BAS score = 0.9565051020408164
- FP = 39
- FP = 34

5.4 Catboost

- Categorical :
  - F1 score = 0.9006451612903225
  - BAS score = 0.9355867346938775
  - FP = 26

  - FN = 51

- Without SMOTE :
  - F1 score = 0.9058064516129032
  - BAS score = 0.9381377551020408
  - FP = 24
  - FN = 49

- With SMOTE :
  - F1 score = 0.9029850746268657
  - BAS score = 0.9527040816326531
  - FP = 41
  - FN = 37

5.5 LightGBM

- Without SMOTE :
  - F1 score = 0.8963855421686747
  - BAS score = 0.9635204081632653
  - FP = 58
  - FN = 28

- With SMOTE :
  - F1 score = 0.9111389236545683
  - BAS score = 0.9541071428571428
  - FP = 35
  - FN = 36

5.6 XGBoost

- Without SMOTE
  - F1 score = 0.9108910891089109
  - BAS score = 0.9589795918367348

  - FP = 40
  - FN = 32
- With SMOTE
  - F1 score = 0.913939393939394
  - BAS score = 0.9700255102040816
  - FP = 48
  - FN = 23

5.7 Conclusions

As expected the decision tree model's performance isn't that great. This is expected as even though we are using SMOTE to balance out training data. Decision Trees and Random Forest dosen't support NaN values so they are only used along with SMOTE. Random Forest is better than Decision Tree because it is a much more robust model and has better imbalance handling capacity.
As expected using SMOTE gives the best balance of FP and FN for all the models. Without SMOTE gives more FN and lesser FP as expected since the data is imbalanced.
The overall best model is XGBoost with SMOTE which gives a very good balance of both FP and FN while also being robust against overfitting.

Code Collab Link