CUNY MSDS – DATA622

Homework 2

**Introduction**

For this assignment, I’ve chosen a dataset of simulated online payments, labelled to identify fraudulent transactions. I sourced the data from Kaggle [1], but the data ultimately comes from a 2016 paper from Lopez-Rojas et al [2] that introduced a payment simulator named PaySim.

The dataset provides ~6.35 million observations and includes nine features and one label:

1. step: represents a unit of time where 1 step equals 1 hour
2. type: type of online transaction
3. amount: the amount of the transaction
4. nameOrig: customer starting the transaction
5. oldbalanceOrg: balance before the transaction
6. newbalanceOrig: balance after the transaction
7. nameDest: recipient of the transaction
8. oldbalanceDest: initial balance of recipient before the transaction
9. newbalanceDest: the new balance of recipient after the transaction
10. isFraud: fraud transaction

The target variable (“isFraud”) is a binary label indicating whether a transaction is fraudulent (1) or not (0). The dataset is heavily imbalanced, with over 99% of the transactions being non-fraudulent.

The overall approach for this assignment is as follows:

* Exploratory Data Analysis
* Preprocessing
* Baseline Training
* Data Adjustment and Re-Training
* Hyperparameter Tuning
* Final Training
* Conclusions

The goal of this analysis is to (i) identify the differences between various tree-based methods for handling classification problems, with particular attention on the bias-variance trade-off; (ii) highlight impacts of changes to the underlying data, focusing on sensitivity versus robustness; and (iii) identify best-fit models for this data, in terms of predictive accuracy.

Code for all the following analyses is saved here: https://github.com/kac624/cuny/tree/main/D622.

**Exploratory Data Analysis and Feature Selection**

For each column, I analyzed the datatype and the number of unique values, summarized in Figure 1.

*Figure 1: Feature Data Types and Unique Values*

*A screenshot of a computer

Description automatically generated*

Based on this, I concluded that “amount”, “oldbalanceOrg”, “newbalanceOrig”, “oldbalanceDest”, and “newbalanceDest” are continuous, numerical features. While tree-based methods are typically robust to varied scales, I typically apply some kind of standardization with machine learning, and these features appeared to be reasonable candidates for such a transformation. Regarding the “step” feature, however, I noted that this was *not* continuous, instead representing discrete intervals of time. As such, I decided this was most likely left as is.

This left the categorical features: “type”, “nameOrig” and “nameDest”. While type had a reasonable number of categories (only five), the two name-related features exhibited very high cardinality. This condition was expected, given the large number of potential payers / payees who might be involved in an online transaction. So, I decided to keep only “type.” The two name-related features may hold some valuable information, but they would likely require significant analysis and processing.

Finally, the “isFlaggedFraud” feature appeared to be a duplicated of the target label, and the dataset’s documentation provided no details on it. So, I decided to drop it.

I examined the distributions of each variable through histograms (or bar charts for categorical features). Figure 2 shows the distribution for variables across the entire dataset, whereas Figure 3 shows the distribution for observations associated with the positive class (i.e. fraudulent transactions).

*Figure 2: Distribution of Variables – All Data*

*A screenshot of a graph

Description automatically generated*

*Figure 3: Distribution of Variables – Positive Class Observations Only*

*A screenshot of a graph

Description automatically generated*

These plots reveal a few key observations. First, the data is heavily imbalanced, with only 0.13% of the transactions related to the positive class. This might indicate the need for oversampling; however, as the following sections will show, this ultimately proved unnecessary). It does, however, confirm the need for stratified sampling when splitting training and validation data. Second, most of the numerical variables are heavily right skewed. As noted previously, tree-based methods are typically robust to such conditions, but his reinforces the potential value-add of scaling. Third, the positive class is associated with only two “type” categories: CASH\_OUT and TRANSFER. We can therefore expect the “type” feature to be important for modeling.

**Preprocessing**

Based on EDA, I applied the following to process data.

* One-hot encode the only categorical variable (“type”)
* Apply sklearn’s StandardScaler to continuous numerical variables (“amount”, “oldbalanceOrg”, “newbalanceOrig”, “oldbalanceDest” and “newbalanceDest”)
* Apply stratified sampling to split the data into training, validation and testing subsets using a 80% / 10% / 10% split.

On this final point, such a large dataset might allow for an even larger proportion of data under the training subset (e.g. 90%). However, given the heavy imbalance, I wanted to ensure that both validation and testing subsets had a sufficient number of positive class observations. So, I kept the training subset at 80%.

**Baseline Training**

In terms of modeling, I chose three algorithms:

* Decision Tree Classifier from sklearn;
* Random Forest Classifier from sklearn; and
* eXtreme Gradient Boosted Trees through the custom XGBoost implementation.

I trained a baseline model using all default hyperparameters. Below is a summary of the results across training and validation subsets.

*Figure 4: Baseline Training Results*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | DT | RF | XGB |
| **Train** | Accuracy | 1.00 | 1.00 | 1.00 |
| Precision | 1.00 | 1.00 | 0.97 |
| Recall | 1.00 | 1.00 | 0.80 |
| F-1 Score | 1.00 | 1.00 | 0.88 |
| **Validation** | Accuracy | 1.00 | 1.00 | 1.00 |
| Precision | 0.91 | 0.96 | 0.95 |
| Recall | 0.91 | 0.79 | 0.77 |
| F-1 Score | 0.91 | 0.86 | 0.85 |

The baseline models have relatively strong results, providing a near-perfect fit (allowing for some rounding) to training data (with the exception of XGBoost). We do see some evidence of over-fitting (i.e. high variance), as the models perform less well on the validation subset. The exception is again XGBoost, which has very similar performance across both subsets, indicating slightly higher bias, but lower variance.

Surprisingly, the single decision tree appears to provide the best generalized fit.

**Data Adjustment and Re-Training**

**Hyperparameter Tuning**

**Final Training**

**Conclusions**

**Sources**

[1] Roy, Rupak (2022). "Online Payments Fraud Detection Dataset." Sourced from https://www.kaggle.com/datasets/rupakroy/online-payments-fraud-detection-dataset/discussion/330548.

[2] Lopez-Rojas, Edger Alonso; Axelsson, Stefan; and Elmir, Ahmed (2017). “PAYSIM: A financial mobile money simulator for fraud detection.” Sourced from https://www.researchgate.net/publication/313138956\_PAYSIM\_A\_FINANCIAL\_MOBILE\_MONEY\_SIMULATOR\_FOR\_FRAUD\_DETECTION.