Fraudulent Transaction Detection

Haozhong Jia / hj2622
Qingcheng Yu / qy2281
Zain Merchant / ztm2105
Nikhil Balwani / nb3096

Abstract

Financial fraud is a major issue faced by businesses and consumers alike, with nearly a $5.8 billion drain on our economy every year (CNBC). To fight against this and make transactions more safe, financial firms have increasingly been investing in automated tools and models to help predict the legitimacy of transactions and automatically block those that seem to be nefarious. This system is not perfect, however. There is a constant aim to reduce false positives that add additional friction and nuisance to the process, while increasing the accuracy of actual fraudulent charges that are blocked. We aim to develop a fraud detection system that can detect fraud transactions with a high recall rate while keeping the false positives low. As part of our experimentation with the IEEE-CIS Fraud Detection dataset (Kaggle), we implemented a variety of different models, and from our findings, found XGBoost to perform the best.

1 Introduction

Whenever you have received a fraud alert or faced an inconveniently blocked transaction, you have unwittingly encountered the problem that the Kaggle IEEE-CIS Fraud Detection dataset aims to address. Although these systems may appear incomprehensibly complex, they are actually powered by the same data science techniques and machine learning models that we have studied throughout the semester in our Applied Machine Learning course. This paper applies our acquired knowledge to this dataset by starting with an exploration of our data and an understanding of the classification metrics in order to do a binary classification on if a transaction is fraudulent or not. We then clean the dataset to make it more easily digestible for the models we aim to train. We experiment with various models such as logistic regression, decision trees, random forests, histogram-based gradient boosting, XGBoost, and Feedforward Neural Networks. We compare their performance on our dataset and optimize the accuracy, AUC-ROC, AUC-PR, and F1 scores, which are commonly used metrics to evaluate model performance. Additionally, we investigate different techniques to enhance the base models and further improve our results.

2 Dataset Description

The dataset is provided by Vesta Corporation from Kaggle. The data comes from Vesta's real-world e-commerce transaction. The dataset consists of two parts: identity and transaction. Two types of data are joined by the feature TransactionID that is a unique timedelta from a given reference datetime. The identity data contains features about identification information - network connection information and digital signature associated with transactions. Due to aspects of security and privacy protection, all the features are masked and represented by numerical id except Device types and Device systems. The transaction data contains features about money transfer and other consumption services. This information provides information about the purchase of the goods as well as information about the seller and seller. There are 871 columns in the dataset.

3 Data Preprocessing

First of all, we checked the missing value rate for each of the features in the training dataset. According to Figure 1, we found that 47.9% of features have higher than 70% missing value rate. That means the dataset is very sparse, and about half of columns provide too little information for further applications. Therefore, we dropped the features with higher than 70% missing data. After removing too sparse columns we found there are some features that are categorical but show int64 and float64. Then we separated these categorical variables from

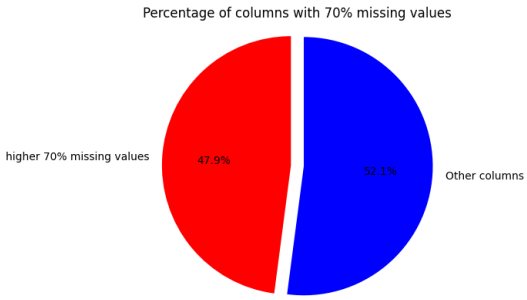

Figure 1: Percentage of columns with missing values

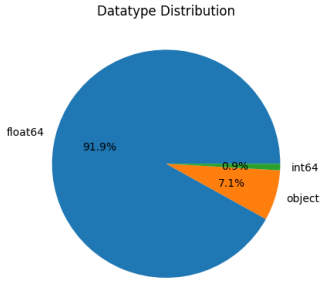

Figure 2: Datatype distribution

the dataset. The data-type distribution is shown in Figure 2.

For categorical columns, we decided not to fill in missing values because the loss of information could be part of the information of samples. We filled in numerical values with median values to improve the classification performance, as was a technique we learned in our course. Then, we drop highly correlated features (> 0.9), due to the fact that highly correlated columns provide redundant information for model training, and this may cause over-fitting. We then split the dataset into a development set and a test set in the ratio of 4:1. We utilized Stratified Splitting here to ensure that the ratio of our target classes in development and test datasets equals that of the original dataset. After, we implemented StandardScaler for numerical features, one-hot encoding for categorical features, and target encoding for high cardinality categorical features. The dataset is then checked for imbalance, which will determine whether to use the resampling method. According to Figure 3, the dataset shows extremely imbalanced that the number of fraudulent samples is much smaller than the number of non-fraudulent samples. To accommodate for this we implemented SMOTE (Synthetic Minority Over-sampling Technique) to resample the development set by synthetically adding minority

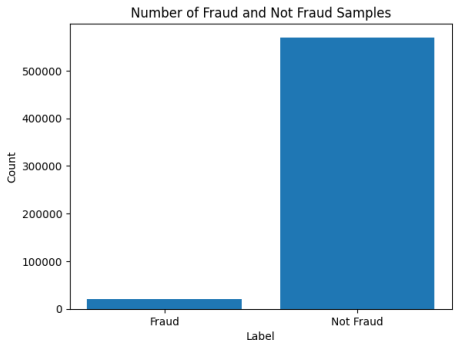

Figure 3: Imbalance in the dataset

class samples similar to ones already existing to reach an acceptable level of balance in the dataset. These were all techniques discussed in class that help refine our initial dataset during preproccesing to help improve and accommodate to the models we implement later.

4 Experiments

The following six experiments were carried out as part of our project:

4.1 Logistic Regression

Logistic Regression is a popular supervised learning algorithm used for binary classification problems. It is a linear model that utilizes the logistic function, which maps input features to a probability value between 0 and 1. This value is then used to predict the class of the input by setting a threshold, commonly at 0.5. Logistic Regression model can easily handle large datasets and can be trained quickly. Additionally, the coefficients of the logistic regression model provide insights into the importance of each feature, which can be valuable for feature selection and understanding the underlying relationships between variables. In this problem where fraudulent transactions are rare, overfitting may be a concern, so we tend to choose a smaller value of C (stronger regularization) by cross-validation, which can help reduce overfitting by constraining the magnitude of the coefficients. Besides, for this typically imbalanced datasets, where the number of fraudulent transactions is significantly lower than the non-fraudulent ones, assigning class weights can help the model pay more attention to the minority class. By using balanced weights, the model assigns weights inversely proportional to the class frequencies, which can lead to better overall performance in terms of metrics such as F1-score, precision, and recall.

Model	Accuracy	AUC-ROC	AUC-PR	F1-score
Logistic Regression	0.03	0.5218	0.0369	0.07
Decision Tree	0.96	0.8851	0.2502	0.48
Random Forest	0.94	0.8814	0.4108	0.41
Hist Gradient Boosting	0.98	0.9268	0.6265	0.59
XG Boost	0.98	0.9476	0.7327	0.67
Feedforward Neural Network	0.98	0.9386	0.7073	0.67

Table 1: Results

4.2 Decision Tree

Decision trees are another popular supervised learning algorithm that can be applied to our classification problem here. Decision trees work by recursively splitting the input data on the most informative features, and create a tree-like structure of decision rules that can be used for prediction. For our trials we used cross validation to tune the values of a few of the hyperparameters, specifically the criterion, max_depth, min_samples_split, and min_samples_leaf, in order to find the optimal combination that improves the AUC ROC of our model. Decision Trees, as we've used them, serve as a baseline to compare against our subsequent Random Forest and Histogram-based Gradient Boosting methods we discuss later. We had also used it to create a listing of our most highly relevant features which we wanted to scope out for our initial data exploration.

4.3 Random Forest

The Random Forest consists of several decision trees, each of which is constructed on a group of randomly sampled training data. Each decision tree corresponds to a random sample, and the random feature subset is used for training, which can reduce the variance of the model and improve the generalization ability. In our project, the hyperparameters for Random Forest model tuning are number of trees, number of features, maximum depth of the single tree, minimum sample split, and minimum sample leaves.

4.4 Hist Gradient Boosting

Histogram-based Gradient Boosting is an ensemble learning technique that combines multiple weak learners, typically decision trees, to form a strong learner. The model is built in a stage-wise manner, with each new tree being added to the ensemble to correct the errors made by the previous trees. The input features are binned into histograms, which accelerates the training process by reducing the number of possible splitting points for each feature. Histogram-based Gradient Boosting model is able to provide high accuracy and are less prone to overfitting than other boosting techniques. The use of histogram binning allows for faster training and prediction, even with such a large dataset. More worthy to mention that the algorithm can handle missing values without the need for explicit imputation. In this problem, we choose max_leaf_nodes, max_depth, min_samples_leaf as hyperparameters, all of which affect the complexity of the trees.To find a balance between underfitting and overfitting, we utilize cross-validation to tune the value of them. Besides those, L2 regularization is added to constrain the magnitude of the leaf values in the trees. In the context of fraudulent transaction detection, where overfitting can be a concern due to class imbalance, tuning this parameter can help achieve better generalization. Therefore, the reason why we choose the Hist Gradient Boosting is that it offers higher accuracy and faster training, making it an attractive option for detecting patterns and relationships in the data that might be missed by simpler models.

4.5 XGBoost

XGBoost is one of the most popular implementations of gradient boosting. In our project, the hyperparameters for XGBoost tuning are the number of estimators, learning rate, regularization parameters, maximum depth, and maximum leaf nodes. The reason we choose to use XGBoost is because it can handle spare data. It uses distributed weighted quantile sketch algorithm, which makes it process weighted data efficiently.

4.6 Feedforward Neural Network

Feedforward neural networks are a class of artificial neural networks where the information flows in a unidirectional fashion form the input layer to the

output layer. Each layer is a linear projection of the input features followed by an activation function that adds non-linearity.

We experimented with various neural network architectures for this problem. The one that performed the best in terms of AUC-ROC and AUC-PR metrics was the one with 500 and 200 neurons in the intermediate layers with ReLU activations. The sigmoid activation function was used on the output layer to indicate the probability of the input being from the positive class.

5 Results

As described in section 4, six different experiments were carried out, namely - Hist-Gradient Boosting, Logistic Regression, Decision Tree, Random Forest, XGBoost, Feedforward Neural Network. Special emphasis was placed on metrics pertaining to the minority class, like AUC-ROC, AUC-PR, and F1-score. These metrics are detailed in table 1. Figure 4 shows the Receiver Operating Characteristic Curves for the five experiments. It is obvious that XGBoost outperforms (or gives the best performance) for all metrics. The other gradient boosting alternative - Hist Gradient Boosting - performs moderately. XGBoost is closely followed by the Feedforward Neural Network experiment. Since the network is two layers deep with ReLU activations to add non-linearity, it can easily capture non-linear patterns in the dataset. The other tree ensemble-based alternative - Random Forest - performs worse than all other tree-based approaches. While the vanilla decision tree performs poorly on AUC-ROC and AUC-PR, it gives moderately good accuracy and F1 score. It appears that boosting-based tree approaches are a good fit for this use-case. With an AUC-ROC score of just above 0.5, logistic regression performs only slightly better than random-guessing. We attribute this behavior to the fact that logistic regression is a linear model and lacks the ability to capture complex non-linear patterns in the dataset.

6 Conclusion

Through our exploration of the Kaggle IEEE-CIS Fraud Detection dataset, we have successfully applied a variety of data science techniques that we have learned throughout our coursework. Starting with a raw dataset, we discussed various ways to handle missing or sparse metrics, categorical and numeric data, and highly correlated features. Additionally, we learned how to work with class imbalanced datasets, a common challenge in real-world applications. Through iterative improvements to the models we applied, using techniques such as K-fold cross validation to select optimal hyperparameters, we were able to identify XGBoost as the optimal model we tested based on all our sought after metrics. Overall, our work on this dataset has provided us with valuable experience and insights that can be applied to real-world problems in the future. We'd like to thank our instructor and TAs for providing valuable instruction and guidance throughout the semester.

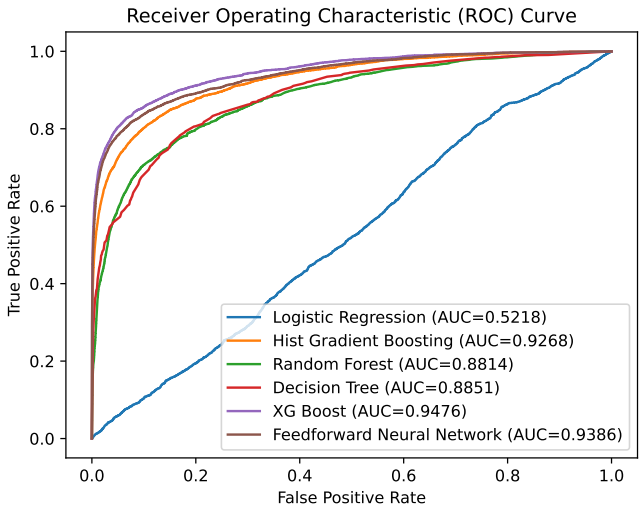

Figure 4: Receiver Operating Characteristic Curves for different experiments

References

CNBC. Consumers lost $5.8 billion to fraud last year — up 70% over 2020.

Kaggle. Ieee-cis fraud detection.