**Online Payment Fraud Detection**

Team 11 - Zihan Wan & Feifan Tai

**1. Introduction**

**1.1 Background**

With the development and popularity of online and mobile payments, people still face a considerable risk of payment fraud while enjoying this convenience. Under the cover of malicious technology, fraudsters are becoming more sophisticated, and the amount of fraud continues to rise. Therefore, it is crucial to identify suspicious transactions quickly and accurately.

**1.2 Motivation**

For enterprises, payment fraud not only hurts revenue, but also undermines consumer confidence and even corporate reputation. So, they need to proactively assess their exposure to fraud risk and respond appropriately. For individual consumers, payment fraud can have more serious consequences, even affecting lives.

In the face of the increasing volume of payment transaction data and the rapidly changing market, machine learning algorithm models need to be constantly iterated and updated, and parallel computing methods can undoubtedly save a lot of time and labor costs.

**1.3 Goal**

We plan to use machine learning to identify online payment fraud so that we can better understand the prevalent characteristics of fraudulent transactions. We also intend to perform data processing and model training on multiple CPUs and multiple GPUs with the help of parallel computing methods to achieve speedup and improve accuracy, and finally analyze and summarize the different results.

**2. Data Description**

Source Link: <https://www.kaggle.com/datasets/jainilcoder/online-payment-fraud-detection>

This dataset (493.53 MB) contains historical information about fraudulent transactions, its basic information is as follows:

|  |  |  |
| --- | --- | --- |
| Data Size | 6362620 rows × 11 columns | |
| Data Types | int64(×3) / float64(×5) / object (×3) | |
| Target | “isFraud” (1/0) | |
| Features | step | time units, 1 step = 1 hour |
| type | type of online transaction (payment / transfer / ...) |
| amount | amount of the transaction |
| nameOrig | name of the customer starting the transaction |
| oldbalanceOrg | balance before the transaction |
| newbalanceOrig | balance after the transaction |
| nameDest | name of the customer receiving the transaction |
| oldbalanceDest | initial balance of recipient before the transaction |
| newbalanceDest | new balance of recipient after the transaction |
| isFraud | whether it is a fraudulent transaction? |
| isFlaggedFraud | Is the transaction identified as fraudulent? |

**3. Methodology**

**3.1 ML Algorithms**

Our main topic can be simply summarized as a binary classification problem to determine whether fraud is happened in a transation, in which case we prefer logistic regression to solve problem. Considering that these columns may be not independent and for a better performance our model has, we plan to try to train different machine learning models, such as decision tree, random forest, logistic regression, XGBoost, etc., to compare and get better prediction results. Besides, random forest is easier to realize parallel operation by training single decision tree in parallel.

Since identifying transaction fraud falls under the scope of risk control and there is an imbalance in the dataset, we also consider using resampling techniques for the dataset.

**3.2 Parallel Methods**

Given the large volume of data, we want to optimize the data processing and data analysis part by Dask to improve the operation efficiency. For example, We can use dask to read data in chunks. Using the chunksize parameter, we can create chunked IO streams for a given dataset, reading up to a specified chunksize of rows at a time. This allows us to split up the tasks for the entire dataset into smaller tasks and aggregate the results.

At the same time, the training and tuning of basic machine learning models takes a lot of time, especially in cross validation and grid search, which require a lot of computing power. Therefore, we plan to perform the above operations with the help of multiprocessing techniques and reasonable deployment on multiple CPUs and GPUs.