Fraud Detection

Ming Ki Toby Cheng, Xinyue (Anna) Jin, Ling Jiang
03/18/2020

Agenda

Problem Description

Data Exploration

Data Pre-processing

Approaches

Challenges & Solutions

Analysis Results

Insights Gained

Future Work

Problem Description

- The growing usage of digital payment increases the likelihood of fraudulent transactions.

- Most of the detection mechanisms are only based on simple thresholds assigned. (Only rejecting transactions over 200K in this case)

Objective

- Find the best model with lowest FN while balancing precision and recall

- Identify key features of fraudulent transactions

Data Exploration

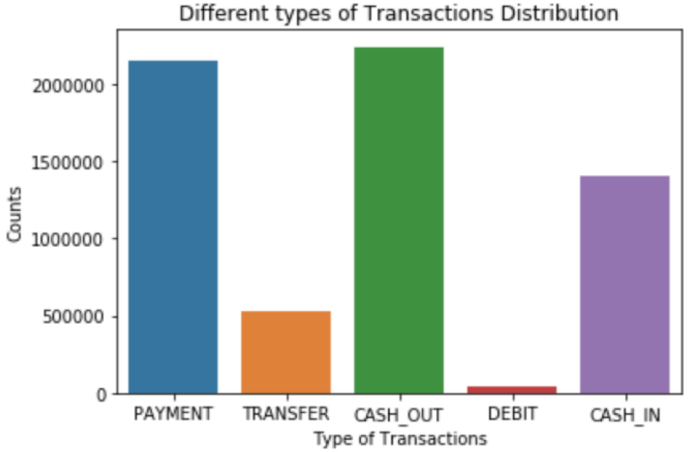
Data Description

- Synthetic financial transaction dataset generated by simulator.
- The dataset contains approximately 6 million observations, which includes 6 million transaction initiators and 2 million recipients.
- Each transaction has 11 variables without missing values.
- Five types of transactions (Cash-in, Cash-out, Debit, Payment, and Transfer).
- Kaggle link: https://www.kaggle.com/ntnu-testimon/paysim1

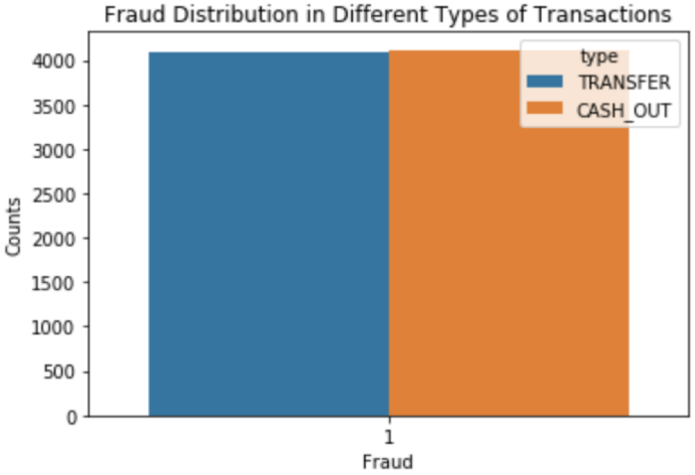
Data Distributions

- Highly imbalanced data

- Most transactions in Payment and Cash-out



Fraud based on Different Types



Fraud transactions only occur in transaction types **CASH_OUT** or **TRANSFER**

Correlations

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest
amount	1.000000	-0.002762	-0.007861	0.294137	0.459304
oldbalanceOrg	-0.002762	1.000000	0.998803	0.066243	0.042029
newbalanceOrig	-0.007861	0.998803	1.000000	0.067812	0.041837
oldbalanceDest	0.294137	0.066243	0.067812	1.000000	0.976569
newbalanceDest	0.459304	0.042029	0.041837	0.976569	1.000000

High Correlations between **oldbalanceOrg** and **newbalanceOrig**
High Correlations between **oldbalanceDest** and **newbalanceDest**

Data Pre-Processing

Data Preparation

Create four new variables:

- **OrigDiff** → the difference in the original user balance
- **DestDiff** → the difference in the end user balance
- **AmountEqOrig** (binary)→ whether or not the Amount transferred equals the original user's starting balance
- **CusToCus** (binary)→ whether the transaction was customer to customer (as opposed to customer to merchant)

Difference in Balances

type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	OrigDiff	DestDiff
PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0	-9839.64	0.0
PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0	-1864.28	0.0
TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0	-181.00	0.0
CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0	-181.00	-21182.0
PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0	-11668.14	0.0

OrigDiff = newbalanceOrg - oldbalanceOrig

DestDiff = newbalanceDest - oldbalanceDest

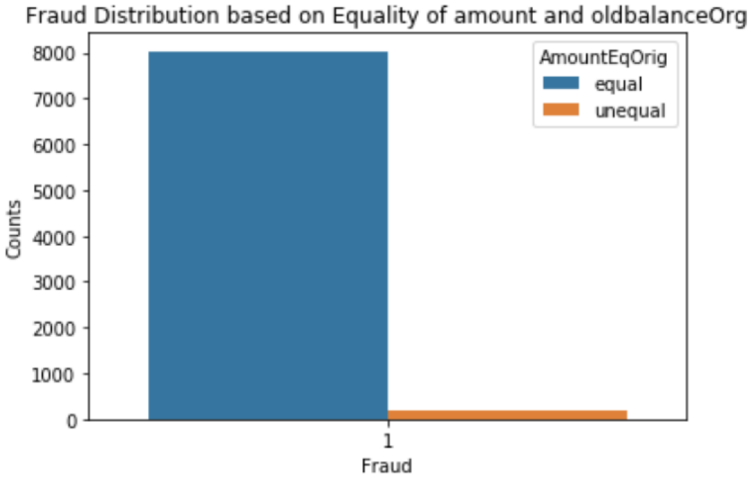
Two Account Types

- “C”(Customer) and “M”(Merchant)
- Only **C2C** and **C2M**
- Fraud transactions only occur in **C2C**

step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Equality Check

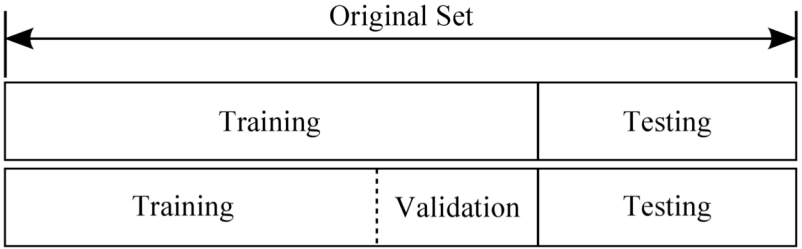
- Intuition: Wrongdoer would like to steal **all** money from the mobile wallet
- Check whether the equality of amount and original balances affect fraud



- Most fraudulent transactions shows that amount **equal to** original balances.

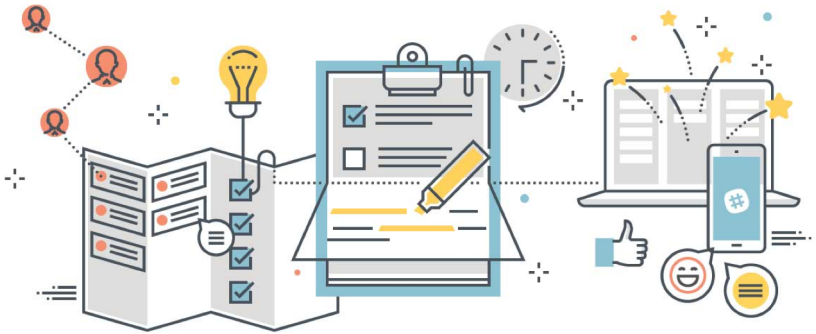
Data Split

- Split into the training and test sets based on a fraction of 80% and 20% respectively for PySpark



- Split into the training, validation, and test sets based on a fraction of 60%, 20%, and 20% respectively for SageMaker

Approaches



Feature Selection

Features chosen for modeling:

type, amount, OrigDiff, DestDiff, AmountEqOrig , CusToCus

	isFraud	amount	payment	transfer	cash-out	debit	cash-in	OrigDiff	DestDiff	AmountEqOrig	CusToCus
0	0	9839.64	1	0	0	0	0	-9839.64	0.0	0	0
1	0	1864.28	1	0	0	0	0	-1864.28	0.0	0	0
2	1	181.00	0	1	0	0	0	-181.00	0.0	1	1
3	1	181.00	0	0	1	0	0	-181.00	-21182.0	1	1
4	0	11668.14	1	0	0	0	0	-11668.14	0.0	0	0

PySpark - Logistic Regression

- Aim to reduce as many False Negatives as possible
- One-Hot-Encode transaction types for PySpark
- Vectorize and Standardize Features
- Create classWeights Column for use in Logistic Regression
- Predict on both train and test and obtain AUC to check for overfitting

PySpark - Multilayer Perceptron Classifier

- Neural Network, feature importance hard to interpret
- Use same features and design as Logistic
- Unable to add class weights
- 1 Hidden layer with 1 Node
- Predict on both train and test and obtain AUC to check for overfitting

SageMaker - Logistic Regression

- One-Hot-Encode transaction types manually
- 60% - 20% - 20% split used for Train/Val/Test
- Used Linear Learner Algorithm with binary predictor type
- Could realistically only predict on test set
- First try model with static hyperparameters, optimizing for loss
- Try additional model with optimizing for maximizing precision at high recall

SageMaker - XGBoost

- Gradient boosted trees algorithm, no weights for every feature
- Static Hyperparameters: max_depth: 5, min_child_weight: 6, Objective: multi:softmax
- Batch Transform on test data
- Hyperparameter tuning job adjusting eta, subsample, min_child_weight, max_depth with Objective validation:f1

Challenges & Solutions

Imbalanced Data

Challenge

- The overall accuracy might be high, but without generating any good insights.

Solution

- Set class weight when creating models.

Feature Selection and Engineering

Challenge

- Features selection had a large impact on model performance.
- High correlations exist between features, which may cause multicollinearity.

Solution

- Spent large portion of time was spent on understanding the data and engineering features
- Linearly combine the independent variables to address the multicollinearity by subtracting old balances from new balances (“OrigDiff” & “DestDiff”).

Working with SageMaker

Challenge

- SageMaker is relatively new (a little over 2 years old) and there are not as many relevant posts with helpful answers for debugging online

Solution

- Spend large amounts of time find ways to debugs.

Model Selection

Challenge

- Choose the best model among a set of candidate models.

Solution

- Based on F1 score to balance precision and recall
- Choose the model with the smallest number of FN (false negative)

Analysis Results

Results

Model Performance Comparison on Testing Dataset					
Models	**Platforms**	**F1 Score**	**AUC**	**False Negative**	**False Positive**
Logistic Regression	PySpark	0.9963	0.9993	8	4
Neural Network (Multilayer Perceptron Classifier)	PySpark	0.9976	0.9989	8	0
Logistic Regression	SageMaker	0.9941	0.9976	8	12
XGBoost	SageMaker	0.9976	0.9976	8	0

Choose **Logistic Regression Model** in PySpark

- Low false negatives & false positives
- Easy to interpret and find the important features

Feature Odds Ratios

Feature	**Odds-Ratio**	**Interpretation (**Assuming all else constant)
Cash-Out	**1.3319**	**33.19% higher fraud probability than Debit transactions**
Payment	0.8565	14.35% lower fraud probability than Debit transactions
Cash-In	0.4113	58.87% lower fraud probability than Debit transactions
Transfer	**1.2521**	**25.21% higher fraud probability than Debit transactions**
Amount	1.5100	51% higher fraud probability per unit increase in amount
OrigDiff	0.9497	5.03% lower fraud probability per unit increase in original balance difference
DestDiff	0.6234	37.66% lower fraud probability per unit increase in destination balance difference
AmountEqOrig	**1.500**	**50% higher fraud probability if Amount transferred equals origin balance**
CusToCus	**1.1675**	**16.75% higher fraud probability if the transaction is customer to customer**

Insights Gained

Fraudulent Transaction Mostly Occur

1. In Cash-Out or Transfer transactions

2. When the transaction is being made from customer to customer

3. When the transfer amount is equal to the original balance

Actions

- Use the current model to generate probability and predictions
- Add verification or other barriers to ensure safety of account
- If unable to verify, block the transaction until secure verification

Future Work

Future Consideration

- Consider the number of FP (false positive) and avoid it being too high

- Improve our model by capturing unidentified patterns
  EX: split “amount” into bins

- Need to update our model to capture new features that may influence on fraud and keep up with the changing fraudulent agents

Thank You!