Detecting Anomalous Transactions: Leveraging Local Outlier Factor for Credit Card Fraud Detection

Yixuan Deng
yixuand@umich.edu

1. Introduction:

In the modern financial landscape, credit card transactions play a significant role in consumer spending, making the detection of fraudulent or abnormal transactions a critical task for financial institutions. The presence of such anomalies can have far-reaching consequences such as improper business decision making. As such, there is a growing need for robust methods that can accurately identify abnormal transaction patterns, ensuring the integrity and account security of the financial system.

This project focuses on leveraging the LocalOutlierFactor (LOF) algorithm to identify outliers within datasets provided by Stori. The datasets encompass the transaction history of all users, with each record capturing essential details of the transactions conducted over specific payment cycles. In this context, outliers are defined as transactions that deviate significantly from normal behavior, often indicative of potential fraud or errors. Through this analysis, I aim to uncover potential fraudulent activities and offer recommendations for checking such anomalous transaction behaviors. The findings of this project could contribute to the development of more sophisticated fraud detection systems.

2. Data Description:

The datasets utilized in this project are *ua.indra_acct_pmt_summary_behavior* and *stori_ccm_pl.account_daily_ledger_info_indra_v2.* A sql script is written which retrieves the data by joining the *ua.indra_acct_pmt_summary_behavior* table with filtered data from the *stori_ccm_pl.account_daily_ledger_info_indra_v2* table. When filtering the daily ledger table, the setup condition guarantees that the snap date is equivalent to the statement cycle due date. Besides that, another setting is made such that only data with snap date after '2024-01-01' will be retrieved. Since daily ledger table is a massive dataset, filtering by a fixed date is very helpful to reduce the total runtime when retrieving data. In addition, the script ensures that only rows

where the accounting_date falls between the cycle_start_dt and cycle_end_dt are included in the result. Lastly, the output is ordered by accounting_date in descending order, and only the first 10,000 row will be selected as the test data. After the output table is generated, I exported into a .csv file for the model testing demonstrated in the next section.

```sql
--filter accounting date btw cycle start and end
SELECT iapsb.*, adlii2.*
FROM ua.indra_acct_pmt_summary_behavior as iapsb
JOIN (
    SELECT *
    FROM stori_ccm_pl.account_daily_ledger_info_indra_v2
    WHERE snap_dt = stmt_cycle_pmt_due_dt
    AND snap_dt >= '2024-01-01'
) AS adlii2
ON adlii2.indra_external_acct_id = iapsb.external_acct_id
AND iapsb.accounting_date BETWEEN adlii2.cycle_start_dt AND adlii2.cycle_end_dt
ORDER BY accounting_date DESC
LIMIT 10000;
```

3. Methodology:

The project employs a structured approach to identifying outliers in a dataset containing credit card transaction histories using the LocalOutlierFactor (LOF) algorithm. The methodology involves several key steps, which are implemented using Python, leveraging powerful libraries such as *pandas, matplotlib, scikit-learn,* and *numpy*. In the following paragraph, the usage of packages will be elaborated.

The first main part of the code does the data loading and preprocessing job. The *pandas* library is utilized to load the dataset (referred to as ledger3.csv) into a DataFrame. Then, three features are selected from the dataset: tot_pmt_cnt (Total Payment Count), tot_pmt_amt (Total Payment Amount), and start_posted_bal_amt (Starting Posted Balance Amount). In addition, to ensure that the features contribute equally to the model, the *StandardScaler* from *sklearn.preprocessing* is applied to normalize the data.

The second part of the code plays a role of detecting outlier using the LOF algorithm. LOF is a density-based anomaly detection algorithm that identifies outliers by comparing the local density of a data point with its neighbors. In fact, there is a build-in LocalOutlierFactor (LOF) function within the sklearn.neighbors module. In fact, when applying that function, the parameter settings are extremely crucial. More specifically, *n_neighbors* set to 100, meaning it considers the 100 nearest neighbors for each data point. The *contamination* parameter is set to 0.01, indicating that approximately 1% of the data is expected to be outliers. Moreover, since I

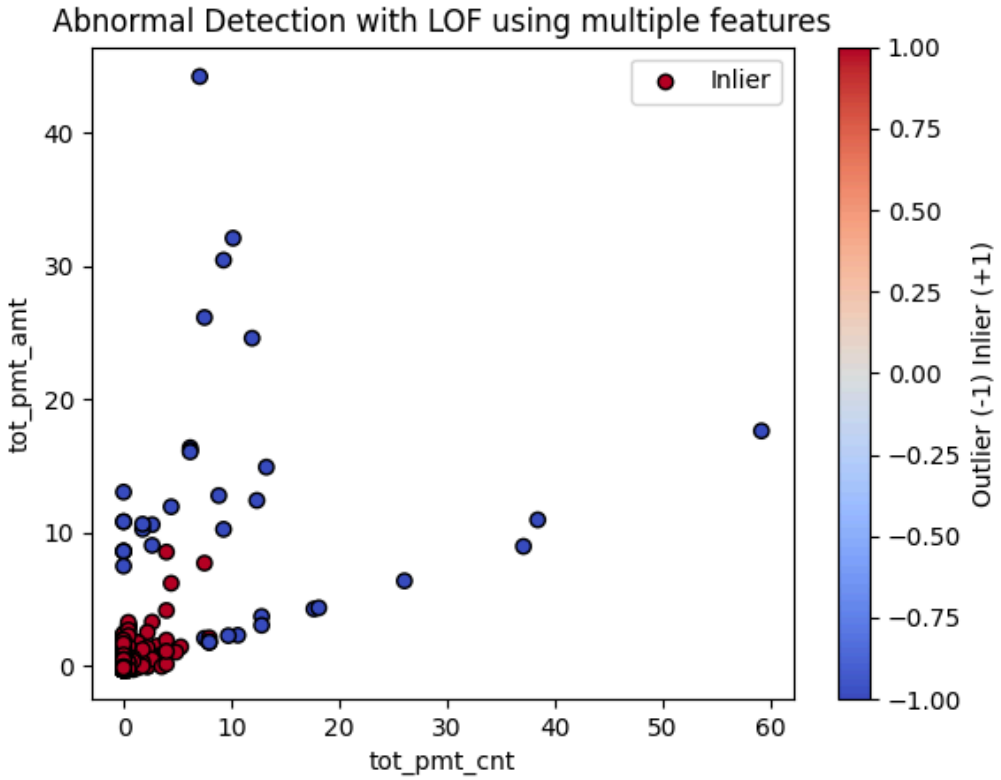
want to detect outliers within the training dataset itself rather than new, unseen data, the last parameter *novelty* will be set to False.

The third part of the code will perform data post-processing conducting the false positive analysis. In this scenario, false positive means normal transactions(inlier) that are classified as abnormal behavior(outlier). Therefore, the main goal of this part is to reduce the false positive rate by filtering out plausible outliers. To achieve this data post-processing, I focused on analyzing *'tot_pmt_amt'* and *'start_posted_bal_amt'*. Since *'start_posted_bal_amt'* have either poositve or negative value, while *'tot_pmt_amt'* only have positive value. It turns out that there are only two ways of combination and hence a threshold for each situation was generated. When *'tot_pmt_amt'* and *'start_posted_bal_amt'* are both positive, the threshold is applied where the total payment amount must be less than 1.5 times the starting balance to be considered normal. On the other hand, when *'tot_pmt_amt'* is positive and *'start_posted_bal_amt'* is negative, the payment amount must not exceed 1.1 times the absolute value of the starting balance to be considered as normal. After data was post processed, *confusion_matrix* and *accuracy_score* from sklearn.metrics will be utilized to calculate the classification accuracy and false positive and false negative rate.

What comes along with post-processing is the visualization. The matplotlib.pyplot package is utilized for visualizing the results. A scatter plot is generated to visually differentiate between inliers and outliers based on the selected features. In addition, the scatter plot uses a color map to represent inliers and outliers, with outliers highlighted distinctly to aid in visual inspection.

4. Results:

Graphic visualization is the most straightforward way to present the classification result. As shown in the output graph below, all the transaction records from the dataset are transformed into points within a coordinate system whose x-axis represents the total payment count and y-axis represents the total payment amount. Since both payment amount and payment count are positive, therefore all the points are distributed within the first quadrant. The points are divided into two different types, red points represent the inliers while blue points represent the outliers. On the left lower corner we can see that a huge group of red points are gathered, which is something we are expecting.



Detected Abnormal Transactions:

	external_acct_id	accounting_date	tot_pmt_cnt	tot_pmt_amt	start_posted_bal_amt	outlier
1616	2063863	2024-04-17	2	762.00	363.11	-1
3549	9622	2024-03-27	2	652.84	326.42	-1
5166	21516	2024-03-07	2	12000.00	-10187.19	-1
5253	21516	2024-03-05	2	12000.00	-10187.19	-1
5423	220763	2024-03-04	135	80667.00	8204.86	-1
5523	85891	2024-03-01	18	10633.00	-26.15	-1
5544	220763	2024-03-01	30	17970.00	8204.86	-1
5671	1505466	2024-02-26	88	50613.00	3387.68	-1
5693	1491331	2024-02-26	41	20500.00	0.00	-1
5792	85891	2024-02-23	19	9480.00	-26.15	-1
5805	1442805	2024-02-22	1	50000.00	0.00	-1
5905	711069	2024-02-19	25	11616.67	1816.67	-1
5984	685528	2024-02-19	23	11483.46	2158.46	-1
6095	1505466	2024-02-16	30	15000.00	3387.68	-1
6107	1505466	2024-02-15	42	20887.68	3387.68	-1
6232	1442805	2024-02-13	1	40000.00	0.00	-1
6252	1841819	2024-02-12	85	41657.00	-9060.00	-1
6298	1442805	2024-02-12	1	60000.00	5017.03	-1
6479	1399338	2024-02-08	1	50000.00	-22036.21	-1
6539	912160	2024-02-07	60	30000.00	1018.94	-1
6619	1836122	2024-02-06	2	688.00	307.68	-1
7234	1718470	2024-02-06	2	150.00	70.63	-1
8113	1498674	2024-01-29	7	49000.00	8416.31	-1
8155	1399338	2024-01-29	1	40000.00	-22036.21	-1

The table above shows the abnormal transactions determined by LOF algorithm and they are indeed seemed to be abnormal. Now printing out the accuracy as well as the false positive rate, we may found that the accuracy is acceptable, while the false positive rate is still relatively higher than expectation.

```
Accuracy: 1.00
False Positive Rate: 0.16
False Negative Rate: 0.00
False Positive Transactions:
      external_acct_id accounting_date  tot_pmt_cnt  tot_pmt_amt  start_posted_bal_amt  outlier  threshold_outlier
1616           2063863      2024-04-17            2       762.00                363.11        1                  0
3549              9622      2024-03-27            2       652.84                326.42        1                  0
5792             85891      2024-02-23           19      9480.00                -26.15        1                  0
6619           1836122      2024-02-06            2       688.00                307.68        1                  0
7234           1718470      2024-02-06            2       150.00                 70.63        1                  0
8892            477027      2024-01-15            2       350.00                -63.00        1                  0
8915           1841819      2024-01-12           19      9296.55               2817.22        1                  0
```

5. Conclusion and Reflection:

This project successfully applied the LocalOutlierFactor (LOF) algorithm to identify outliers within a dataset of credit card transaction histories. By focusing on detecting abnormal payment behaviors, the project aimed to enhance the ability to flag potentially fraudulent or erroneous transactions, a critical task in the financial sector. The results demonstrated that the LOF algorithm, coupled with careful post-processing, is a powerful tool for detecting anomalous transactions. The visualization of the detected outliers provided clear insights into the transaction behaviors that were flagged as abnormal, offering valuable information for further investigation.

In fact, the future development capacity of this project is huge in several perspectives. Firstly, the choice of parameter during model tuning is not universal. In this project, by setting *n_neighbors* to 100 and *contamination to 0.01* for a 10,000 total sample space gives me good visualization result. However, if I add up to a sample size of 50,000 and 100,000, the overall performance will be gravely affected, mostly reflecting on the increased false positive rate. Secondly, regarding the false positive points rate, in this project, the false positive rate ends up with 0.16, which is still not in the acceptable range. The false positive rate can be furtherly reduced by more accurate and specific post-processing condition. Overall, the project met its objectives by identifying outliers in the provided dataset. Future work could involve integrating this approach into greater sample size testing, expanding the feature set, and exploring other anomaly detection methods to enhance the accuracy.

6. Python Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score

# Load the datasets
ledger = pd.read_csv('D://Python_Code//Stori//ledger3.csv')

features = ['tot_pmt_cnt', 'tot_pmt_amt', 'start_posted_bal_amt']
X = ledger[features]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
lof = LocalOutlierFactor(n_neighbors=100, novelty=False, contamination=0.01)
outliers = lof.fit_predict(X_scaled)

amt_mean_value = ledger['tot_pmt_amt'].mean()
amt_std_value = ledger['tot_pmt_amt'].std()

for i, (amt, bal_amt) in enumerate(zip(X['tot_pmt_amt'],
X['start_posted_bal_amt'])):
   # Case 1: Positive Start Posted Balance with Positive Total Payment Amount
   if bal_amt > 0 and amt > 0:
       if amt < bal_amt * 1.5:
           outliers[i] = 1  # Set as inlier

   # Case 2: Negative Start Posted Balance with Positive Total Payment Amount
   if bal_amt < 0 and amt > 0:
       if amt <= abs(bal_amt) * 1.1:
           outliers[i] = 1

# Visualization
plt.title("Abnormal Detection with LOF using multiple features")
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=outliers, cmap='coolwarm',
edgecolor='k', label='Inlier')
plt.colorbar(label='Outlier (-1) Inlier (+1)')
```

```
plt.legend()
plt.xlabel(features[0])
plt.ylabel(features[1])
plt.show()

# Adding outlier detection results to the data
ledger['outlier'] = outliers
abnormal_transactions = ledger[ledger['outlier'] == -1]

pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', None)
pd.set_option('display.width', 1000)
print("Detected Abnormal Transactions:")
print(abnormal_transactions[['external_acct_id', 'accounting_date',
'tot_pmt_cnt', 'tot_pmt_amt', 'start_posted_bal_amt', 'outlier']])

######################################################################
# False positive/False Positive rate
n = 2   # n is the parameter used in μ + n * σ
threshold = amt_mean_value + n * amt_std_value

# Classify as outlier (-1 for outlier, 1 for inlier) based on threshold
ledger['threshold_outlier'] = (ledger['tot_pmt_amt'] > threshold).astype(int)
ledger['outlier'] = (ledger['outlier']== -1).astype(int)

accuracy = accuracy_score(ledger['threshold_outlier'], ledger['outlier'])
tp, fp, fn, tn = confusion_matrix(ledger['threshold_outlier'],
ledger['outlier']).ravel()

# Calculate false positive rate (FPR) and false negative rate (FNR)
fpr = fp / (fp + tn)
fnr = fn / (fn + tp)

print(f"Accuracy: {accuracy:.2f}")
print(f"False Positive Rate: {fpr:.2f}")
print(f"False Negative Rate: {fnr:.2f}")
```

```
# Identify and print all false positives
false_positives = ledger[(ledger['threshold_outlier'] == 0) &
(ledger['outlier'] == 1)]

pd.set_option('display.max_columns', None)
pd.set_option('display.max_colwidth', None)
pd.set_option('display.width', 1000)
print("False Positive Transactions:")
print(false_positives[['external_acct_id', 'accounting_date', 'tot_pmt_cnt',
'tot_pmt_amt', 'start_posted_bal_amt', 'outlier', 'threshold_outlier']])
```