HANOI UNIVERSITY

Machine Learning and Applications

UNSUPERVISED LEARNING FOR FINANCIAL FRAUD DETECTION

Teacher: Bui Quoc Khanh

Student: Dao Anh Thanh - 1801040200

November, 2023

Contents

1 Introduction and Literature Review

In today's rapidly evolving digital landscape, the financial sector faces a constant and escalating threat from sophisticated fraudulent activities. Detecting fraudulent transactions has become a paramount concern for financial institutions, as the consequences of such activities can be financially devastating and erode customer trust. Traditional methods of fraud detection often struggle to keep pace with the evolving tactics employed by fraudsters. However, the advent of advanced machine learning techniques has provided a promising avenue for improving the accuracy and efficiency of fraud detection systems.

This report delves into the implementation of one such cutting-edge machine learning algorithm: Isolation Forest. Designed specifically for anomaly detection, Isolation Forest offers a powerful solution to identify fraudulent transactions amidst a sea of legitimate financial data. This innovative approach isolates anomalies, making it particularly adept at detecting outliers, which are indicative of potential fraud. By leveraging the inherent differences between fraudulent and genuine transactions, Isolation Forest can effectively identify irregular patterns that escape traditional rule-based systems.

Throughout this report, I explore the theoretical foundations of Isolation Forest, its underlying mechanisms, and the reasons behind its effectiveness in detecting financial fraud. I will delve into the practical implementation aspects, discussing how to preprocess data, tune hyperparameters, and interpret results. Additionally, I will showcase real-world case studies and performance evaluations, demonstrating the algorithm's efficacy in various financial scenarios.

The objective of this report is to provide financial institutions, data scientists, and researchers with a comprehensive understanding of Isolation Forest's application in fraud detection. By embracing this innovative technique, organizations can bolster their defenses against fraudulent activities, ensuring a more secure financial environment for both businesses and consumers alike.

2 Isolation Forest Methodology

2.1 Rationale for Method Selection

When it comes to addressing classification problems, logistic regression stands out as a widely favored option, as well as decision trees and random forests. However, when it comes to identifying anomalies, the isolation forest algorithm is the preferred choice. This popularity stems from its simplicity and effectiveness in detecting anomalies within extensive datasets. Moreover, it operates as an unsupervised learning algorithm, eliminating the need for labeled data during the training process. This advantage is particularly crucial in applications like fraud detection, where obtaining labeled data is often challenging. Additionally, the isolation forest algorithm excels in detecting anomalies present in both numerical and categorical data, making it ideal for datasets that encompass diverse data types.

— Phan nay can nhac cho vao —

One way to tackle this problem could be to devise some hard-coded criteria for 'normal' transactions that are based on domain knowledge. For instance, we might know that

transactions from account x on day y usually do not exceed a certain amount; so all transactions that do not fulfill this condition could be flagged as anomalies. The drawback of this type of approach is that we need to know in advance what an outlier is going to look like. In most cases it is not feasible, even for domain experts, to anticipate all the forms that an anomaly could assume, since the space of possible criteria is too vast to search manually. Luckily, this is a problem where machine learning can help us out. Instead of having to define certain criteria for outlier detection, we can build a model that learns these criteria (although less explicitly) by training on large amounts of data. In this article we are going to apply such a model, namely an Isolation Forest, and we will use an additional explanatory model that is going to help us make sense of the decisions our anomaly detection model makes. — Het —

2.2 Intuition

Imagine there is a big basket of various fruits like apples, oranges, and bananas. Most of the fruits in the basket are apples, and there are only a few rare fruits like dragon fruits or star fruits. We want to find these rare fruits quickly.

Isolation Forest works a bit like a game where we have a special power to magically separate these rare fruits from the common ones. Here's how it works:

1. **Random Guessing**: We close our eyes and randomly pick a fruit from the basket. Then ask questions like, "Is this fruit red?" or "Is this fruit smaller than a tennis ball?" Based on the answers, we split the fruits into groups.

2. **Isolating Rare Fruits** Because the rare fruits are different, they need fewer questions to separate them from the common fruits. For example, if we ask, "Is this fruit not red?" and we're looking for a dragon fruit, we can quickly find it because most fruits are red (like apples), but dragon fruits are different.

3. **Counting Questions** We keep track of how many questions we asked to find each fruit. The rare fruits, being different, need fewer questions. If we find a fruit with only a few questions, it's likely a rare fruit.

4. **Identifying Rare Fruits** After asking questions and separating fruits, we check which fruits needed the fewest questions. These fruits are the rare ones we were looking for!

In the same way, Isolation Forest uses random questions to separate rare or unusual data points (like the rare fruits) from the common ones. By figuring out which data points need fewer questions to be isolated, Isolation Forest can quickly find the rare and unusual patterns in large sets of data, making it useful for finding things like fraud in a bunch of financial transactions!

2.3 Dataset

There is not much real dataset for this topic since transactions data is a sensitive information. Therefore I decided to choose a synthetic dataset GENERATED by a program called PaySim, which is a mobile money transactions simulator. The simulator was developed based on a sample of real transactions extracted from one month of financial logs from a mobile money service implemented in an African country. The original logs were

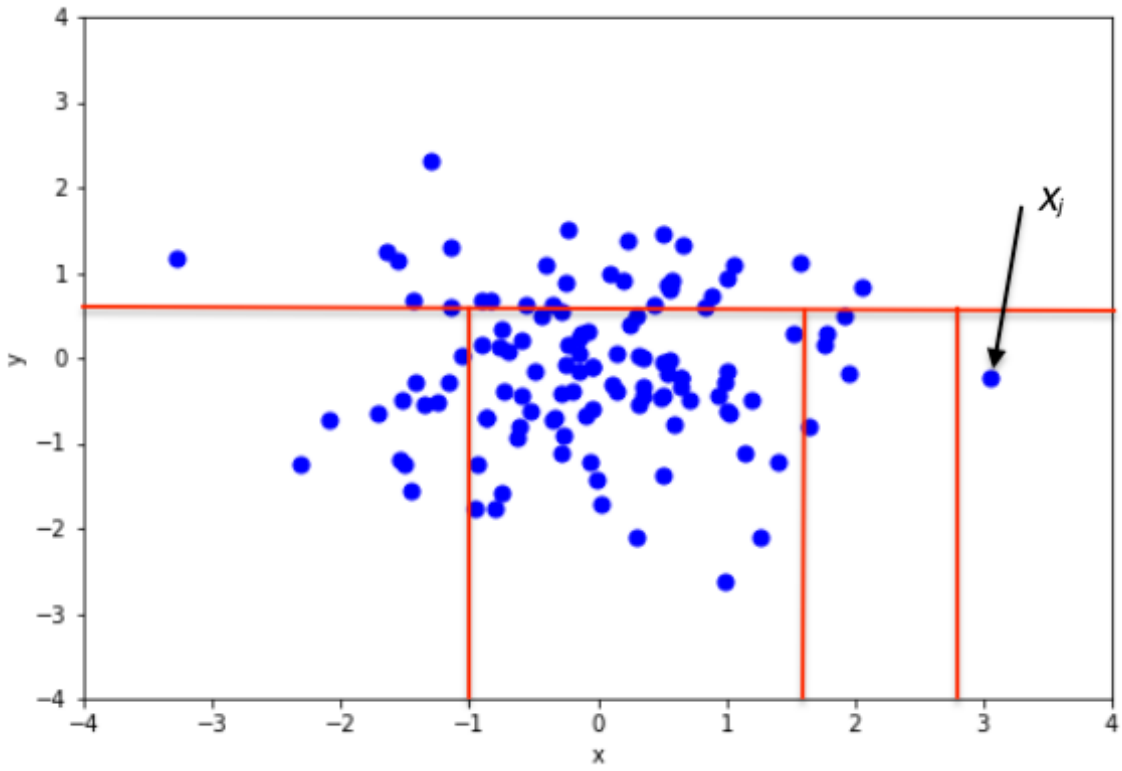

Figure 2.1: An example of isolating an anomalous point in a 2D Gaussian distribution after 4 phrases.

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0
5	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.0	0.0	0	0

Figure 2.2: Fist 06 rows of the dataset

provided by a multinational company, who is the provider of the mobile financial service which is currently running in more than 14 countries all around the world.

Data we have include both numerical and categorical data, and some of them are not useful for our analysis. Therefore, we need to do some data cleaning and feature engineering to make the data more suitable for our analysis.

2.4 Feature Engineering

In order to manage categorical data, it is essential to employ one-hot encoding to 'type' feature. Other attributes seem to be usable because they can easily be used as inputs for the model. However, I need to do some data cleaning to make the data more suitable for our analysis. The first thing I did was to remove the 'nameOrig' and 'nameDest' columns because they are not useful for our analysis. The 'isFlaggedFraud' column is a simple indicator variable for whether the amount transferred in a given transaction exceeds the threshold of 200,000 - also must be removed. After derive the 'step' field to 'hour', the final dataset is shown in Figure 2.3.

	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	changebalanceOrig	changebalanceDest	hour	CASH_IN	CASH_OUT	DEBIT	PAYMENT	TRANSFER
0	9839.64	170136.0	160296.36	0.0	0.0	-9839.64	0.0	1	False	False	False	True	False
1	1864.28	21249.0	19384.72	0.0	0.0	-1864.28	0.0	1	False	False	False	True	False
2	181.00	181.0	0.00	0.0	0.0	-181.00	0.0	1	False	False	False	False	True
3	181.00	181.0	0.00	21182.0	0.0	-181.00	-21182.0	1	False	True	False	False	False
4	11668.14	41554.0	29885.86	0.0	0.0	-11668.14	0.0	1	False	False	False	True	False
5	7817.71	53860.0	46042.29	0.0	0.0	-7817.71	0.0	1	False	False	False	True	False

Figure 2.3: Final dataset

2.5 Implementation

Isolation Forest algorithm is built on the basis of the following assumptions:

- **Fewness** - anomalous samples are rare, constituting only a minority, and they appear in limited numbers within any dataset.
- **Different** - anomalous samples exhibit values or attributes that significantly diverge from those of normal samples.

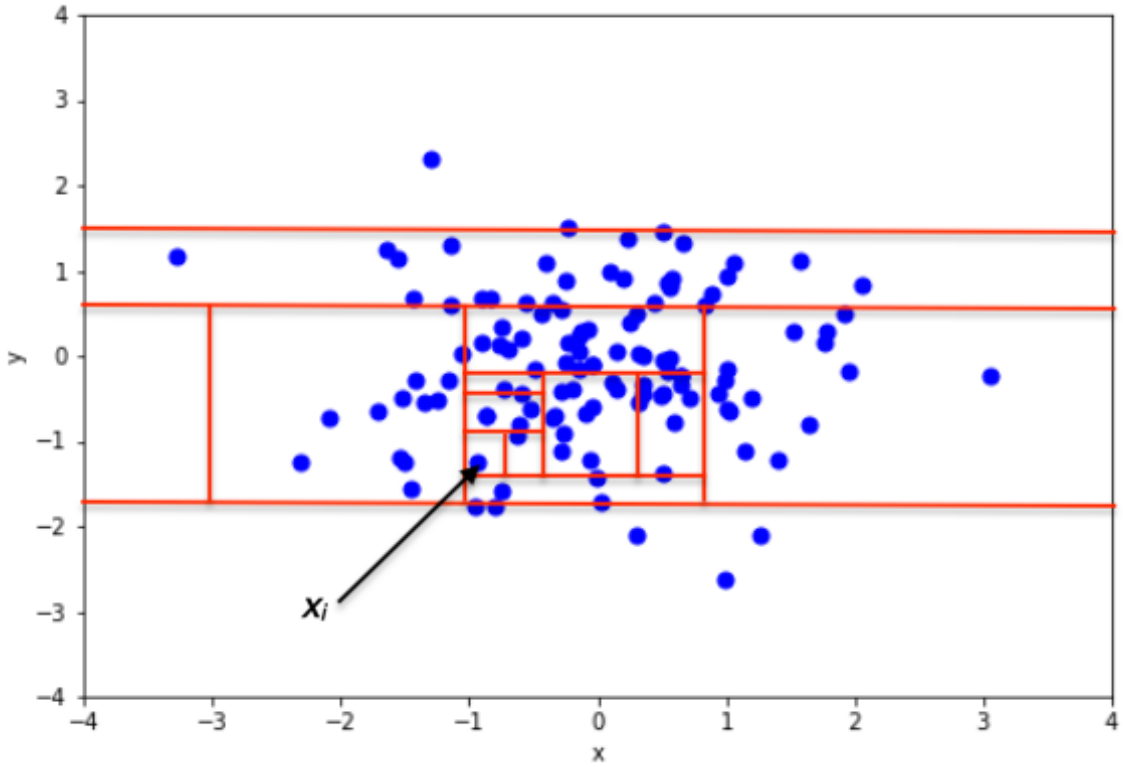

Figure 2.4: Two above assumptions make detecting and isolating anomalous from dataset easier.

There are a several python libraries such as ScikitLearn.IsolationForest, which provides a simple and efficient way to implement the algorithm In submitted code, I used library to implement Isolation Forest algorithm. The library is called **scikit-learn**. The library provides a simple and efficient way to implement Isolation Forest algorithm. However, to better understand the algorithm, I will try to explain the algorithm in my own words.

Given a dataset X with n samples, the algorithm works as follows:

1. Randomly select a feature f from the dataset X.
2. Randomly select a split value s between the maximum and minimum values of feature f.
3. Split the data into two parts: one with values $<= s$ and the other with values $> s$.
4. Repeat steps 1-3 until either the samples are completely isolated or all the values at the current node are identical.

The algorithm iterates through steps 1-4 several times, generating multiple Isolation Trees, which collectively form an Isolation Forest. By observing how Isolation Trees are constructed and understanding the characteristics of anomalous points, it becomes apparent that the majority of anomalies tend to be positioned closer to the tree's root. This is because they are easier to isolate compared to normal points.

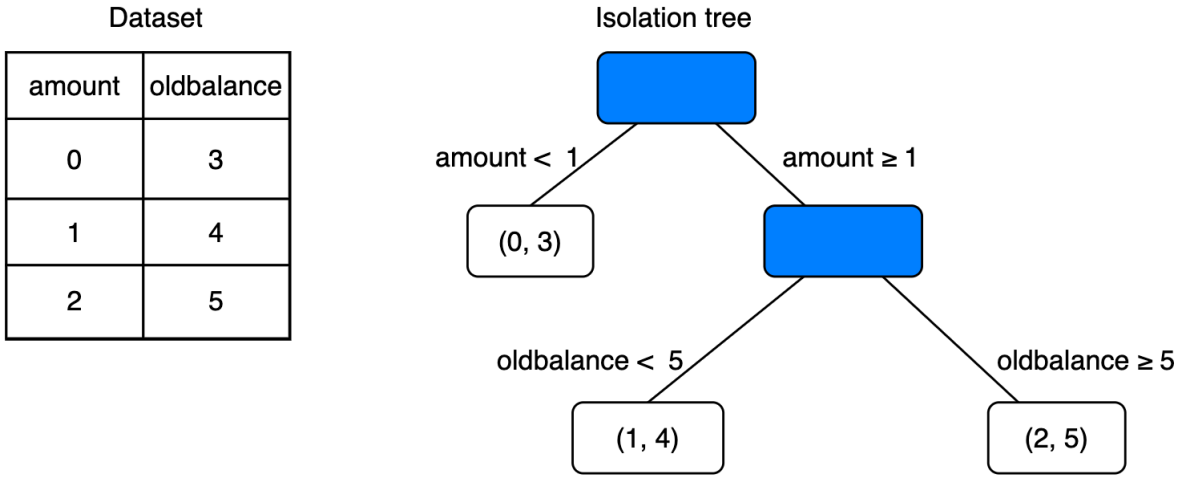

Figure 2.5: Illustration of the Isolation Forest algorithm.

We then need a metric to measure the degree of isolation of each data point. This metric is called the **anomaly score**. The anomaly score is calculated as follows:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}} \tag{1}$$

where

- x is a data point.
- n is the number of data points.
- $E(h(x))$ is the average value of $h(x)$.
- $h(x)$ is the path length of data point x.

- and $c(n)$ is the average path length of unsuccessful search in a binary tree, which is given by:

$$c(n) = \begin{cases} 2\ln(n-1) - \frac{2(n-1)}{n}, & \text{if } n > 2 \\ 1, & \text{if } n = 2 \\ 0, & otherwise \end{cases} \tag{2}$$

  where $H(i)$ is the harmonic number and is approximately equal to $\ln(i) + \mathrm{e}$.

This is the python code for calculating the anomaly score, the completed code can be found in the attached file:

```
def c(size):
    if size > 2:
        return 2 * (np.log(size-1)+0.5772156649) - 2*(size-1)/size
    if size == 2:
        return 1
    return 0

def anomaly_score(X) -> np.ndarray:
    avg_length = self.path_length(X)
    scores = np.array([np.power(2, -l/c(sample_size))for l in avg_length])
    return scores
```

The anomaly score **s(x, n)** is a value between 0 and 1. The closer the score is to 1, the more likely it is that the data point is an anomaly. The anomaly score is then used to determine whether a data point is an anomaly or not. If the score is greater than a predefined threshold, the data point is considered an anomaly. Otherwise, it is considered normal.

3 Experiments and Results

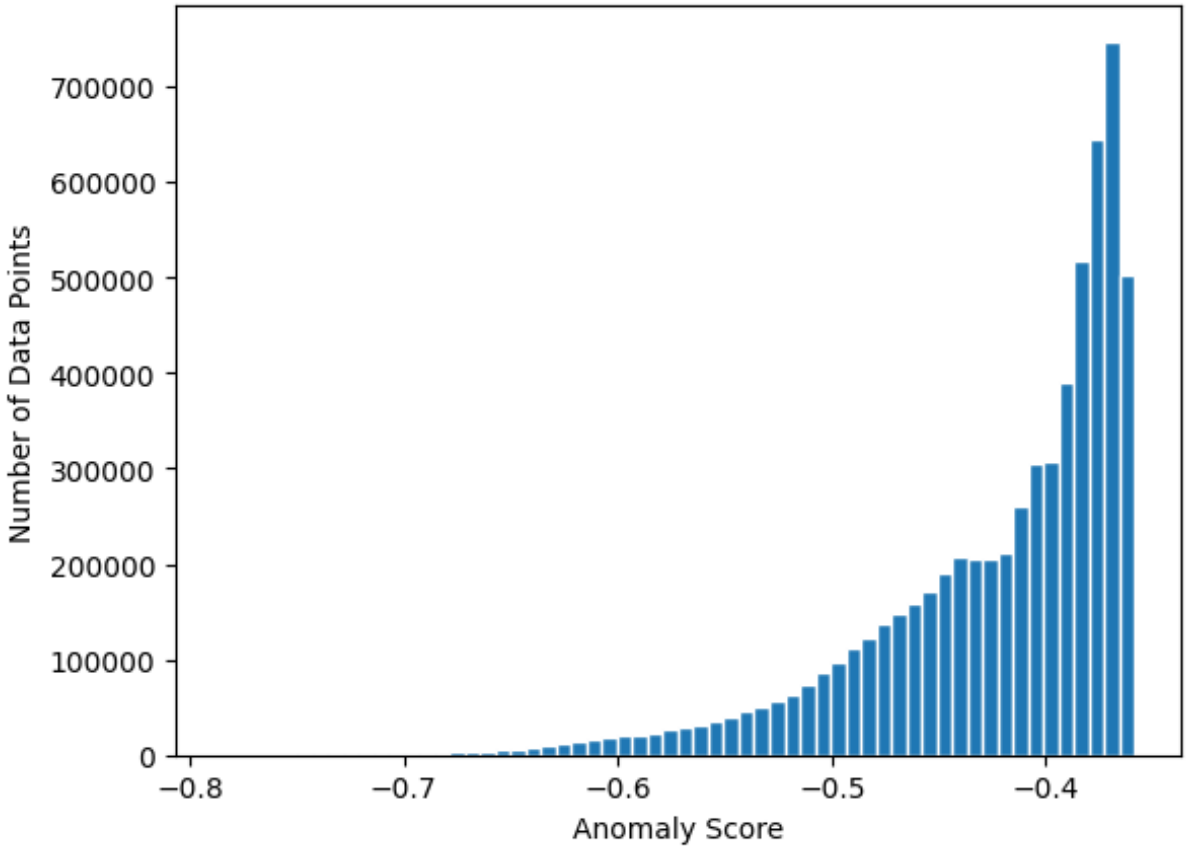

Figure 3.1: Distribution of abnormal scores

After fitting the features and get the distribution of all the anomaly scores, I expect its distribution to be a normal distribution or at least the similar form. Indeed it is, as shown in Figure 3.1