Identity Theft Detection

Yu Qiu

Problem Understand

Nowadays, identity theft has been a big problem in our life. This causes a series of serious consequences, such as credit card fraud.

Hence, the goal of our project is to design a system which can detect the identity theft problems during the online transactions.

Main Steps of the Project

- Find enough dataset
- Feature selection
- Model selection
- Deploy the system

Dataset

I found a dataset which contains transactions made by credit cards in September 2013 by european cardholders. This dataset presents 284,807 transactions that occurred in two days. However, It contains only numeric input variables which are the result of a PCA transformation.

Only this dataset is not enough for the whole project, so we need to look for more useful dataset.

Feature Selection

According to the materials online and my own background knowledge, I conclude several aspects of feature that might be helpful for our project.

- User info: gender, age, ect
- Transaction attribute: country, state, zip, address, ip, auth result, product type, hour of day, amount, ect
- Device fingerprint: device os, browser, app, flash enabled, VPN, VM, ect
- Linkage feature: amount of accounts on the same devices, etc
- Risk rating: geo-region, email domain
- Time based aggregation: amount of transaction by a device in a time period
- Other info: bank issuer, type of card

Model Selection

Before training our model, we need to figure out a problem -- our project is different with common classification job. Since the amount of fraud data is really little compared with normal data. This means that the dataset is very unbalanced. Thus, it is not reasonable to apply supervised classification models directly on our project.

Based on this problem, I give two solutions:

- Apply abnormal detection methods
- Preprocess data before applying supervised model: over sampling, down sampling, synthetic minority over sampling(SMOTE), class weight, etc

Anomaly Detection - Gaussian Distribution

For a given dataset, we can determine the approximate gaussian distribution by using the following parameter estimation:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2$$

And then according to the Gaussian Distribution:

$$p(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

Anomaly Detection - Gaussian Distribution

We can calculate the probability of each data point in a dataset, where xi is a feature:

$$p(x) = p(x_1; \mu_1, \sigma_1^2) p(x_2; \mu_2, \sigma_2^2) \cdots p(x_n; \mu_n, \sigma_n^2)$$

Finally, we can determine whether it is a outlier by:

$$p(x_{test}) < \epsilon, \text{ flag as outlier or anomaly}$$
$$p(x_{text}) \geq \epsilon, \text{ flag as normal or non-anomalous}$$

The largest problem of this method is how to determine the threshold. I use grid search on cross validation to get it.

Anomaly Detection -- Isolation Forest

The main steps of this algorithm includes:

1. randomly select a feature from dataset for each tree
2. select a random split value between the minimum and maximum value of the selected feature
3. iteratively split the node by selecting random value between the minimum and maximum value until all leaf only contain one data point or reach the maximum iterations defined in advance

Anomaly Detection -- Isolation Forest

After the construction of tree, we can find that the data points are more possible to be anomaly if they are more closed to the root. This is very easy to understand. Intuitively, the outlier should be far away from the normal data points, so it is more easy to be partitioned.

Compared with Gaussian Distribution method, Isolation Forest is more suitable to be deployed on distributed system because it is a kind of bagging tree model. However, Isolation Forest cannot fit high-dimension dataset.