Credit Fraud Detection using Anomaly Detection Methods

by Huayu Li

January 12, 2021

Content

- Introduction of Financial Fraud Detection
- Introduction of Anomaly Detection methods: One-class Classification, Isolation Forest and One-class ELM
- Application upon the fraud detection dataset
- Conclusion

Introduction of Financial Fraud Detection

My understanding of Fraud Detection

- Fraud detection is a set of activities undertaken to prevent money or property from being obtained through false pretenses. Fraud detection is applied to many industries such as banking or insurance. In banking, fraud may include forging checks or using stolen credit cards. Other forms of fraud may involve exaggerating losses or causing an accident with the sole intent for the payout.
- All techniques used to detect fraud include the use of: Data Mining, Machine Learning and Pattern Recognition. For traditional machine learning methods, we can regard the fraud detection as the classification problem, given the fraud class a proper weight to do classification; but these methods are really restricted, for the precision scores can be extremely low; although this seems to be common, it may probably lead to recognizing a large percentage of data into fraud. So the one-class classification methods seem to be possible to deal with this problem.

Introduction of Anomaly Detection methods

One-class Classification(One-class SVM)

- This is the most-common One-class classification method, and we can directly use sklearn.svm.OneClassSVM to gain the one-class classification.
- It is quite similar with the normal SVM method, but the optimization functions are quite different. Here we assume the normal data points (the data that is not in the abnormal class) as $\{x_i\}$, then the minimization problem becomes:

$$\min_{R,a} R^2 + C\sum_{i=1}^{n}\xi_i$$

s.t.

$$||x_i - a||^2 \leq R^2 + \xi_i \qquad (i = 1, 2, ..., n)$$

$$\xi_i \geq 0 \qquad (i = 1, 2, ..., n)$$

- For kernel version of One-class SVM, the minimization problem just becomes:

$$\min_{R,a} R^2 + C\sum_{i=1}^{n}\xi_i$$

s.t.

$$K(x_i - a, x_i - a) \leq R^2 + \xi_i \qquad (i = 1, 2, ..., n)$$
$$\xi_i \geq 0 \qquad (i = 1, 2, ..., n)$$

- For a new given data point z, here we just need to judge if $K(z - a, z - a) \leq R^2$; if not, then the given data point is abnormal. It is just similar with normal SVM, using a hyperplane to separate the data points, although here the boundage is a ball with radius R.

Introduction of Anomaly Detection methods

Isolation Forest

- Isolation forest is an unsupervised learning algorithm for anomaly detection that works on the principle of isolating anomalies, instead of the most common techniques of profiling normal points; here in python, we can directly use sklearn.ensemble.IsolationForest to gain this method.
- For each node (N) of the given tree, the basic split process is as following:
  (1) We gain the sub-dataset of this node(X), and the attributes of the dataset(Q).
  (2) For each node in From the attributes (also we can say dimensions) of the dataset, we randomly choose one attribute ($q \in Q$); then we get the minimum and maximum value of this attribute ($x_{min}, x_{max} \in X$)
  (3) Randomly select the point x_{split} from the set $\{x_{min}, x_{max}\}$; then use x_{split} to split the sub-dataset into N_{left}, N_{right} according to the attribute q. Then we have finished this split step.

- Using this method, we can generate a isolation tree, and anomalies will be isolated in only a few steps. Note that for each isolation tree, when we denote the initial data size as ψ, then we actually set the height limit of isolation tree as $ceiling(log_2(\psi))$. Therefore, we can use bootstrap methods to repeat for multiple times to construct a bunch of isolation trees, which is just the isolation forest. Here we assume the Isolation forest as $\{T_1, T_2, ..., T_k\}$.
- For each new given data point x_0, here we calculate the path lengths for the given point in each data point; here this length is measured by the number of edges x_0 traverses the tree T_i from the root node until the traversal is terminated at an external node. We note the lengths as $h_i(x_0), i = 1, 2, ..., k$, and then we calculate the mean of the lengths as $E(h(x_0)) = (\sum_{i=1}^{k} h_i(x_0))/k$

- Assume the new data point is in the dataset X_{test} with n instances, then we can calculate the anomaly score for this point as:

$$s(x_0, n) = 2^{-\frac{E(h(x_0))}{c(n)}}$$

Where

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}$$

and

$$H(i) = log(i) + \gamma$$

where $\gamma \approx 0.577216$ is the Euler constant.

- Then we judge the anomaly as following:
  (1) If $s(x_0, n)$ is really close to 1, then this point can be regard as definitely anomaly.
  (2) Otherwise, is $s(x_0, n)$ is quite smaller than 0.5, then the point is safe to be regarded as normal instance.
  (3) Moreover, if all instances in the given dataset have the anomaly score close to 0.5, then the entire sample does not really have any distinct anomaly.

Introduction of Anomaly Detection methods

One-class Extreme Learning Machine

- Extreme learning machines are feedforward neural networks for classification, regression, clustering, sparse approximation, compression and feature learning with a single layer or multiple layers of hidden nodes, where the parameters of hidden nodes (not just the weights connecting inputs to hidden nodes) need not be tuned.
- Assume the training dataset $\{(x_i, t_i)|x_i \in R^d, t_i \in R^m\}$,$x_i$ input, t_i target output. Then the training becomes the optimizing problem

$$\min_{\beta} ||H\beta - T||_F^2$$

where β is the output weight matrix, and $H = \{g_{ij}(\cdot)\}_{N\times L}$ with $g_{ij}(\cdot) = g(w_j x_i + b_j)$ as the hidden layer output matrix. Here L is the number of hidden nodes, g is the activation function. Note that for regularized version, the function we need to minimize is

$$\frac{C}{2}||H\beta - T||_F^2 + \frac{1}{2}||\beta||_F^2$$

- Note that in one-class classification, only the data of target class is available for model training; then the learning of OC-ELM changes to:

$$\min_{\beta} \frac{C}{2}||H\beta - t||_2^2 + \frac{1}{2}||\beta||_2^2$$

and here t is a vector with all components equal. Besides, we define the error as:

$$\epsilon(x_i) = |\sum_{j=1}^{L} \beta_j g(w_j x_i + b_j) - t|$$

Then we can set a threshold θ, comparing the error and θ to judge whether we judge the target as abnormal. Actually we can rank the errors and set a given quantile as the threshold.

- Note that in one-class classification, only the data of target class is available for model training; then the learning of OC-ELM changes to:

$$\min_{\beta} \frac{C}{2}||H\beta - t||_2^2 + \frac{1}{2}||\beta||_2^2$$

and here t is a vector with all components equal. Besides, we define the error as:

$$\epsilon(x_i) = |\sum_{j=1}^{L} \beta_j g(w_j x_i + b_j) - t|$$

Then we can set a threshold θ, comparing the error and θ to judge whether we judge the target as abnormal. Actually we can rank the errors and set a given quantile as the threshold.

- The One-class ELM is a basic form of neural network upon anomaly detection, and it has many updated versions, for example, the Multilayer One-class ELM (ML-OCELM), the Multilater Kernel One-class ELM (MK-OCELM)(Dai et al., 2019), and maximum correntropy criterion (MCC) based OC-ELM (MC-OCELM)(Cao et al., 2020) and so on.
  Note that the definition of correntropy among two random variables is as following:

$$V(X, Y) = E[\kappa(X, Y)]$$

  where κ is the kernel function; for example, the Gaussian kernel is as following:

$$\kappa(X, Y) = \kappa_\sigma(X - Y) = G(X - Y) = e^{-\frac{(X-Y)^2}{2\sigma^2}}$$

Application upon the fraud detection dataset

- The dataset we use is the Credit Card Fraud Detection dataset from Kaggle: `https://www.kaggle.com/mlg-ulb/creditcardfraud` . The dataset is not concrete, for it mainly contains the 28 principles components of the original data ($V_1, ..., V_{28}$); the other variable contains is the transaction amount. Although we can't find the origin of the dataset, this shows us a possible way: doing PCA and extract the most important principle components may be a possible way to deal with raw dataset.
- The dataset has 284807 rows, and about 0.172% individuals are in fraud class. Here we randomly divide the dataset into training and testing dataset with the proportion 4:1. Then we will try the three different kind of anomaly detection methods.

Application upon the fraud detection dataset

Result for One-class SVM

- For the original dataset, the result is as following:

	Pred_Normal	Pred_Fraud
True_Normal	56718	126
True_Fraud	97	21

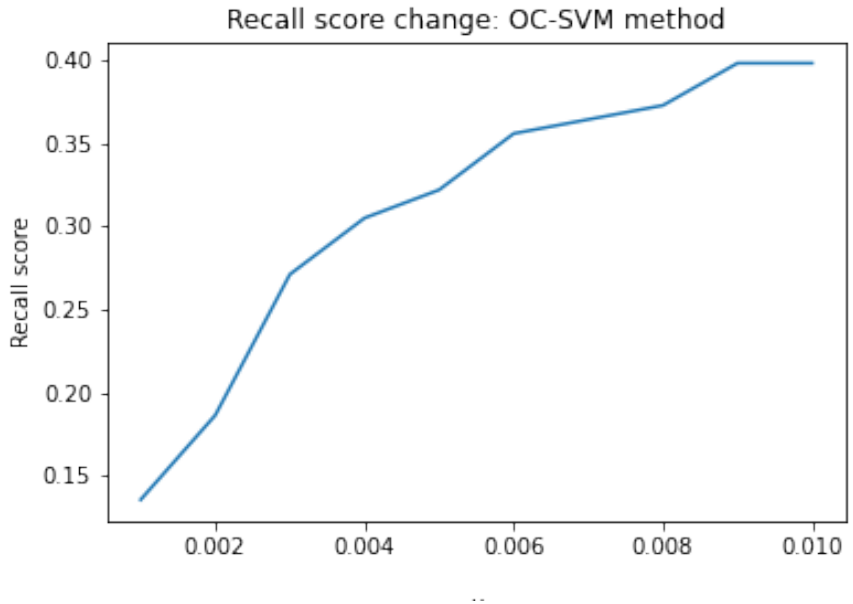
Table: Result of One-class SVM ($kernel = rbf$, $\nu = 0.0172$)

The recall score is only 0.178, which means that only 17.8% fraud individuals are detected, and the performance is not good at all.

Application upon the fraud detection dataset

Result for One-class SVM

- Here we change the value of ν, and detect the change of recall score (Here ν refers to the proportion, so we can set the ν value larger). The result is as following:



- From the plot, we can find out that when the fraud proportion is set to 0.01, about 40% fraud individuals can be discovered; but this is still a really low rate.

Application upon the fraud detection dataset

Result for Isolation Forest

- For the original dataset, the result for Isolation Forest is as following:

	Pred_Normal	Pred_Fraud
True_Normal	56731	113
True_Fraud	87	31

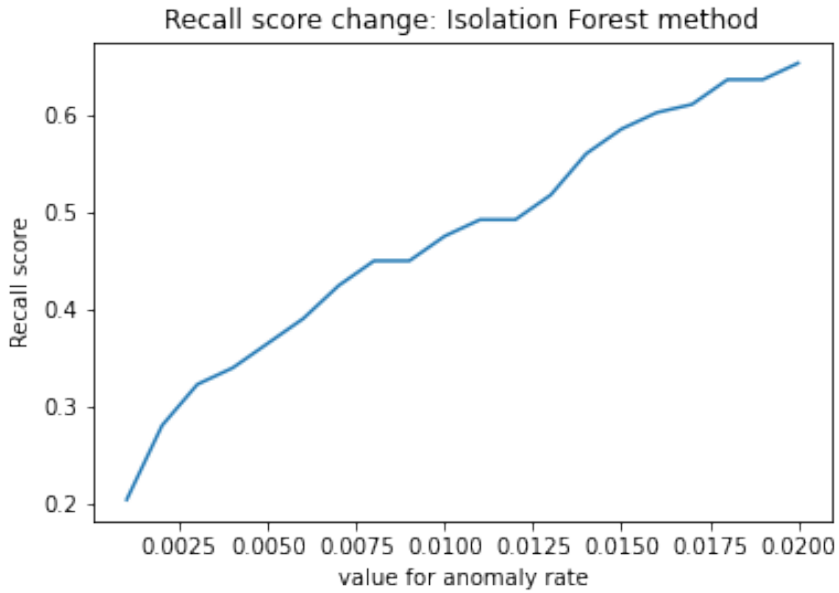
Table: Result of Isolation Forest (n_estimators=50)

The recall score is only 0.2627, which means that only 26.3% fraud individuals are detected, and the performance is not good at all.

Application upon the fraud detection dataset

Result for One-class SVM

- Here we change the value of ν, and detect the change of recall score (Here ν refers to the proportion, so we can set the ν value larger). The result is as following:

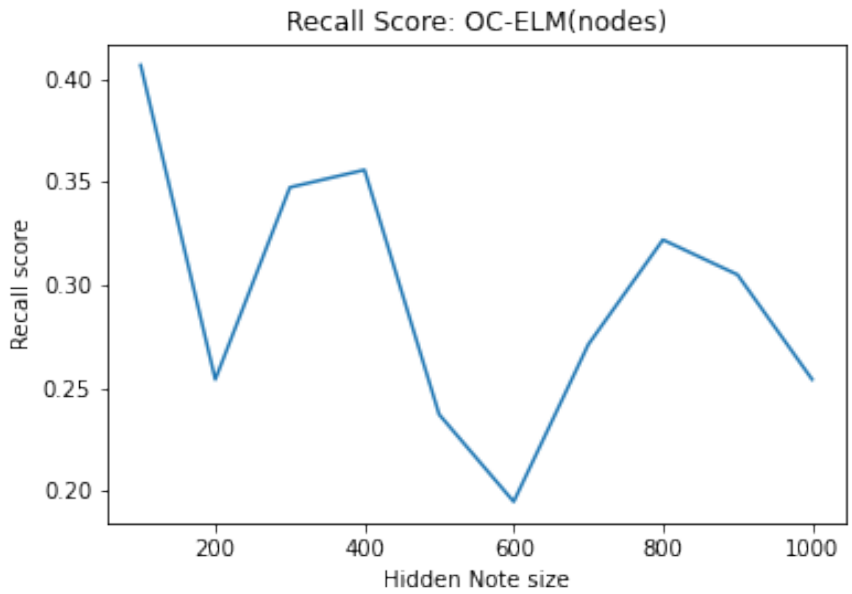


- From the plot, we can find out that when the fraud proportion is set to value between 0.01 and 0.02, about $50\% - 60\%$ fraud individuals can be discovered; this rate is decent, but not good enough.

Application upon the fraud detection dataset

Result for One-class ELM

- For One-class ELM, we only consider the most simple condition; here we change the size of hidden nodes, then we can get the recall rate plot as following:



From the plot, we can find out that the best selection of the size of hidden nodes is 100, and then the recall score is about 0.4068 (here the ν value is 0.00172, implies that 0.172% of the testing data are predicted as fraud).

Application upon the fraud detection dataset

Result for One-class ELM

- When setting the hidden nodes size as 100, then the predicting result is as following:

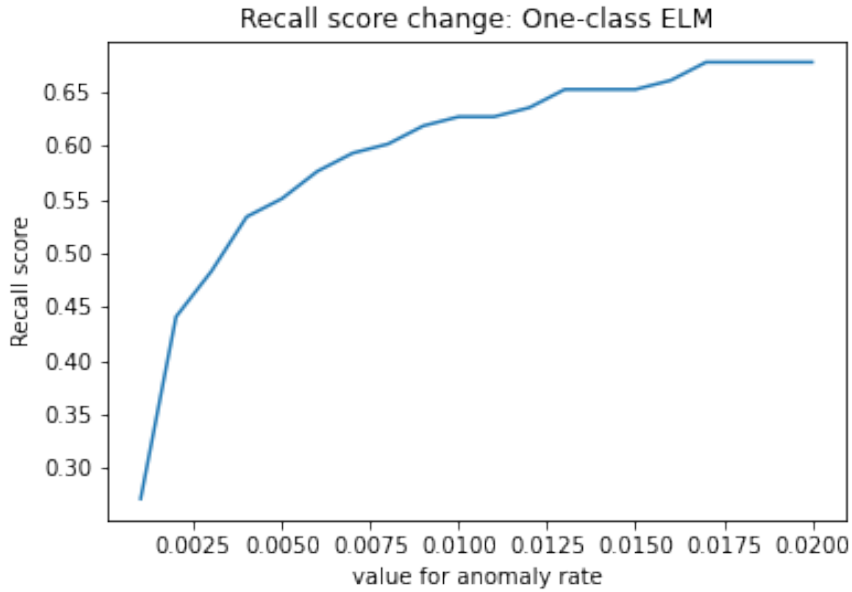
	Pred_Normal	Pred_Fraud
True_Normal	56794	50
True_Fraud	70	48

Table: Result of One-class ELM

- It is obvious that this method performs better than the Isolation forest and One-class SVM. And when we change the value of ν, the recall score will change like the following graph:

Application upon the fraud detection dataset

Result for One-class ELM



- From the plot, we can find out that when the fraud proportion is set to value between 0.01 and 0.02, about $60\% - 65\%$ fraud individuals can be discovered; this rate is similar as the Isolation Forest; but we can also find out that when increasing the proportion of the predicted fraud, the improving rate for One-class ELM will decrease a lot; this implies that the performance of One-class ELM can reach to a stable state in a much smaller change upon anomaly rate.

Application upon the fraud detection dataset

Comparing with traditional machine learning methods

- Here we only consider the Logistic Regression and Random Forest method. After training and testing, the two methods all have the same classification result:

	Pred_Normal	Pred_Fraud
True_Normal	55233	1611
True_Fraud	6	112

Table: Result of Traditional ML methods(Random Forest and Logistic Regression)

- Here we can find out that the traditional methods have the better performance in classification, for the accuracy for the normal and fraud classes can both reach to a high standard (97.16% and 94.92%), but the methods can reach the situation that too many normal individuals are judged as fraud; comparing to the traditional methods, the one-class ELM and Isolation Forest can have a nice trade off between the False Negative proportion and the True

Conclusion

- In this part we have gone through several new machine learning methods for anomaly detection: One-class SVM, Isolation Forest and One-class ELM.
- We use an existing dataset about Fraud Detection from Kaggle, and we applied these new methods upon the dataset; the result shows that the performance of Isolation Forest and One-class ELM methods can both increase a lot when increasing the proportion for predicted fraud, and the One-class ELM method can reach to a nice performance condition faster.

Conclusion

- We also compared these methods with two traditional classification methods(Logistic Regression and Random Forest); the result shows that the traditional methods can have better classification performance, but the anomaly detection methods can prevent from judging too many normal individuals into fraud situation.
- As a result, when we want to make the classification accuracy for fraud really high, we can consider the traditional classification methods; but when we aim to prevent judging too many normal individuals into fraud situation, we can consider One-class ELM or Isolation Forest, and One-class ELM is the better choice.

References

Tax, David Martinus Johannes (2002)

One-class classification: Concept learning in the absence of counter-examples

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2008)

Isolation Forest

2008 Eighth IEEE International Conference on Data Mining pp. 413-422. IEEE

Dai, Haozhen, Jiuwen Cao, Tianlei Wang, Muqing Deng, and Zhixin Yang (2019)

Multilayer one-class extreme learning machine

Neural Networks 115, pp. 11-22

Cao, Jiuwen, Haozhen Dai, Baiying Lei, Chun Yin, Huanqiang Zeng, and Anton Kummert (2020)

Maximum correntropy criterion-based hierarchical one-class classification

IEEE Transactions on Neural Networks and Learning Systems (2020)

Dataset source: `https://www.kaggle.com/mlg-ulb/creditcardfraud`

The End