Automating Anomaly Detection

Orr Striechman (ID. 315241265), Gilad Battat (ID. 206947814)

Tabular Data Science Course, BIU, Match 18, 2022

1 Abstract

When analyzing huge amounts of data, sometimes, detecting anomalies is crucial. For example, detecting credit-card frauds, intrusion attempts on our system, problem that keeps clients away from our e-commerce shop and many more examples.

The process of finding the anomalies isn't easy, partly because one has to manually select the (hopefully) optimal model and adjust its hyper-parameters (CASH (choose algorithm and Hyper-Parameters) problem). And there is not a universal best algorithm/method to solve that problem, yet. The goal of our project is to help solving the **model-selection** problem, and trying to automate it. Our original idea was to analyze and find characteristics of the datasets, Then choosing according to them and our analysis, the most suitable algorithm to use in our specific problem.

We thought about many meta-features we could use to help us identify our dataset, and then during our research on the internet we found some great state-of-the-art ideas which did exactly that. Especially metaOD[5], which is a fascinating paper of how to automate unsupervised anomaly detection tasks (we will explain more about this solution later), that was a great improvement on our way to solution but still it doesn't solve the whole problem, so we tried to add some more functionality to metaOD, and make an automation alternative for datasets where metaOD doesn't work at.

2 Problem Description

Many Data Science pipelines need to deploy some anomaly detection techniques. And the fact that there isn't an universal best anomaly detection algorithm, means that this process, And especially the model-selection problem, is currently done **manually and not optimally**.

The data scientist have to choose manually the set of anomaly detection algorithms in order to achieve its goals. Some of the algorithms are based on statistics and some on Machine Learning, we will focus on the latter.

Most of the algorithms for anomaly detection are unsupervised due to the nature of the problem, since many times there are not enough samples from the anomalous category in order to train a supervised model.

Among the many algorithms for anomaly detection you can find: Isolation Forest, KNN, LOF, OCSVM, SVM and AutoEncoders. all of these models are not based on Neural Networks except from AutoEncoders which are a special type of Neural Nets which are suitable for anomaly detection using an unsupervised learning. Further read on how to build a Deep AutoEncoder for Outlier Detection could be found in the following article [6]. SVM and KNN are known for their rule in supervised learning, but both of them has an unsupervised versions, with SVM it's called OCSVM - One Class Support Vector Machine [3] which is a model that is designed specifically for outlier detection. Isolation Forest detects anomalies by selecting random features and random value splits and then splitting the dataset by those choices, then repeating those separations until all the data is separated, the idea here is that anomalies are easier to separate and that's how we can detect them using this approach. LOF - Local Outlier Factor is a model that assigns each sample an anomaly score based on its distance to its nearest neighbors and the density of neighbors that are close to him.

After choosing the model, a data scientist has to adjust the hyper-parameters of the model. Our goal is to automate this process, but we came to a realization that the model choosing problem is the one which needs the most automation, since there are many good known algorithms for parameter-tuning.

3 Solution Overview

3.1 Internet searching and finding pyod and MetaOD

Before implementing our own ideas, we wanted to make sure we exhausted the internet search space. So we looked at lots of papers on outlier and anomaly detection. After some time, we found pyod[4], which is a python library that contains many anomaly detection algorithms, that was a good starting point because it contains many implementations of algorithms and thus give the programmers the ability to use a variety of models with an easy to use library, but still it doesn't automate anything by itself.

So after some more research, we found an article that presents a package called metaOD that can perform automation for OD (outlier-detection). So first we learned about what they did, which I will explain briefly here, and then we added some improvement.

They used a meta-learning based approach (learning about learning) to OD that selects an effective model (detector and its associated hyper-parameters) to be employed on a new detection task.

In summary (more details can be found in their article), they used 200 meta-features from each dataset (including for example: number of samples, number of features, moments, statistical features, and more). Also they took a fixed number of M models, and then in the offline learning stage they took many datasets and run all the M models on them, and stored the results. Now after that, in the **online learning stage**, they took a new online dataset, apply the gen-meta-features function on it, used collaborative filtering, to see which of the offline datasets our new online dataset is the most similar to, and then choose the model that worked the best on those offline datasets on the new online dataset. And they proved it worked well and improves the results (better than any "dumb" selection of 1/combination of some models).

So that's great start for our project but still, it suffers from two basic problems:

1. It doesn't find the optimal hyper-parameters (because there is fixed number

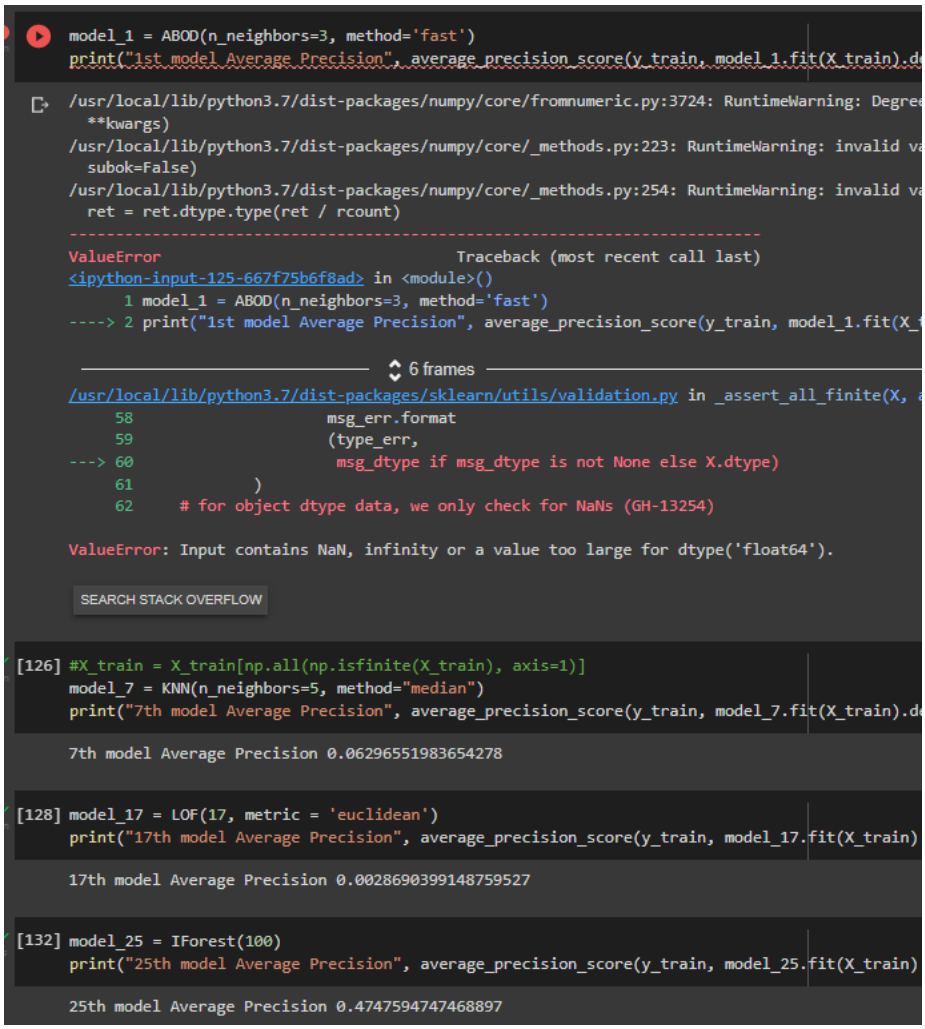
of M models, it only finds the best one of them).

2. It only works for the unsupervised case.

3.2 Extending metaOD

At first we thought to tackle those both problems, and extend metaOD to run on all datasets, and also, to try find **better Hyper-Parameters** configurations than metaOD.

We started by running metaOD against our Fraud Detection dataset. And it did **terribly bad**, the most recommended algorithm crashed (as seen in the picture below). The top 20 algorithms had horrible performance, and the 25th algorithm had much better performance than them.



Than we run it on the pyod generated dataset, and it did great. (results close to 1, the recommended models got better results, as expected).

```
[151] selected_models

      array(['LODA (5, 150)', 'LODA (5, 100)', 'LODA (15, 20)', 'LODA (15, 10)',
             'LODA (25, 200)', 'LODA (5, 40)', 'Iforest (10, 0.5)',
             'Iforest (20, 0.8)', 'LODA (25, 150)', 'LODA (30, 10)',
             'LODA (10, 50)', 'LODA (25, 50)', 'LODA (20, 40)',
             'Iforest (30, 0.1)', 'LODA (10, 20)', 'LODA (20, 200)',
             'Iforest (10, 0.9)', 'LODA (5, 20)', 'LODA (10, 10)',
             'Iforest (20, 0.4)'], dtype='<U22')

model_1 = LODA(n_bins= 5, n_random_cuts = 150)
print("1st model Average Precision", average_precision_score(y_train, model_1.fit(X_train).decision_scores_))

      1st model Average Precision 1.0

[147] model_7 = IForest(n_estimators=20, max_features=0.8)
      print("7th model Average Precision", average_precision_score(y_train, model_7.fit(X_train).decision_scores_))

      7th model Average Precision 0.9999999999999999

[150] model_16 = IForest(n_estimators=10, max_features=0.9)
      print("16th model Average Precision", average_precision_score(y_train, model_16.fit(X_train).decision_scores_))

      16th model Average Precision 0.9988095238095238

[140] model_18 = LODA(n_bins= 10, n_random_cuts = 10)
      print("18th model Average Precision", average_precision_score(y_train, model_18.fit(X_train).decision_scores_))

      18th model Average Precision 0.9938679245283019
```

Therefore, we did the following things:

For datasets that the metaOD don't work well, we manually runs the models that metaOD uses on them and select the best ones.

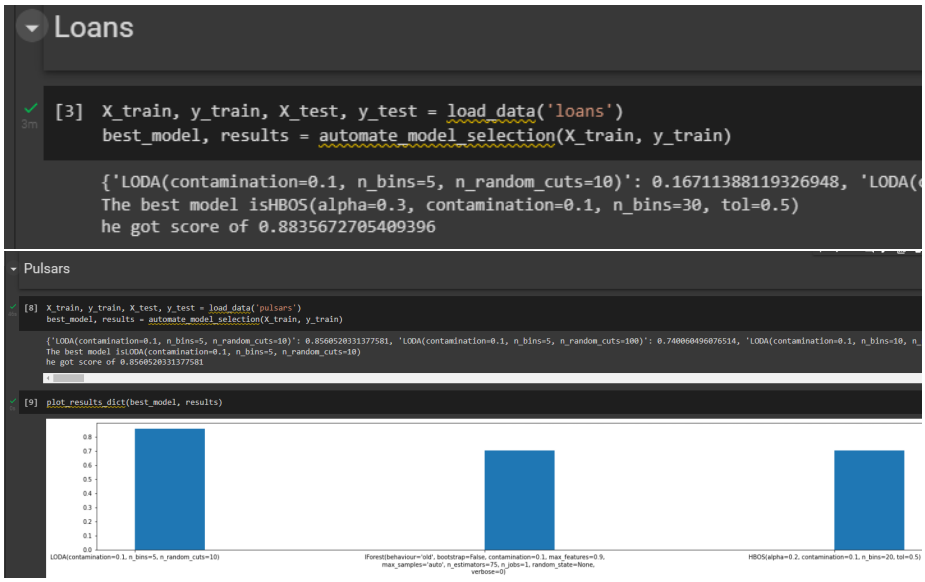
For datasets that metaOD works well, we output first the best algorithm by metaOD, then we try to improve his hyper-parameters even more (we try to improve the hyper-parameters configuration).

That's an improvement both over metaOD by itself, and also over using any specific algorithm by itself.

4 Experimental evaluation

Since the code runs and usually output a different algorithm as the best algorithm. That means we made an improvement over using any single algorithm.

For example the pulsar's dataset best algorithm, is different than the loans dataset one. As we can see in the two images below.



At last, we also see the potential in our metaOD improvement, because our algorithm selected one different (that should be better) than the one metaOD selected. Since both got score of 1.0 that doesn't mean it's better, but in harder datasets where the score won't be 1.0, our algorithm might help doing better. Also it's up to the data scientists to add more complexity to our search (in the expense of longer run time). Our code is flexible enough to do it.

5 Related Work

We looked especially on metaOD and pyOD, as explained above and we've used both tools extensively. We gathered useful information from an article that does a systematic review of anomaly detection through 290 research articles. [2] Another useful article research autoML for anomaly detection [1]. Also we learned about anomaly detection algorithms and ideas from countless more papers and websites, some of them included in the reference.

6 Conclusion

Automation in anomaly detection is a growing field, with some new state-of-the-art researches done in the recent years. Still there is a lot of work to be done, our work might be usable for a Data Scientist that just want a tool to run an anomaly detection algorithm (as none is exists to our knowledge), and also in thinking of how to improving further metaOD, but still there is a lot of research to be done (especially how to derive meta-features from general datasets and make metaOD work on them as well).

References

[1] Maroua Bahri, Flavia Salutari, Andrian Putina, and Mauro Sozio. Automl: state of the art with a focus on anomaly detection, challenges, and research directions. *International Journal of Data Science and Analytics*, 2022.

[2] Ali Bou Nassif, Manar Abu Talib, Qassim Nasir, and Fatima Mohamad Dakalbab. Machine learning for anomaly detection: A systematic review, 2021.

[3] OCSVM. sklearn.svm.oneclasssvm — scikit-learn 1.0.2 documentation.

[4] pyOD. All models — pyod 0.9.8 documentation.

[5] Yue Zhao, Ryan A. Rossi, and Leman Akoglu. Automating outlier detection via meta-learning. 9 2020.

[6] Chong Zhou and Randy C. Paffenroth. Anomaly detection with robust deep autoencoders. volume Part F129685, pages 665–674. Association for Computing Machinery, 8 2017.