A Demonstration of AutoOD: A Self-Tuning Anomaly Detection System

Dennis Hofmann
Worcester Polytechnic Institute
dmhofmann@wpi.edu

Peter VanNostrand
Worcester Polytechnic Institute
pvannostrand@wpi.edu

Huayi Zhang
Worcester Polytechnic Institute
hzhang4@wpi.edu

Yizhou Yan
Meta
yizhouyan@fb.com

Lei Cao
Massachusetts Institute of Technology
lcao@csail.mit.edu

Samuel Madden
Massachusetts Institute of Technology
madden@csail.mit.edu

Elke Rundensteiner
Worcester Polytechnic Institute
rundenst@wpi.edu

ABSTRACT

Anomaly detection is a critical task in applications like preventing financial fraud, system malfunctions, and cybersecurity attacks. While previous research has offered a plethora of anomaly detection algorithms, effective anomaly detection remains challenging for users due to the tedious manual tuning process. Currently, model developers must determine which of these numerous algorithms is best suited for their particular domain and then must tune many parameters by hand to make the chosen algorithm perform well. This demonstration showcases AutoOD, the first unsupervised self-tuning anomaly detection system which frees users from this tedious manual tuning process. AutoOD outperforms the best unsupervised anomaly detection methods it deploys, with its performance similar to those of supervised anomaly classification models, yet without requiring ground truth labels. Our easy-to-use visual interface allows users to gain insights into AutoOD's self-tuning process and explore the underlying patterns within their datasets.

PVLDB Reference Format:
Dennis Hofmann, Peter VanNostrand, Huayi Zhang, Yizhou Yan, Lei Cao, Samuel Madden, and Elke Rundensteiner. A Demonstration of AutoOD: A Self-Tuning Anomaly Detection System. PVLDB, 15(12): XXX-XXX, 2022.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/dhofmann34/AutoOD_Demo.

1 INTRODUCTION

Background. Anomaly detection aims to identify objects that do not conform to an expected standard of behavior and may differ significantly from the remainder of the data. Due to its wide applicability, anomaly detection is a critical component of analytic systems in domains such as financial fraud detection, system malfunction diagnosis, and cybersecurity attack prevention. Despite this ubiquity and importance, in practice, anomaly detection remains challenging due to the scarcity of available labeled anomalies. Anomalous instances are by definition rare and may require expert analysis to properly identify. This makes it impractical to label a substantial number of anomalous samples required for supervised learning. For this reason, a wide variety of unsupervised anomaly detection methods have been devised that aim to leverage data characteristics to identify anomalies with a reasonable degree of accuracy, including statistics-based methods [2, 3], density-based methods [4, 11], and nearest neighbor-based methods [5, 14].

Challenges. A critical challenge in applying anomaly detection is how to choose the most effective solution from a set of available techniques and tune its parameters [1]. Since an anomaly detection method that works well on one dataset might yield poor results on another, users often have to manually select a method appropriate to the given task. Moreover, the performance of unsupervised anomaly detection methods can be very sensitive to the proper selection of hyper-parameters, with these values varying significantly between datasets. Consequently, analysts often need to iteratively rerun their methods with different combinations of hyper-parameters in a time-consuming and computationally expensive process. These factors make the method selection and parameter tuning process tedious as it can be difficult for a user to discriminate between techniques that are poorly suited for their application and those which simply need additional hyper-parameter tuning.

Automatic Machine Learning (AutoML) [7, 8, 13] techniques have been proposed as a potential solution to automate this tuning process. However, these techniques require a sufficient number of labeled instances to evaluate the accuracy of results and to search for optimal models and hyper-parameters. While AutoML has been proven successful for some classification tasks, the scarcity of high-quality labeled data for anomaly detection makes AutoML approaches impractical in many cases.

Proposed Solution. We have thus developed AutoOD, a *self-tuning* anomaly detection system, that addresses the above challenges of tedious method selection and hyper-parameter tuning without requiring access to human-supplied ground truth labels.

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:XX.XX/XXX.XX

The core idea is that rather than carefully selecting an appropriate anomaly detection method for a given task and then tuning its parameters, AutoOD turns the unsupervised problem into a supervised one. AutoOD utilizes an ensemble of unsupervised anomaly detectors with varying hyper-parameters to *automatically* produce high-quality labels. This is achieved by discovering objects from the input data that can reliably be detected as an anomaly or inlier. Using these automatically produced labels, AutoOD then trains a *supervised classification* model. The latter then is applied to the remaining (unlabeled) objects to infer their status, producing the final anomaly detection results. In this way, AutoOD leverages supervised classification to achieve high accuracy in anomaly detection while no longer having to rely on domain experts to manually supply labels as required for supervised or AutoML approaches [7, 8, 13]. As we will showcase in this demonstration, AutoOD consistently outperforms the best unsupervised anomaly detector selected from hundreds of other detectors. Further, its accuracy is comparable to supervised anomaly classification models trained with actual ground truth labels.

For this demonstration, we have designed a rich AutoOD visual interface that enables users to gain insights into the system's self-tuning process. The interface is composed of multiple different views that the user can interact with to explore the inlier or anomaly status of objects inferred by AutoOD. The user can also inspect the performance of specific unsupervised anomaly detectors to further build their understanding of AutoOD's training process and performance. Our demonstration visualizes the different stages of this *self-tuning* process so that the users can gain trust in the system and its final predictions.

2 THE AUTOOD SYSTEM

AutoOD is composed of seven major components (Figure 1). Next, we examine the core idea of each component and briefly highlight the technical contributions.

(1) System Configurator is where the user begins interactions with AutoOD. The System Configurator takes in the input dataset, the expected range representing the anomaly percentage of the dataset, and the desired anomaly detection methods provided by the user. These inputs are inspected for validity and stored in the AutoOD database to be used later in the training process.

(2) Unsupervised Detector Generator leverages a rich library of diverse unsupervised anomaly detectors, including but not limited to Density-based methods [4, 11], Isolation Forest (IF) [10], K-Nearest Neighbors-based methods [9, 12], and statistical-based anomaly detection [1]. Users can add additional algorithms by instantiating the detector template with the corresponding name, algorithm implementation, its parameters, and typical parameter ranges. Or, they can remove the default algorithms that do not fit their applications. For each detection method, AutoOD selects its hyper-parameter configurations from a reasonable parameter range maintained by our system. AutoOD instantiates several detectors of each selected method type with hyper-parameter values selected from within this range. This results in an ensemble consisting of multiple diverse anomaly detection methods with many different combinations of hyper-parameter configurations. Each detector is deployed in parallel to detect anomalies in the dataset of interest.

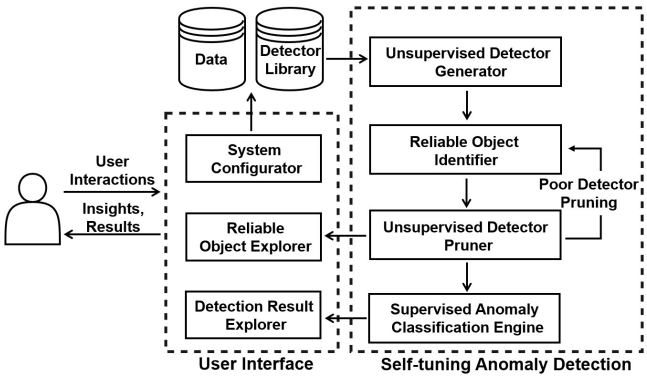

Figure 1: AutoOD Architecture

(3) Reliable Object Identifier. The number and quality of the automatically produced labels by the previously mentioned instantiated detectors are critical to AutoOD's effectiveness. To this end, we design several methods for the identification of objects with reliable labels. For example, one of our core identification methods starts with discovering a small but reliable set of labels based on a strong consensus of *all* the deployed unsupervised anomaly detectors. Utilizing the anomaly detection results from the unsupervised detectors, the Reliable Object Identifier separates the input data into two groups: reliable objects and unsure objects. Where each group may contain inliers and anomalies. Intuitively, an object is considered reliable if all detectors agree on its label. The remaining non-reliable objects are added to the unsure set.

(4) Unsupervised Detector Pruner. Given the pseudo-labeled reliable object set, AutoOD leverages these objects to identify any truly poor performing detectors using various machine learning strategies. After pruning the poor performing detectors from the ensemble, AutoOD gets more reliable objects in the reliable object set. This is because the smaller set of remaining detectors are more likely to agree on an object's label, thus placing that object into the reliable object set. This process repeats iteratively until the reliable object set converges into a stable set. At this point, the pruning process is complete and the detector ensemble has been reduced to a set of well-tuned methods.

(5) Supervised Anomaly Classification Engine. Next, AutoOD leverages supervised classification techniques to improve its anomaly detection capabilities beyond those of the purely unsupervised ensemble. AutoOD treats the pseudo-labeled reliable objects described above as a labeled training set to train a supervised classification model. Examples of such models include Support Vector Machines (SVM), Logistic regression, and Random Forest. In this demonstration, we use SVM as the classification method, as it, while simple, shows superior performance in anomaly detection without requiring hyper-parameter tuning. The resulting trained classification model is used to predict the final labels for all objects that had remained in the unsure object set. To resolve the imbalance of anomaly and inlier labels and reduce the time complexity of the training process, we chunk the inlier labeled dataset into multiple partitions, train multiple models in parallel using multi-processors, and combine the results. In this training process, each processor

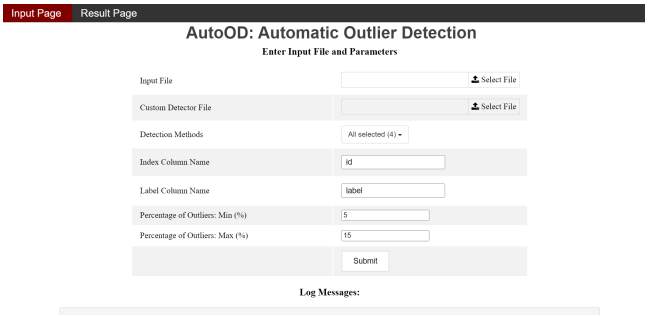

Figure 2: Input Interface. Users can upload data, provide their own anomaly detection methods, specify the column of labels, and customize the expected percentage range of anomalies in their dataset.

uses all anomaly labels (given there are typically a lot fewer anomalies than inliers). Doing so allows AutoOD to perform automatic tuning, yet with only a light computational overhead.

(6) Detection Result Explorer. AutoOD features an interactive visualizer that enables users to explore their dataset throughout the training process. This way, users can gain insights into which objects have been identified as anomalies by which detectors. The Detection Result Explorer uses t-SNE to reduce the dimensionality of the input data, which is needed to plot the data into a 2-dimensional display. AutoOD then color-codes the objects based on the final classification result. Doing so, allows users to view the natural grouping of their data in relation to their anomaly detection status – empowering them to understand how AutoOD operates. The Detection Results Explorer also includes an object inspection tool that allows users to examine the anomaly detection scores of each detector for that object. This can help in identifying methods that may be poorly suited to anomaly detection on the given dataset and thus should be removed from the ensemble.

(7) Reliable Object Explorer. As AutoOD's classification performance depends heavily on the iterative pruning, achieved in collaboration between the Reliable Object Identifier and the Detector Pruner, it is vital to visually display this information. The Reliable Object Explorer visually depicts changes to the reliable object set at each iteration as detector pruning progresses. This allows users to examine the reliable object set as it grows over the iterations to understand AutoOD's poor detector pruning process.

3 DEMONSTRATION WALKTHROUGH

Spambase Benchmark Dataset. We demonstrate AutoOD using various use cases, including the anomaly detection benchmark dataset Spambase [6]. This dataset consists of email data where the goal is to detect anomalous instances which could be spam emails. Spambase consists of 4,601 instances and 57 attributes.

AutoOD Configuration. For this demonstration, a data scientist, henceforth referred to as Jo, aims to find anomalous emails such as unwanted marketing emails or phishing attempts at their company. Jo starts interactions via the AutoOD interface shown in Figure 2. Jo will be asked to upload their dataset and identify the columns corresponding to instance id and labels. By default, AutoOD uses all the anomaly detection methods available in its built-in library, but also allows Jo to customize the selection of methods to be used for detection. To guide the tuning of AutoOD, Jo is also asked to enter the expected percentage range of anomalies in the dataset. This allows the user to leverage their domain knowledge to aid in the selection of hyper-parameters to experiment with, if so desired. Otherwise, the default range maintained by the system will be deployed. Upon clicking submit, AutoOD instantiates different detectors with diverse hyper-parameters for each detector type selected by Jo. The system then proceeds with the training process described in Section 2.

Reliable Object Purification. Once AutoOD completes its run, Jo will be greeted via AutoOD 's Data Analytics Display seen in Figure 3. This display allows the user to interact with the Detection Result Explorer (Figure 3A) on the left of the screen and the Reliable Object Explorer (Figure 3B) on the bottom of the screen. The first interaction point is the data scatterplot in Figure 3A which displays the t-SNE projected data points with identified inliers color-coded in blue and anomalies in yellow. Here Jo can explore the model's performance by zooming in on regions of the data to examine points in relation to their neighbors and filter based on ground truth labels, predicted results, and/or correct predictions.

Assume our data scientist Jo is curious as to why the inlier point p_1 in the scatterplot at the top of Figure 3A is surrounded by anomalies. To investigate this, Jo uses the Detection Result Explorer to select two points of interest to drill down into more specific information about those instances for comparison. Here we picture the user selecting p_1 and its nearest anomaly p_2 both seen towards the top of the scatterplot in Figure 3. After selecting these points, two bar charts appear on the right of the screen (Figure 3D) providing the anomaly scores for each unsupervised detector along with a table of the instances' attributes. The height of each bar represents the anomaly score from each of the anomaly detectors, while the color represents the detectors' overall prediction (blue denotes inlier, yellow anomaly). Leveraging these charts for points surrounding p_1 (such as p_2) and their domain knowledge of typical inlier attributes, Jo can build an understanding as to why AutoOD predicted p_1 to be an inlier while the surrounding points were classified as anomalies. By doing so, not only does Jo gain insights into AutoOD but they also gain trust in the system's training process.

Poor Detector Pruning. While Jo is exploring AutoOD's results, they may determine that one or more detectors exhibit consistent poor performance, meaning the detectors are responsible for many mislabeled instances. For example, in Figure 3D, Isolation Forest (IF) provided a low anomaly score for both points p_1 and p_2 and mislabeled p_2. To examine the impact of each detector on the reliable object set, and therefore the overall model's performance, Jo can adjust the Reliable Object Explorer seen in Figure 3B.

This component keeps a log of the changes to the reliable object set allowing Jo to step through the iterations of AutoOD's training process at their own speed. To visually depict this information, AutoOD updates the scatterplot to show the state of the reliable object set after each training iteration. By moving the slider depicted at the bottom of Figure 3B through each iteration, Jo can watch the reliable object set change. At any time Jo can select a point to view the contribution of each detector to its predicted status. If the Isolation Forest detector is the source of many mislabeled

Figure 3: Data Analytics Display: Demonstrates AutoOD's Detection Result Explorer (A) and Reliable Object Explorer (B). Users can filter based on metrics provided (C) and interact with points by hovering over them to view summary statistics. A click on a point will provide that respective points anomaly score for each detector and attributes (D).

objects, we would expect to see the reliable object set growing across iterations to include several mislabeled points with the Isolation Forest contributing strongly to the status of these objects. Once Jo has identified a poor-performing detector, they hit the rerun button at the top of Figure 3. This will prompt Jo to select which detectors they want to rerun with. Here, Jo removes the IF from the set of selected methods. After this, AutoOD will start training again this time without the IF detector.

4 CONCLUSION

AutoOD is a self-tuning anomaly detection system that combines the benefits of unsupervised anomaly detection and supervised classification. AutoOD addresses the challenges of the lack of labels, method selection, and tedious hyper-parameter tuning by automatically producing a set of high-quality labels that reliably capture key differences between anomalies and inliers. AutoOD provides rich visual views which allow the user to interactively gain insights into how AutoOD operates, build an understanding of patterns within their dataset, and adjust the detection methods to maximize the performance of the system for their dataset.

ACKNOWLEDGMENTS

This research was supported in part by NSF under grants IIS-1910880, CSSI-2103832, CNS-1852498, NRT-HDR-1815866 and by the U.S. Dept. of Education under grant P200A180088. We also thank all members of the DAISY group for their input on this research.

REFERENCES

[1] Charu C. Aggarwal. 2017. *Outlier Analysis: Second Edition.* Springer.

[2] Peter Bailis, Edward Gan, Samuel Madden, Deepak Narayanan, Kexin Rong, and Sahaana Suri. 2017. Macrobase: Prioritizing attention in fast data. In *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 541–556.

[3] Vic Barnett, Toby Lewis, et al. 1994. *Outliers in statistical data*. Vol. 3. Wiley New York.

[4] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: Identifying Density-Based Local Outliers. In *SIGMOD*. 93–104.

[5] Lei Cao, Yizhou Yan, Caitlin Kuhlman, Qingyang Wang, Elke A Rundensteiner, and Mohamed Eltabakh. 2017. Multi-Tactic Distance-Based Outlier Detection. In *ICDE*. IEEE, 959–970.

[6] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml visited on 07/05/2022.

[7] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf visited on 07/05/2022.

[8] Xin He, Kaiyong Zhao, and Xiaowen Chu. 2021. AutoML: A survey of the state-of-the-art. *Knowledge-Based Systems* 212 (Jan 2021), 106622. https://doi.org/10.1016/j.knosys.2020.106622 visited on 07/05/2022.

[9] Edwin M. Knorr and Raymond T. Ng. 1998. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *VLDB*. 392–403.

[10] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*. 413–422. https://doi.org/10.1109/ICDM.2008.17 visited on 07/05/2022.

[11] Spiros Papadimitriou, Hiroyuki Kitagawa, Phillip B. Gibbons, and Christos Faloutsos. 2003. LOCI: Fast Outlier Detection Using the Local Correlation Integral. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*. 315–326.

[12] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim. 2000. Efficient Algorithms for Mining Outliers from Large Data Sets. In *SIGMOD*. 427–438.

[13] Zeyuan Shang, Emanuel Zgraggen, Benedetto Buratti, Ferdinand Kossmann, Philipp Eichmann, Yeounoh Chung, Carsten Binnig, Eli Upfal, and Tim Kraska. 2019. Democratizing Data Science through Interactive Curation of ML Pipelines. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 1171–1188. https://doi.org/10.1145/3299869.3319863 visited on 07/05/2022.

[14] Yizhou Yan, Lei Cao, and Elke A Rundensteiner. 2017. Scalable Top-n Local Outlier Detection. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.