AnomalyKiTS: Anomaly Detection Toolkit for Time Series

Dhaval Patel, Giridhar Ganapavarapu, Srideepika Jayaraman,
Shuxin Lin, Anuradha Bhamidipaty, Jayant Kalagnanam,

IBM Thomas J. Watson Research Center
Yorktown Heights, New York, USA, 10598
{pateldha@us., giridhar.ganapavarapu@, j.srideepika@, shuxin.lin@, anubham@us., jayant@us.}ibm.com

Abstract

This demo paper presents a design and implementation of a system AnomalyKiTS for detecting anomalies from time series data for the purpose of offering a broad range of algorithms to the end user, with special focus on unsupervised/semi-supervised learning. Given an input time series, AnomalyKiTS provides four categories of model building capabilities followed by an enrichment module that helps to label anomaly. AnomalyKiTS also supports a wide range of execution engines to meet the diverse need of anomaly workloads such as Serveless for CPU intensive work, GPU for deep-learning model training, etc.

Introduction

With wider adoption of Industry 4.0, many industrial applications are harvesting data from ongoing processes in real time. The collected data are of increasingly wide range of formats such as time series, images, alarms, quality inspection reports, etc. Among all these diverse data modalities, time series is the most common data format across multiple applications and has recently gained significant attention. For example, our recent work in the time series domain includes AutoAI-TS (Shah et al. 2021), Smart-ML (Patel et al. 2020a), TransformerML (Zerveas et al. 2021), FLOps (Patel et al. 2020b), etc.

Broadly, two types of time series data analysis toolkits are developed in the literature: General purpose toolkits such as sktime (Löning et al. 2019), pyFTS (Silva et al. 2018), tsdl (Hyndman and Yang 2018. v0.1.0), tslearn (Tavenard et al. 2020), GluonTS (Alexandrov et al. 2020) etc, and purpose-built toolkits such as Anomaly Detection toolkits (Zhao, Nasrullah, and Li 2019; Ying et al. 2020; Ren et al. 2019; Zhang, Nie, and Yuan 2020; Buda, Caglayan, and Assem 2018; Gao et al. 2020; Geiger et al. 2020; Lee, Lin, and Gran 2020; Bhatnagar et al. 2021), etc. The former one provides a range of algorithms to the end user for quick exploration; on the other hand, the latter one is tailored to support a specific usecase and is more rigid in the customization. We noticed a recent surge in unsupervised learning based anomaly toolkits. This is due to the fact that obtaining a label for supervised learning in an automated and reliable manner is a challenging task for time series data.

Although both types of system development aim to enable easy access to build anomaly solutions centered around time series, current systems overload the user with a large variety of options and APIs. Some form of standardization, formal definition and automation of the anomaly task is required. Scikit-learn library (Buitinck, Louppe, and etl 2013) has been popular among data scientists, but it is limited to mostly tabular data. The sktime (tslearn) library extended definition to support time series data but mainly concentrated on forecasting (classification) functionality. PyOD is the popular outlier detection toolkit but lacks support for time series data. Moreover, the data size and the nature of anomaly varies from application to application, and the current off-the-shelf toolkits do not cover all the usecases such as Semi-supervised anomaly, Prediction Based unsupervised anomaly, etc. In this demo paper, we present a design and implementation of a system that enables data scientists and AI practitioners to get a unified access to various anomaly detection machinery for time series data.

AnomalyKiTS : System Overview

Figure 1 gives an overview of AnomalyKiTS's layered architecture. AnomalyKiTS is based on Sklearn compliant standardized architecture, components and output schema.

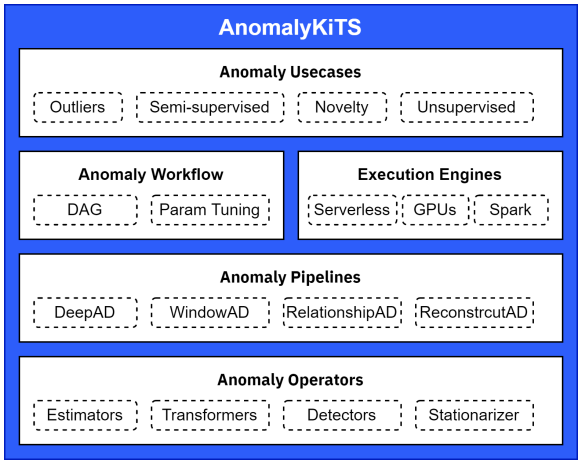


Figure 1: Layered architecture of AnomalyKiTS

Anomaly Operators. The bottom-most layer consists of basic machine learning primitives such as Estimators, Transformers, Outlier Detectors, Data Stationarizers, etc. These components tend to perform one specific task/function and are referred to as Operators. At present, we have implemented 30+ Operators for anomaly related tasks along with the components that are already available in other libraries.

Anomaly Pipelines. The second layer implements advance machine learning primitives in the form of an anomaly pipeline that logically connects different components from the lower layer. We introduced 4 types of anomaly pipelines:

- DeepAD
- RelationshipAD
- ReconstructAD
- WindowAD

These four pipelines cover a wide range of anomaly detection approaches such as: DeepAD uses an ensemble of time series forecasting models for anomaly detection (Buda, Caglayan, and Assem 2018), whereas RelationshipAD is based on the pair-wise relationship between variables for anomaly detection (Zong et al. 2018; Liu et al. 2018). Apart from generating the anomaly score in a unique way, each of the pipelines provides an additional capability in the form of "anomaly thresholding" to generate the anomaly labels (+1 for normal sample and -1 for anomalous) and if possible the predicted contribution of an individual variable. The pipeline supports two types of anomaly labeling methods: Static and Dynamic.

Anomaly Workflow. The left side of the third layer is a core data science workflow module and is inspired by the fact that, the data scientist would be interested in exploring multiple pipelines and picking the one that meets their need. To simplify the multiple pipelines specification, we adopted a Directed Acyclic Graph (DAG) based workflow construction as discussed in detail (Shrivastava et al. 2019; Patel et al. 2020a). Along with DAG, user can also configure the parameters for each forecasting pipeline to conduct hyper-parameter tuning.

Execution Engines. The right side of the third layer is an execution engine for scalable workflow exploration. This layer is an important module to meet the need of exploring multiple pipelines and/or a single pipeline with a large dataset in an efficient and scalable way. Compared to other libraries, our system provides a more uniform access to multiple execution platforms such as Watson Machine Learning for GPU based training, Spark and Serverless (Ray, Cloud Function, Code Engine) for CPU intensive task level parallelism, etc.

Anomaly Usecases. The top most layer is an application layer that offers various pre-built industrial templates to build reusable applications. In the case of un-supervised exploration, our system provides several ranking methods such as EM Score and AL Score (Goix 2016) that do not require explicit label information. In the case of semi-supervised exploration, the user provides a small amount of labeled data for obtaining the rank of each pipeline in the Workflow.

AnomalyKiTS : Benchmark and Deployment

AnomalyKiTS is tested for various datasets ranging from synthetically generated time series data (e.g., Argots) to client engagement, and public sources (Geiger et al. 2020; Wu and Keogh 2021). In the following Figure 2, we provided box plot of more that 10,000 experiments on various Static anomaly thresholds. Briefly, we train various WindowAD based anomaly detection algorithms, and then apply different scoring method to obtain the anomaly label. The generated anomaly label is compared with available ground truth. We used "recall" of an algorithm as a ranking criteria. X axis is average rank and Y-axis is various scoring methods with different parameter settings. In this case, a parameter-free "ostu" method turns out to be a winner.

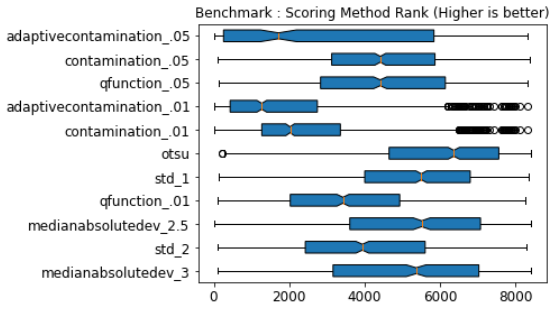
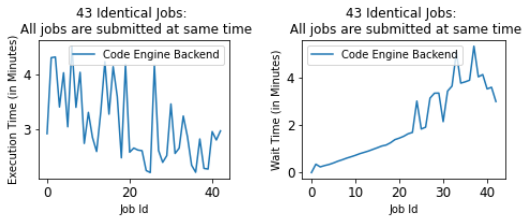


Figure 2: Benchmark : 80+ public datasets

Service Deployment. AnomalyKiTS is deployed on IBM API Hub[1]. Currently it supports two types of requests:

- **Batch mode.** scan the entire time series and detect the anomaly from any where
- **Train-Test mode.** use the historical data as a training and then detect anomaly in the most recent data



(a) Execution Time (b) Wait Time

Figure 3: Anomaly Service Testing on Code Engine

For each incoming request, Anomaly services obtain new training resources (i.e., CPU/GPU) using Serverless Code Engine or Watson Machine Learning. Figure 3 shows the promptness of the service to handle homogeneous workload of 43 incoming requests using IBM Code Engine. Each request was allocated 4 CPU with 16 GB RAM and the size of the data varies up to 10k record and 5 features.

[1]https://developer.ibm.com/apis/catalog/ai4industry--anomaly-detection-product/Introduction

References

Alexandrov, A.; Benidis, K.; Bohlke-Schneider, M.; Flunkert, V.; Gasthaus, J.; Januschowski, T.; Maddix, D. C.; Rangapuram, S.; Salinas, D.; Schulz, J.; Stella, L.; TÃ¼rkmen, A. C.; and Wang, Y. 2020. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116): 1–6.

Bhatnagar, A.; Kassianik, P.; Liu, C.; Lan, T.; Yang, W.; Cassius, R.; Sahoo, D.; Arpit, D.; Subramanian, S.; Woo, G.; Saha, A.; Jagota, A. K.; Gopalakrishnan, G.; Singh, M.; Krithika, K. C.; Maddineni, S.; Cho, D.; Zong, B.; Zhou, Y.; Xiong, C.; Savarese, S.; Hoi, S.; and Wang, H. 2021. Merlion: A Machine Learning Library for Time Series. arXiv:2109.09265.

Buda, T. S.; Caglayan, B.; and Assem, H. 2018. DeepAD: A Generic Framework Based on Deep Learning for Time Series Anomaly Detection. In Phung, D.; Tseng, V. S.; Webb, G. I.; Ho, B.; Ganji, M.; and Rashidi, L., eds., *Advances in Knowledge Discovery and Data Mining*, 577–588. Cham: Springer International Publishing. ISBN 978-3-319-93034-3.

Buitinck, L.; Louppe, G.; and etl, M. B. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop*, 108–122.

Gao, J.; Song, X.; Wen, Q.; Wang, P.; Sun, L.; and Xu, H. 2020. RobustTAD: Robust Time Series Anomaly Detection via Decomposition and Convolutional Neural Networks. arXiv:2002.09545.

Geiger, A.; Liu, D.; Alnegheimish, S.; Cuesta-Infante, A.; and Veeramachaneni, K. 2020. TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks. In *2020 IEEE International Conference on Big Data (Big Data)*, 33–43.

Goix, N. 2016. How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms? arXiv:1607.01152.

Hyndman, R.; and Yang, Y. 2018. v0.1.0. *tsdl: Time Series Data Library*.

Lee, M.-C.; Lin, J.-C.; and Gran, E. G. 2020. ReRe: A Lightweight Real-time Ready-to-Go Anomaly Detection Approach for Time Series. arXiv:2004.02319.

Liu, H.; Paffenroth, R. C.; Zou, J.; and Zhou, C. 2018. Anomaly Detection via Graphical Lasso. arXiv:1811.04277.

Löning, M.; Bagnall, A.; Ganesh, S.; Kazakov, V.; Lines, J.; and Király, F. J. 2019. sktime: A Unified Interface for Machine Learning with Time Series. In *Systems for ML NeurIPS*.

Patel, D.; Shrivastava, S.; Gifford, W.; Siegel, S.; Kalagnanam, J.; and Reddy, C. 2020a. Smart-ML: A System for Machine Learning Model Exploration using Pipeline Graph. In *2020 IEEE International Conference on Big Data (Big Data)*, 1604–1613.

Patel, D.; Yousaf Shah, S.; Zhou, N.; Shrivastava, S.; Iyengar, A.; Bhamidipaty, A.; and Kalagnanam, J. 2020b. FLOps: On Learning Important Time Series Features for Real-Valued Prediction. In *2020 IEEE International Conference on Big Data (Big Data)*, 1624–1633.

Ren, H.; Xu, B.; Wang, Y.; Yi, C.; Huang, C.; Kou, X.; Xing, T.; Yang, M.; Tong, J.; and Zhang, Q. 2019. Time-Series Anomaly Detection Service at Microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, 3009–3017. New York, NY, USA: Association for Computing Machinery. ISBN 9781450362016.

Shah, S. Y.; Patel, D.; Vu, L.; Dang, X.-H.; Chen, B.; Kirchner, P.; Samulowitz, H.; Wood, D.; Bramble, G.; Gifford, W. M.; Ganapavarapu, G.; Vaculin, R.; and Zerfos, P. 2021. *AutoAI-TS: AutoAI for Time Series Forecasting*, 2584–2596. Proceedings of the 2021 International Conference on Management of Data, SIGMOD.

Shrivastava, S.; Patel, D.; Gifford, W. M.; Siegel, S.; and Kalagnanam, J. 2019. ThunderML: A Toolkit for Enabling AI/ML Models on Cloud for Industry 4.0. In Miller, J.; Stroulia, E.; Lee, K.; and Zhang, L.-J., eds., *Web Services – ICWS 2019*, 163–180. Cham: Springer International Publishing. ISBN 978-3-030-23499-7.

Silva, P. C. L.; et al. 2018. *pyFTS: Fuzzy Time Series for Python*.

Tavenard, R.; Faouzi, J.; Vandewiele, G.; Divo, F.; Androz, G.; Holtz, C.; Payne, M.; Yurchak, R.; Rußwurm, M.; Kolar, K.; and Woods, E. 2020. Tslearn, A Machine Learning Toolkit for Time Series Data. *Journal of Machine Learning Research*, 21(118): 1–6.

Wu, R.; and Keogh, E. J. 2021. Current Time Series Anomaly Detection Benchmarks are Flawed and are Creating the Illusion of Progress. arXiv:2009.13807.

Ying, Y.; Duan, J.; Wang, C.; Wang, Y.; Huang, C.; and Xu, B. 2020. Automated Model Selection for Time-Series Anomaly Detection. *CoRR*, abs/2009.04395.

Zerveas, G.; Jayaraman, S.; Patel, D.; Bhamidipaty, A.; and Eickhoff, C. 2021. A Transformer-Based Framework for Multivariate Time Series Representation Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD, 2114–2124.

Zhang, Z. Z.; Nie, K.; and Yuan, T. T. 2020. Moving Metric Detection and Alerting System at eBay. arXiv:2004.02360.

Zhao, Y.; Nasrullah, Z.; and Li, Z. 2019. PyOD: A Python Toolbox for Scalable Outlier Detection. *Journal of Machine Learning Research*, 20(96): 1–7.

Zong, B.; Song, Q.; Min, M. R.; Cheng, W.; Lumezanu, C.; Cho, D.; and Chen, H. 2018. Deep Autoencoding Gaussian Mixture Model for Unsupervised Anomaly Detection. In *International Conference on Learning Representations*.