Generic and Scalable Framework for Automated Time-series Anomaly Detection

Nikolay Laptev
Yahoo Labs
Sunnyvale, CA, USA
nlaptev@yahoo-inc.com

Saeed Amizadeh
Yahoo Labs
Sunnyvale, CA, USA
amizadeh@yahoo-inc.com

Ian Flint
Yahoo
Sunnyvale, CA, USA
ianflint@yahoo-inc.com

ABSTRACT

This paper introduces a generic and scalable framework for automated anomaly detection on large scale time-series data. Early detection of anomalies plays a key role in maintaining consistency of person's data and protects corporations against malicious attackers. Current state of the art anomaly detection approaches suffer from scalability, use-case restrictions, difficulty of use and a large number of false positives. Our system at Yahoo, EGADS, uses a collection of anomaly detection and forecasting models with an anomaly filtering layer for accurate and scalable anomaly detection on time-series. We compare our approach against other anomaly detection systems on real and synthetic data with varying time-series characteristics. We found that our framework allows for 50-60% improvement in precision and recall for a variety of use-cases. Both the data and the framework are being open-sourced. The open-sourcing of the data, in particular, represents the first of its kind effort to establish the standard benchmark for anomaly detection.

1. INTRODUCTION

While rapid advances in computing hardware and software have led to powerful applications, still hundreds of software bugs and hardware failures continue to happen in a large cluster compromising user experience and subsequently revenue. Non-stop systems have a strict uptime requirement and continuous monitoring of these systems is critical. From the data analysis point of view, this means non-stop monitoring of large volume of time-series data in order to detect potential faults or anomalies. Due to the large scale of the problem, human monitoring of this data is practically infeasible which leads us to automated anomaly detection using Machine Learning and Data Mining techniques.

An anomaly, or an outlier, is a data point which is significantly different from the rest of the data. Generally, the data in most applications is created by one or more generating processes that reflect the functionality of a system. When the underlying generating process behaves in an unusual way, it creates outliers. Fast and efficient identification of these outliers is useful for many applications including: intrusion detection, credit card fraud, sensor events, medical diagnoses, law enforcement and others [1].

Current approaches in automated anomaly detection suffer from a large number of false positives which prohibit the usefulness of these systems in practice. Use-case, or category specific, anomaly detection models [4] may enjoy a low false positive rate for a specific application, but when the characteristics of the time-series change, these techniques perform poorly without proper retraining. Section 6.3 demonstrates the shortcoming of 'one size fits all' principle in practice.

Our system at Yahoo is called EGADS (Extensible Generic Anomaly Detection System) and it enables the accurate and scalable detection of time-series anomalies. EGADS separates forecasting, anomaly detection and alerting into three separate components which allows the person to add her own models into any of the components. Note that this paper focuses on the latter two components.

EGADS uses a set of default models that are tuned to reduce the number of false positives, which by itself suffices for the average user. More advanced use-cases, however, will require the system to capture some types of anomalies while ignoring others. The anomalies of interest may vary in magnitude, severity or other parameters which are unknown apriori and depend on the use-case. For this reason the alerting component of EGADS uses machine learning to select the most relevant anomalies for the consumer.

To the best of our knowledge EGADS is the first comprehensive system for anomaly detection that is flexible, accurate, scalable and extendible. EGADS is being open-sourced along with the anomaly detection benchmarking data. The open-sourcing of the data and the system will provide the first of its kind benchmarking data and the framework to help the academics and the industry collaborate and develop novel anomaly detection models. At Yahoo, EGADS is used on millions of time-series by many teams daily.

In Section 2 we describe the EGADS architecture. The algorithms and the alerting module are described in Sections 3 and 4 respectively. Previous work is described in Section 5 followed by the real-world use-cases and conclusion in Sections 6 and 7 respectively.

2. ARCHITECTURE

The EGADS framework consists of three main components: the time-series modeling module (TMM), the anomaly detection module (ADM) and the alerting module (AM). Given a time-series the TMM component models the time-series producing an expected value later consumed by the ADM and AM components that, respectively, compute the error and filter uninteresting anomalies. These components are described in detail in Sections 3 and 4.

EGADS was built as a framework to be easily integrated into an existing monitoring infrastructure. At Yahoo, our internal Yahoo Monitoring Service (YMS) processes millions of data-points every second. Therefore, having a scalable, accurate and automated anomaly detection for YMS is critical. We describe the integration details into YMS next.

2.1 System Integration

EGADS operates as a stand-alone platform that can be used as a library in larger systems. Therefore, designing an interface between EGADS and an internal Yahoo monitoring service (YMS) is critical. A key constraint of YMS is scale; the platform needs to evaluate millions of data points per second. As a result, many of the integration architecture decisions are focused on optimizing real-time processing. The integration with YMS is shown in Figure 1.

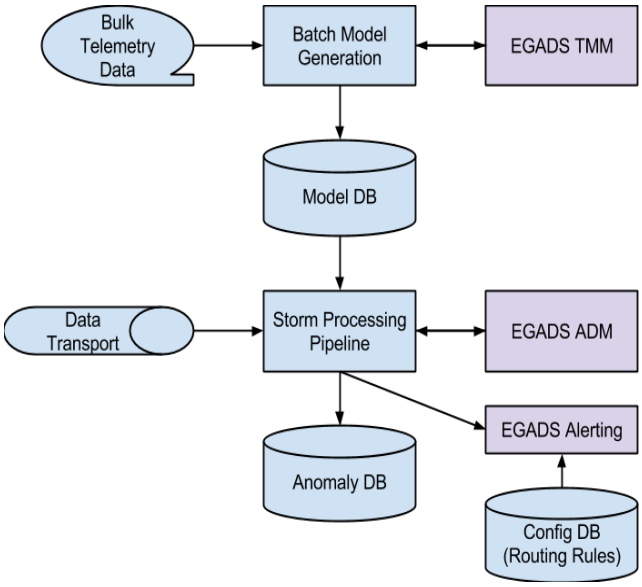


Figure 1: EGADS-YMS Architecture

Several support components are required to drive action based on detected anomalies. First of all, all anomaly detection models are generated in batch and applied in real time. The batch flow is comprised of three steps:

1. Telemetry (i.e. the monitored time-series) data are stored in bulk on a Hadoop cluster.
2. A batch model generator runs against these data and builds models for targeted time-series.
3. The models are stored in a model database.

The online flow then utilizes the stored models.

1. Data flows into a Storm [22] stream-processing topology.
2. One of the bolts (modules) in the topology calls the EGADS ADM to evaluate incoming data points based on models stored in the model database.
3. If an anomaly is present, this is fed to a secondary rule flow, consisting of combinatorial rules and other use-cases specific logic (see Section 4).
4. Based on the rules, if the anomaly is an alert event, the event is generated, stored in a status database, and forwarded to an alert routing system.
5. The alert routing system applies routing configuration rules to send the alert to the appropriate support staff.

2.2 Scalability

The monitoring use case for EGADS requires the evaluation of millions of data-points per second, across over one hundred million time-series. This has scalability implications in terms of CPU load, I/O, and memory footprint. The evaluation of a datapoint needs to be as efficient as possible. This means that as much of the model as possible should be precomputed. It is not practical to read a model from disk each time a datapoint arrives because of the rate of inbound traffic. This suggests that the models should be stored in memory. In order to contain costs, the models should be as small as possible.

One optimization is to share models across multiple similar time-series. This is practical in the context of a large web serving environment, since applications are broken into horizontal tiers of similar servers. This optimization will reduce the memory footprint, the batch workload, and I/O against the model database.

Another possible optimization is to investigate self-tuning models; models that update themselves based on a stream of inbound data via online learning rather than requiring periodic batch generation. Models of this type may need to be initialized in batch, but overall they will reduce the batch workload. Depending on implementation, however, they may increase writes against the model database since they are being constantly refined.

Yet another optimization involves a trade-off between model size, training speed and accuracy. Depending on the characteristics of the time-series a light and fast forecasting model can provide similar accuracy as a more sophisticated one. We evaluate some of these optimization approaches in Section 6.2.2.

3. ANOMALY DETECTION ALGORITHMS

In this section, we give a big picture overview of the anomaly detection algorithms supported by EGADS. Currently, EGADS is capable of detecting three classes of anomalies:

(a) **Outliers:** given an input time-series x, an outlier is a timestamp-value pair $\langle t, x_t \rangle$ where the observed value

x_t is *significantly* different from the expected value of the time-series at that time, i.e. $\mathbb{E}(x_t)$.

(b) **Change points:** given an input time-series x, a change point is a timestamp t such that the behavior of the time-series is *significantly* different before and after t.

(c) **Anomalous time-series:** given a set of time-series $X = \{x^{(i)}\}$, an anomalous time-series $x^{(j)} \in X$ is a time-series whose behavior is *significantly* different from the majority of the time-series in X.

In the following sections, we give the general sketch of the methods that are currently used in EGADS for detecting the aforementioned anomaly types.

3.1 Outlier Detection

Detecting outliers is the most important functionality in many monitoring applications. For this reason the main focus of this paper is on outlier detection and unless it is explicitly specified, by anomalies, we refer to outliers by default.

EGADS offers two classes of algorithms for detecting outliers, which are described in this section.

3.1.1 Plug-in methods

The first class of methods for time-series outlier detection in EGADS are called *plug-in* methods. These methods *explicitly* model the normal behavior of the time-series such that a significant deviation from this model is considered an outlier. To model the normal behavior of the input time-series we can plug-in a wide range of time-series modeling and forecasting models (e.g. ARIMA [26], Exponential Smoothing [11], Kalman Filter [9], State Space Models [6], etc.) depending on the application and the nature of time-series. That is why we refer to this general strategy as the plug-in methods. It should be noted that all these models are used in EGADS for time-series forecasting which is another feature of our framework; however, since the focus of this paper is on anomaly detection, we do not give more details on modeling and forecasting features of EGADS.

Our proposed Plug-in framework consists of two main components: the time-series modeling module (TMM) and the anomaly detection module (ADM). Given a time-series $X = \{x_t \in \mathbb{R} : \forall t \geq 0\}$, the TMM provides the predicted value of x_t at time t, denoted by u_t. We also refer to this quantity as the expected value of x_t (not to be confused with the mathematical notion of expectation). The TMM can be a machine learned model which makes predictions based on some training data or a rule-based system which encodes expert's knowledge about how x_t behaves at time t. In this paper, we *do not* make any assumption regarding the TMM; that is, the TMM is just a black box module in our proposed method that generates predictions u_t. In this sense, our proposed framework is *generic* and does not depend on any specific time-series modeling framework.

Given the predicted value u_t and the actual observed value x_t, the ADM computes some notion of deviation which we refer to as the deviation metric (DM). The simplest measure of deviation is the prediction error, $PE_t = x_t - u_t$. If the error falls outside some fixed thresholds, an alert is issued. This simple method may work in some cases, but it will not be a good strategy for most because it does not capture the relative error. The *relative error*, RE_t is defined as a factor of u_t:

$$RE_t = \frac{x_t - u_t}{u_t} = \frac{x_t}{u_t} \tag{1}$$

By thresholding the relative error, one can detect anomalies while normalizing out the dependence on the magnitude of the expected value. The values of these thresholds, indeed, determine how sensitive the anomaly detection module is. Various thresholding techniques are described in Section 4. Despite its common usage and effectiveness, however, there is no reason to believe the relative error is always the optimal metric for anomaly detection on a given time-series. In fact, the choice of the optimal metric for a given time-series highly depends on the nature of the time-series as well as the TMM performance. For instance, if we are dealing with a very regular time-series for which we have an accurate model, using the prediction error for anomaly detection might be sufficient as it is expected to be Normally distributed. In other cases, the optimal metric might be something between the prediction error and the relative error. For this reason, EGADS tracks a set of deviation metrics by default and the person using the system can create her own error metrics. These error metrics, together with other features, such as the time series characteristics, are used in the alerting module (AM), described in Section 4, to learn consumer's preferences and filter unimportant anomalies.

3.1.2 Decomposition-based methods

The second class of outlier detection methods in EGADS is based on the idea of time-series decomposition. In particular, in the time-series analysis literature, it is a common practice to decompose a time-series into three components: *trend*, *seasonality* and *noise*. By monitoring the noise component, one can capture the outliers. More precisely, if the absolute value of the noise component of point x_t is greater than a certain threshold, one can announce x_t as an outlier.

The decomposition of time-series can be done both in the time-domain via smoothing or in the frequency-domain via spectral decomposition. STL (Seasonal-Trend Decomposition based on Loess) [5] is a famous technique that uses Loess smoothing for decomposition. The frequency-domain methods can be further divided into parameteric and non-parametric methods. For the parametric methods, the basis used for spectral decomposition has a known parametric form (such as Fourier transform [2] or wavelet transform [19]) whereas, for non-parametric methods, the basis is data-driven [18].

3.2 Change Point Detection

Change points are those points in time where the behavior of the time-series starts to deviate from what is expected. The big difference between change points and outliers is that change points correspond to more sustained, long-term changes compared to volatile outliers. A common strategy

for detecting change points in the literature is to move two side-by-side windows on the time-series and compute the difference between the behavior of the time-series in the two windows as a measure of the deviation metric [12, 27, 17, 20]. The behavior of the time-series in each window is typically modeled by the distribution of the values, motifs, frequencies, etc. that are present in the time-series. We refer to these techniques as the *absolute* techniques because they do not make explicit assumptions regarding the expected behavior of the time-series.

In EGADS, currently we have taken a different approach which we refer to as the *relative* or *model-based* methods. In these methods, the expected behavior of the time-series is explicitly modeled through one of the modeling techniques mentioned in Section 3.1.1. In particular, we incorporate the plug-in approach described in Section 3.1.1 to compute the sequence of *residuals* (or deviations from the model expectation) for an input time-series. Then we apply the absolute change point detection methods on the series of residuals to detect a change in the distribution of the residuals. We have used Kernel Density Estimation [7] to non-parametrically estimate the distribution of the residuals and the Kullback-Leibler divergence [16] to measure the change in the distribution.

We believe the model-based change point detection methods are more useful than the absolute methods in the practical applications. This is because the change points are meaningful as much as our models cannot explain the behavior of the time-series after a certain time point. However, if the model can explain the time-series behavior even after an absolute change point, from the practical point of view, there is no reason for us to consider that time point as a change point. In other words, the change points are relative to the underlying model used to explain the behavior of the time-series, which in turn gives rise to the relative change-point detection techniques.

3.3 Detecting Anomalous Time-series

Another class of anomaly detection techniques supported by EGADS involves detecting anomalous time-series. An anomalous time-series T is defined as a time-series whose average deviation from the other time-series is significant. Assuming all time-series are homogeneous and come from the same source (i.e. are part of the same cluster) one can simply compute the average deviation for time-series (i) relative to other time-series. In EGADS our current approach involves clustering the time-series into a set of clusters C based on various time-series features including trend & seasonality, spectral entropy, autocorrelation, average Euclidean distance etc. After clustering we perform intra or inter-cluster time-series anomaly detection by measuring the deviation within or among the cluster centroids and the time-series (i). A common use-case for this EGADS anomaly detection type involves triaging. For example if a network engineer wants to find an anomalous server amongst millions of time-series, it can be impractical with the previous approaches because the modeling is done on the *per* time-series basis without taking into account the behavior of other metrics. Another application of this anomaly detection type is in finding similar anomalies, which is the inverse of the previous use-case.

4. ALERTING

The end-goal of anomaly detection is to produce accurate and timely alerts. EGADS achieves this via a two stage process by first generating a set of candidate anomalies by *threshold selection* and then *filtering* the irrelevant anomalies for a given use-case.

4.1 Threshold Selection

The job of threshold selection is to select appropriate thresholds on the deviation metrics produced by the anomaly detection module (ADM). Currently EGADS implements two algorithms for threshold selection based on (a) $K\sigma$ deviation and (b) density distribution.

The first approach is parametric and assumes that the data is normally distributed with a well-defined mean and standard deviation. Relying on the Gaussian distribution we can apply a well known statistical tool called the 'three-sigma rule' which states that 99.73% of all samples lie within three standard deviations of the mean. Therefore, depending on the value of K in $K\sigma$, one can be confident as to the probability of observing a sample at time t. Depending on the desired level of sensitivity, one can measure if a given sample lies within the 95.45% or 68.27% of all the samples for $K = 2$ or 1 respectively. Note that the assumption here was that our deviation metrics are normally distributed.

The second approach is non-parametric and is useful for the cases when the deviation metric is not normally distributed. The basic idea is to to find low density regions of the deviation metric distribution. One approach is to use an algorithm such as Local Outlier Factor (LOF) [3] which is based on a concept of a local density, where locality is given by nearest neighbors, whose distance is used to estimate the density. By comparing the local density of an object to the local densities of its neighbors, one can identify regions of similar density, and points that have a substantially lower density than their neighbors. These are considered to be outliers.

4.2 Filtering

Filtering performs the last stage post-processing on the anomalies which are then delivered to the consumer. While the candidate anomalies, which are the input to the filtering stage, are statistically significant, not all of them will be relevant for a particular use-case. For example some consumers are interested in spikes in the time-series, while others are interested in dips, yet others are interested in change points. EGADS provides a simple and intuitive interface which allows users to mark the regions of the time-series that are anomalous. This feedback is then used by EGADS together with time-series and model features to train a classifier that predicts if an anomaly a_i is relevant to user u_j. The time-series features tracked by EGADS are shown in Table 1 and are described in more detail in [25]. Section 6.4 explores the performance of a filtering module for a specific use-case. Like other components of EGADS, the filtering component is extensible in terms of models and features.

Time-series feature	Description
Periodicity (frequency)	Periodicity is very important for determining the seasonality.
Trend	Exists if there is a long-term change in the mean level
Seasonality	Exists when a time series is influenced by seasonal factors, such as month of the year or day of the week
Auto-correlation	Represents long-range dependence.
Non-linearity	A non-linear time-series contains complex dynamics that are usually not represented by linear models.
Skewness	Measures symmetry, or more precisely, the lack of symmetry.
Kurtosis	Measures if the data are peaked or flat, relative to a normal distribution.
Hurst	A measure of long-term memory of time series.
Lyapunov Exponent	A measure of the rate of divergence of nearby trajectories.

Table 1: Time-series features used by EGADS

Figure 2 shows the feature profile of a sample time-series. Note that the metrics beginning with *dc* are obtained on the adjusted time-series (i.e. after removing trend and seasonality). In Section 6.2 we look at how these time-series characteristics impact the model performance.

5. RELATED WORK

There are a number of anomaly detection techniques in the literature. The techniques range from point anomaly detection algorithms to change-point detection algorithms. In [21] authors propose an outlier detection technique based on hypothesis testing, which is very accurate at detecting extreme outliers. In fact Twitter, [23], uses [21] in conjunction with piecewise approximation of the underlying long-term trends to remove many of the false positives. Twitter's approach is fast and enjoys an impressive precision and recall, however it is specific to the use-case of Twitter. There are also a number of open-source point anomaly detection techniques available including [24, 15].

Authors in [13] provide an anomaly detection technique that finds 'Change Points' or 'Level Shifts'. Change Points (CP) are different form point anomalies or point outliers in that CP reflect a change in underlying statistic of the time-series (e.g., Mean shift). CP typically occurs in a time-series with a launch of a new product feature or a new platform. There are a number of open-source change point detection algorithms available including [14].

In our experience, a particular anomaly detection algorithm is usually applicable to only a specific use-case. As authors in [1] mention the anomalies will have typically a high anomaly score, but the high score alone is not a distinguishing factor for an anomaly. Rather, it is the analyst, who regulates the distinction between noise and anomaly. Similarly, authors in [4] provide a concise overview of the anomaly detection technique per category, citing the fact that only a set of anomaly models are most appropriate for a given anomaly category of interest. Therefore, based on the observation that 'One Size Fits All' is a myth in the anomaly detection world, EGADS uses a strategy where a collection of well trained anomaly detection models with a post-processing use-case-specific anomaly filtering stage is used.

6. EXPERIMENTAL STUDY

We present the experiments for the modeling, anomaly detection and alerting components of EGADS next.

6.1 Data

The dataset used for the experiments is comprised of a mixture (50/50) of synthetic and real data. We have created a synthetic time-series generation tool that is being open-sourced along with the framework and the benchmarking data. Using the tool, each synthetic time-series is generated by specifying the length, magnitude, number of anomalies, anomaly type, anomaly magnitude, noise level, trend and seasonality. These parameters are picked from a fixed distribution. The real dataset is comprised of Yahoo Membership Login (YML) data. The YML data tracks the aggregate status of user logins to the Yahoo network. Both the synthetic and real time-series contain 3000 data-points each, which for the YML data represent 3 months worth of data-points. Unless otherwise stated, all experiments were run on 1000 randomly picked time-series and the results were averaged. Also note that both the synthetic and real-time data have anomaly labels, that are either synthetically or editorially generated, allowing us to measure precision and recall.

6.2 Modeling Experiments

Time-series modeling (captured by the TMM component in EGADS) is a fundamental part of anomaly detection. It is often the case that the anomaly detection is as good as the underlying time-series model. Due to a large number of candidate models, model-selection becomes critical and depends upon time-series characteristics and available resources. In the experiments that follow, we demonstrate the impact of time-series features on the model performance and show the trade-off between accuracy, memory usage and training time. The models and the error metrics used in the experiments are described in Tables 2 and 3 respectively. More details about the models and the metrics can be found in [10] and [25].

6.2.1 Time-series Characteristics and Model Performance

To demonstrate the impact of time-series features on model performance we compare the error metrics of different models when fitting time-series with different features (see Section 4.2). Figure 3 shows that time-series characteristics play an important role in model behavior. For example the Olympic Model, which is a seasonal model, performs poorly on a dataset with no seasonality and a strong trend. EGADS keeps track of the historic time-series characteristics and model performance. Using this historical information, EGADS selects the best model (given the time-series features) judged by the error metrics described in Table 3. In practice, performing model selection based on the data features is much faster than performing cross-validation for every model.

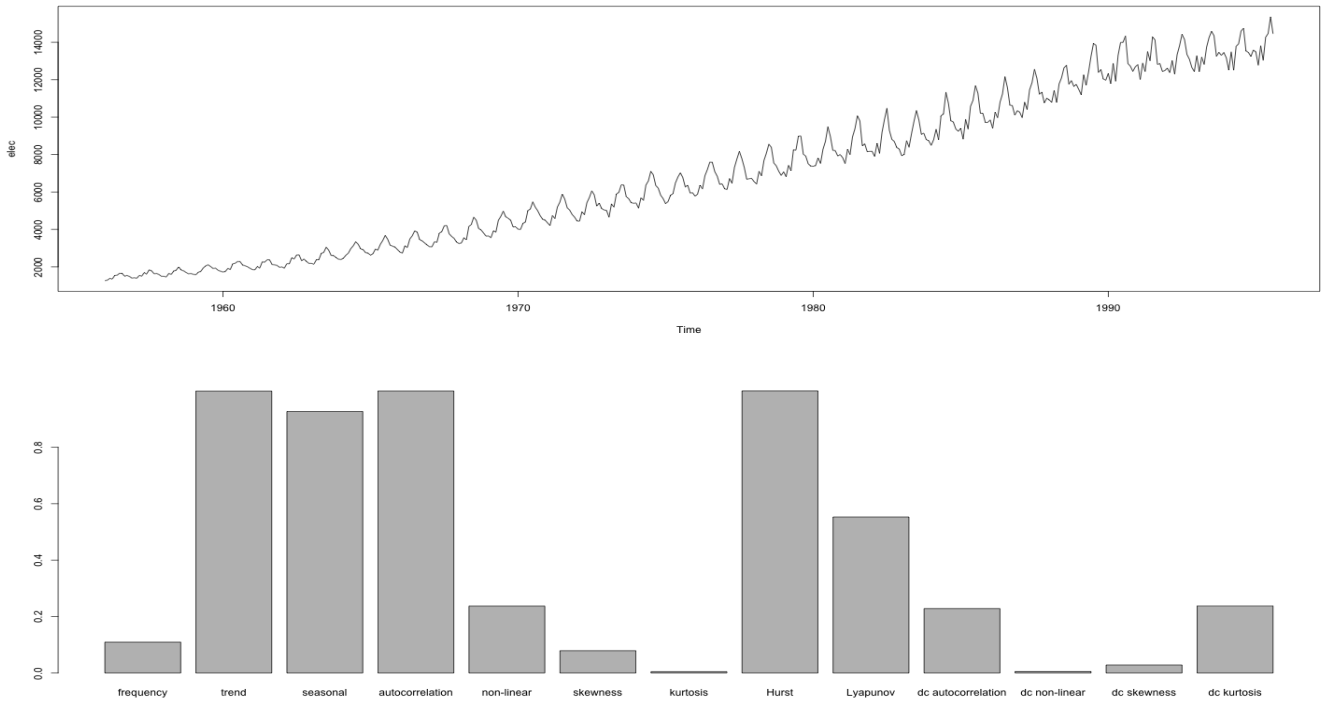


Figure 2: An example of the time-series and its characteristics extracted by EGADS. These characteristics are used by EGADS for filtering and model selection.

Model	Description
Olympic Model (Seasonal Naive)	The naive seasonal model where the prediction for next point is a smoothed average over previous n periods.
Exponential Smoothing Model	A popular model used to produce smoothed time-series. Double and Triple exponential smoothing variants add trend and seasonality into the model. the ETS model used for the experiments automatically picks the best 'fit' exponential smoothing model.
Moving Average Model	In this mode, the forecast is based on an artificially constructed time series in which the value for a given time period is replaced by the mean of that value and the values for some number of preceding and succeeding time periods. The Weighted Moving Average and Naive Forecasting Model are special cases of the moving average model.
Regression Models	Models the relationship between x & y using one or more variable.
ARIMA	Autoregressive integrated moving average.
(T)BATS Family	(Trigonometric) Exponential smoothing state space model with Box-Cox transformation.

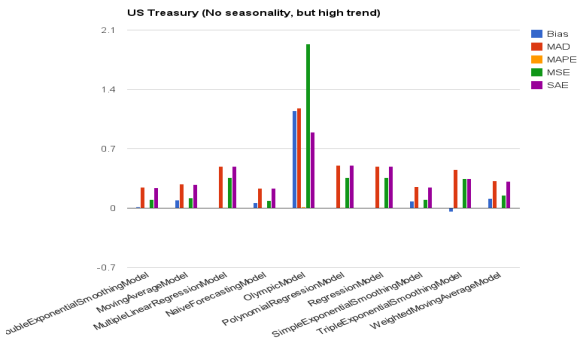
Table 2: Models Used for Modeling Experiments

Model	Description
Bias	The arithmetic mean of the errors.
MAD	The mean absolute deviation. Also known as MAE.
MAPE	The mean absolute percentage error.
MSE	The mean square of the errors.
SAE	The Sum of Absolute Errors.
ME	Mean Error.
MASE	Mean absolute scaled error.
MPE	Mean percentage error.

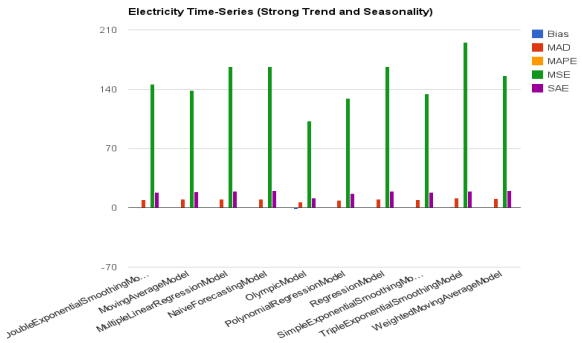
Table 3: Metrics Used for Modeling Experiments

6.2.2 *Time-series Model Scalability*

As discussed in Section 2 it is often prohibitive to build models for every time-series and optimization techniques are required to support real-time performance over massive (e.g., millions of points every second) data-streams. A fundamental optimization performs a trade-off between model size, training time and accuracy. Such a trade-off is shown in Figures 4(a) and 4(b). From the figure, for example, it is clear that the Seasonal Naive model is quick to train but has a relatively large memory requirement and a high average error. At Yahoo, a target in terms of resources and training time is first set and then the models are picked accordingly. In other words, the objective is to minimize the errors in Table 3 subject to the resource and model building time constrains. Other optimization techniques including time-series sampling and model sharing are being investigated.

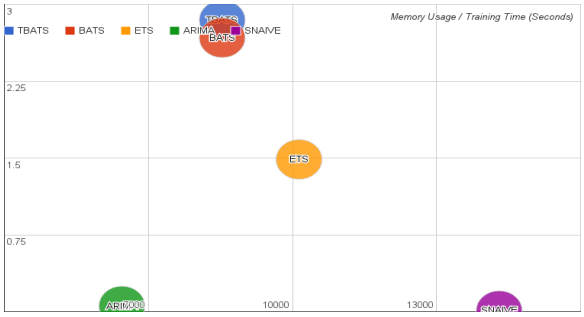


(a) Model performance on TimeSeries with trend.

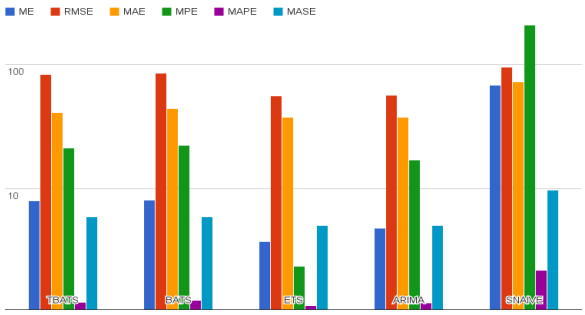


(b) Model performance on TimeSeries with seasonality.

Figure 3: Model performance on time-series with varying characteristics.



(a) Model Tradeoff: Model Size vs Training Time



(b) Model Accuracy

Figure 4: Model trade-offs

6.3 Anomaly Detection Experiments

In this section we compare open source system against EGADS. The open source systems considered are shown in Table 4. The results on the data described in Section 6.1 are shown in Figure 5. The results are compared in terms of the standard $F_1\text{-}Score = 2 \times \frac{precision \times recall}{precision + recall}$. The results indicate that there is no best anomaly detection model for all use-cases. In particular different algorithms are best at detection different types of anomalies. For example Twitter [13] performs best on dataset *TS-2* while ExtremeLowDensity model is best on *TS-3*. These datasets contain a mixture of anomaly types (e.g., outliers, change-points), and one might argue that comparing an algorithm that is only meant for change-point detection is not fair. Recall, however, that the motivation for EGADS was that the user should be agnostic to the type of time-series and the type of anomalies that are in the data. The system must be able to gracefully and robustly deal with a wide variety of anomalies present in the data. For this reason, EGADS is built as a library that combines a set of anomaly detection models into a single framework. The anomalies from these models are forwarded to the filtering component for accurate anomaly detection.

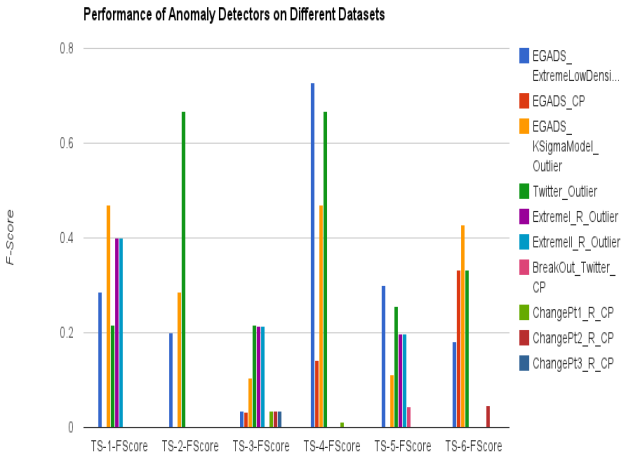


Figure 5: Anomaly model performance on different datasets. Observe that there is no single model that is best on all datasets.

6.4 Anomaly Filtering Experiments

The importance of an anomaly often depends on the use-case. Specifically, some users may be interested in the time-series behavior that exhibits a malicious attack, while others may be interested in revenue drops. Yahoo Membership (YM) use-case refers to the former set of users. Specifically for the YM use-case, editors supplied feedback to EGADS

Model	Description
EGADS ExtremeLow-DensityModel Outlier	EGADS density-based anomaly detection.
EGADS CP	EGADS kernel-based change-point detection.
EGADS KSig-maModel Outlier	EGADS re-implementation of the classic k-sigma model.
Twitter Out-lier	The Open-Source Twitter-R anomaly detection library based on the Generalized ESD method.
ExtremeI & II R Outlier	Open source univariate outlier detection that threshold the absolute value and the residual to detect anomalies.
BreakOut Twitter CP	A package from Twitter that uses an ESD statistics test to detect change points.
ChangePt1 R CP	An R library that implements various mainstream and specialized change-point methods for finding single and multiple change-points within data. Method I uses a change in variance.
ChangePt2 & 3 R CP	Detects a change in the mean and the variance.

Table 4: Open Source systems used for evaluation

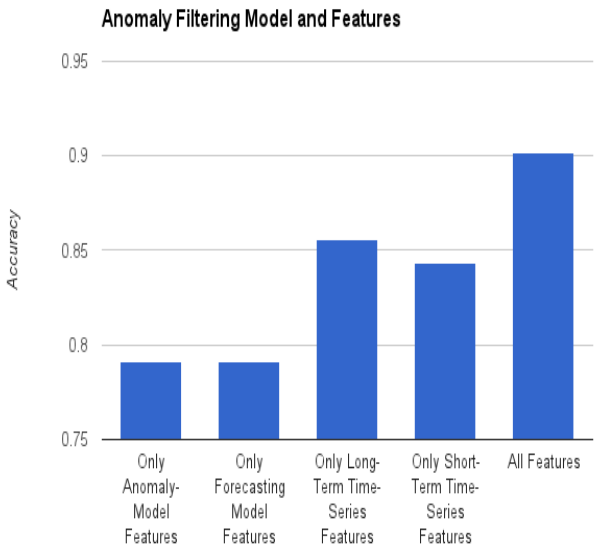


Figure 6: Accuracy of the filtering stage using different types of features.

of instances that exhibited abnormal spikes and level shifts. *Abnormal* in the case of YM meant *seasonal* followed by *non-seasonal* behavior which characterizes most of the attacks. Also the YM editors did not care about *traffic-shift* behavior, where a large drop in traffic was observed in a time-series due to router table being updated.

To address this requirement the filtering stage scanned all anomalies a_i from all models and using a model classified if a_i was a true positive. The model used in the filtering stage for the YM use-case is a boosted tree model based on AdaBoost [8]. The features used in the model are described in Table 1. The core principle of AdaBoost is to fit a sequence of weak learners (e.g., small decision trees) on repeatedly modified version of the data. The final result is then produced via a combined weighted majority vote. On each iteration, the examples that are difficult to predict receive a higher importance in the next iteration and therefore each subsequent weak learner focuses on the examples that are missed by the previous learners in the sequence. Besides the time-series features described in Table 1 we use the model features described in Section 6.2. The experiments in Figure 6 indicate an impressive precision/recall even with just the time-series features compared to just using the model alone without the filtering stage. This experiment underlines an important principle and a critical component of any anomaly detection framework: an anomaly is use-case specific and must be learned automatically for a fully scalable and automated solution.

7. CONCLUSION

Anomaly detection is a critical component at the heart of many real-time monitoring systems with applications in fault detection, fraud detection, network intrusion detection and many others. Despite its crucial importance, implementing a fully-automatic anomaly detection system in practice is a challenging task due to the large problem scale and the diverse use-cases residing in the real-world setting. These challenges typically result in solutions that are either not scalable or highly specialized, which would in turn result in a high rate of false positives when applied to other use-cases.

In this paper, we introduced EGADS, the generic anomaly detection system implemented at Yahoo to automatically monitor and alert on millions of time-series on different Yahoo properties for different use-cases ranging from fault detection to intrusion detection. As we described in the paper, the parallel architecture of EGADS on Hadoop as well as its stream processing mechanism through Storm enable it to perform real-time anomaly detection on millions of time-series at Yahoo. Furthermore, EGADS employs different time-series modeling, and anomaly detection algorithms to handle different monitoring use-cases. By incorporating this array of algorithms combined with a machine-learned mechanism in the alerting module, EGADS automatically adapts itself to the anomaly detection use-case that is important to the user. All of these features effectively create a powerful anomaly detection framework which is both generic and scalable. Our showcase experiments on real and synthetic datasets have shown the superior applicability of our framework compared to its rival solutions.

Last but not least, EGADS by its very nature is extendable, providing an easy mechanism to plugin new models and algorithms into the system. This feature specifically creates an oppurtunity for the community to contribute to EGADS. Finally, to further engage with the anomaly detection and monitoring community, our framework together with all its datasets are contributed to the open source repository.

8. REFERENCES

[1] C. Aggarwal. *Outlier Analysis*. Springer New York, 2013.

[2] P. Bloomfield. *Fourier analysis of time series: an introduction*. John Wiley & Sons, 2004.

[3] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[5] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[6] J. Durbin and S. J. Koopman. *Time series analysis by state space methods*. Number 38. Oxford University Press, 2012.

[7] V. A. Epanechnikov. Non-parametric estimation of a multivariate probability density. *Theory of Probability & Its Applications*, 14(1):153–158, 1969.

[8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1996.

[9] S. S. Haykin, S. S. Haykin, and S. S. Haykin. *Kalman filtering and neural networks*. Wiley Online Library, 2001.

[10] R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, pages 679–688, 2006.

[11] R. H. Jones. Exponential smoothing for multivariate time series. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 241–251, 1966.

[12] Y. Kawahara, T. Yairi, and K. Machida. Change-point detection in time-series data based on subspace identification. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 559–564. IEEE, 2007.

[13] A. Kejariwal and P. Kumar. Mitigating user experience from 'breaking bad': The twitter approach. In *Velocity 2014*, New York, NY, Sept. 2014.

[14] R. Killick. *changepoint, an R package that implements various mainstream and specialised changepoint methods.*, 2014.

[15] L. Komsta. *outliers, an R package of some tests commonly used outlier detection techniques.*, 2011.

[16] S. Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[17] S. Liu, M. Yamada, N. Collier, and M. Sugiyama. Change-point detection in time-series data by relative density-ratio estimation. *Neural Networks*, 43:72–83, 2013.

[18] V. Moskvina and A. Zhigljavsky. An algorithm based on singular spectrum analysis for change-point detection. *Communications in Statistics-Simulation and Computation*, 32(2):319–352, 2003.

[19] D. B. Percival and A. T. Walden. *Wavelet methods for time series analysis*, volume 4. Cambridge University Press, 2006.

[20] B. K. Ray and R. S. Tsay. Bayesian methods for change-point detection in long-range dependent processes. *Journal of Time Series Analysis*, 23(6):687–705, 2002.

[21] B. Rosner. Percentage points for a generalized esd many-outlier procedure. *Technometrics*, 25(2):165–172, 1983.

[22] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy. Storm@twitter. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 147–156, New York, NY, USA, 2014. ACM.

[23] O. Vallis, J. Hochenbaum, and A. Kejariwal. A novel technique for long-term anomaly detection in the cloud. In *6th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 14)*, Philadelphia, PA, June 2014. USENIX Association.

[24] M. van der Loo. *extremevalues, an R package for outlier detection in univariate data*, 2010. R package version 2.0.

[25] X. Wang, K. Smith-Miles, and R. Hyndman. Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series. *Neurocomput.*, 72(10-12):2581–2594, June 2009.

[26] W. W.-S. Wei. *Time series analysis*. Addison-Wesley publ, 1994.

[27] Y. Xie, J. Huang, and R. Willett. Change-point detection for high-dimensional time series with missing data. *Selected Topics in Signal Processing, IEEE Journal of*, 7(1):12–27, 2013.