Detecting Anomalies in Log Files

Using Supervised and Unsupervised ML Models

Sree Maganti Manel Mahroug Thanu Stewart

INTRODUCTION

Background

Due to the increase in size and complexity of software systems, traditional log anomaly detection technique, such as keyword searches, no longer suffice. Machine learning techniques have instead gained prominence due their ability to automate, scale, and continuously improve.

Objective

To detect anomalies in large computer system log files using a generalized approach of parsing and feature extraction that can be easily adopted by various log files

Data Source

Loghub is a publicly available data source for log files. This project focused on using the following data sets for its analysis:

- BlueGene/L supercomputer system (BGL) log files
- Thunderbird log files.

Scan for Source code

METHODOLOGY

Sampling

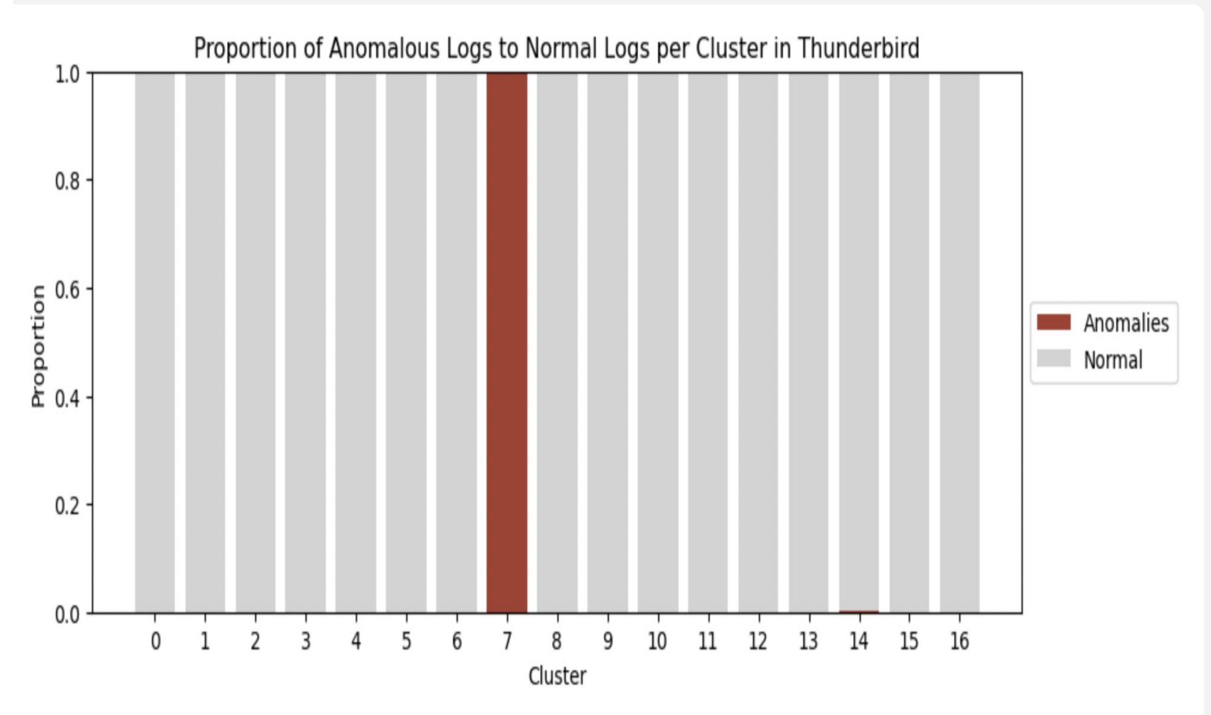
Sampling was conducted with two key consideration in mind:(1) maintaining the proportion of anomalies to normal log lines (2)limiting discontinuity by selecting a large sample chunk size.

Parsing

The Parser extracted timestamps, labels, and other parts of each log line that did not vary for the entire file

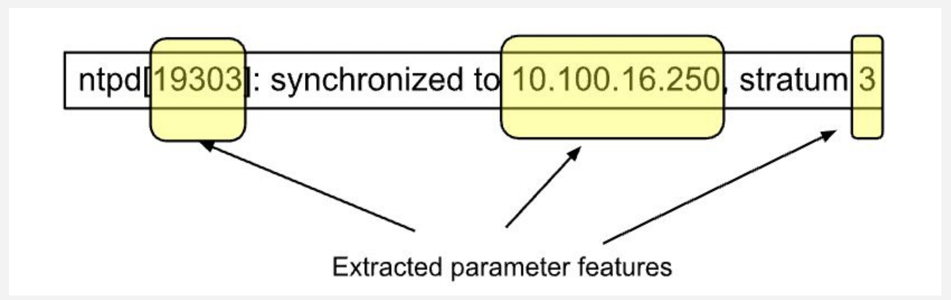
TFIDF/Clustering

TFIDF vectorization of log texts was performed, followed by K-Means clustering. Logs with similar structure were grouped together.The figure below shows an example of how well the clustering algorithm performed in discriminating between anomalies and normal logs



Parameter Extraction & Filtering

Within each cluster, infrequent tokens were extracted and added as features to the models.



ntpd[19303]: synchronized to 10.100.16.250, stratum 3
Extracted parameter features

The remaining tokens were then stored in a separate column and used for a second stage TFIDF vectorization.

Sliding Window

To account for the possibility of context dependent anomalies, we extracted a new set of features that document the count of cluster types within a fixed window

MODELING

The following models were ran during the course of our project

Supervised Models

Logistic Regression: used as baseline model, specifying class_weight as balanced to address the class imbalance. Can only model linearly separable data

XGBoost: Tree-based model (instead of a linear model. We specified a low max_depth as not overfit, and the default learning rate of 0.3.

Gradient Boosted Trees:Unlike Logistic regression, GBT can model highly non-linear decision boundaries.

Unsupervised Models

K-means: used to create clusters of similar log lines

Isolation Forest: used to solate anomalies by recursively partitioning the data into subsets.

One Class SVM:selected as it's designed to perform well under class imbalance

RESULTS

The tables below display the F1 score of our models, highlighting how different features influence model performance

	TFIDF raw text	TFIDF filtered text	Clusters Only	Clusters w/Params	Sliding Window
Logistic Regression	.9549	.9967	.9257	.9257	.1028
Gradient Boosting	.9459	.9830	.9257	.9257	.3139
XGBoost	.9299	.9817	.6092	.6092	.1697
Isolation Forest	.0000	.0000	.0000	.0000	.0000
One Class SVM	.0539	.1102	.2218	.2218	.2299

BGL Test F1 Scores by model and dataset

	TFIDF raw text	TFIDF filtered text	Clusters Only	Clusters w/Params	Sliding Window
Logistic Regression	1.000	.9986	.9982	.9982	.9990
Gradient Boosting	.9991	.9986	.9982	.9982	.9990
XGBoost	.9982	.9982	.9982	.9982	.9990
Isolation Forest	.0000	.0000	.0000	.0000	.2512
One Class SVM	.0003	.0083	.0489	.0489	.3984

Thunderbird Test F1 Scores by model and dataset

DISCUSSION

Our biggest takeaways are as follow:

1. The anomalies in our datasets present as single lines and are not context dependent
2. Unsupervised methods didn't perform well because the anomalies were clustered together, making them inliers rather than outliers.