Zero Event Anomaly Detection in Big Data using Spark for Fast and Streaming Applications

M. Sughasiny, 
Assistant Professor, 
Department of Computer Science, 
Srimad Andavan Arts & Science College, Trichy. 
drsugha.2008@gmail.com

Abstract—Explosion in technology has led to an increase in the amount of online users, which served as a means for luring hackers. Though anomaly detection has been mainstream since the process of networking the varieties of attack vary and adaptations and upgrading has been observed not only from the customers point, but also from the attackers point. This paper presents a zero event anomaly detection in Big Data using Spark to enable real time processing for real time applications. This approach analyzed two algorithms, the Naïve Bayes algorithm and the Random Forest algorithm implemented using Spark to identify the efficiency of their working on imbalanced and huge data. Experiments were conducted on seven datasets representing all the categories of data in terms of size and imbalance and the results were analyze.

Keywords—Intrusion Detection; Classification; Naïve Bayes; Random Forest; Big Data; Spark.

I. INTRODUCTION

Data that is continuously being generated from machines, sensors from Internet of Things, mobile devices, network data traffic, and application logs contains valuable information. By fusing, analysing, and correlating these various data sets together in real-time, organizations are able to have end-to-end visibility to identify and detect situations relating to operational inefficiencies, opportunities to improve profitability, and reduce security threats. Timely detection of outliers or anomalies and performing causality analysis for automating corrective actions as well as preventing future situations is very important [1]. Some important use cases include Malware Analysis, Botnet Analysis, Network Failure Detection, Dynamic Load Balancing, Intrusion Detection etc [2].

Traditional IT monitoring approaches automatically analyze less than one percent of the data available looking for 'known bad' behaviours and other Key Performance Indicators (KPIs). When a problem is found, an alert is raised that provides information on what happened and the troubleshooting teams then have to manually 'mine' the other 99% of the data to find out why there was an alarm in the first place [3]. The increasing amount and variety of sensors deployed across various Internet of Things applications presents a significant challenge [4].

Hadoop for Big Data is one of the most recent solutions provided to the data analysis community by the technologists. Hadoop operates on the Hadoop Distributed File System (HDFS) that enables the storage of large amount of data on a distributed manner. MapReduce is used to operate on the stored data. The major problem in Hadoop is the usage of secondary storage devices (hard disk) for all the intermediate operations. This increases the latency of the application to a large extent. Though HDFS enables storage of large amounts of data, processing them at the speed requirements for an intrusion detection system is not practically feasible.

The proposed anomaly detection system will provide the organizations effective insight into their entire environment, discovering unusual patterns, events, rates and timelines that in isolation would be lost in the huge data. The goal of this anomaly detection is to discover previously unknown events from big data at the earliest time to provide fast and accurate detection rates.

II. RELATED WORKS

Intrusion detection has been a longstanding topic, since the inception of networks. However as the networks and the detection mechanisms evolve, so did the intrusions. Hence the techniques for intrusion detection are many, still several requirements linger incomplete. This section discusses several techniques used for intrusion detection in the recent times.
A. Intrusion Detection using Naïve Bayes

Naïve Bayes [21] is a probability based classifier that trains itself using the available data to predict results for unknown instances. It uses a conditional probability model that classifies instances represented of the form \( A = (a_1, a_2, \ldots, a_n) \) with \( n \) features to identify a class from the available set of classes \( C = (c_1, c_2, \ldots, c_m) \). The class assignment is carried out by identifying the probability of occurrence \( p \) for every class \( p(c_i | x_1, x_2, \ldots, x_n) \). The conditional probability is represented as,

\[
p(C_i | X) = \frac{P(C_i) \cdot P(X | C_i)}{\sum_{C_j \in C} P(C_j) \cdot P(X | C_j)}
\]

This method calculates the probability of occurrence of every class for the current input instance. Hence every class has a probability associated with it corresponding to a particular instance. The class with maximum probability is
finally chosen as the final resultant class. Naïve Bayes is considered to be the simplest classifier due to the simplistic level of operations associated with it. This approach is still considered in our application due to its generic nature of operation that can incorporate all types of data (numerical and categorical). Further, it was also identified that Naïve Bayes exhibits low impacts on data with imbalance and its intrinsically parallelizable operating nature has made it a valid candidate for the intrusion detection domain. The current approach implements Naïve Bayes algorithm using Spark. Resilient Distributed Data (RDD) available in Spark makes the grouping operations effective and fast.

B. Intrusion Detection using Random Forest

The Random Forest classifier is an ensemble model that uses a collection of decision trees to process the data and provides the result aggregated from each of the trees as the final result. The first process of intrusion detection using Random Forest is to create subsets to be passed for each tree. Subset creation is carried out such that each of the decision trees is provided with at least 66% of the training data. This is to guarantee the presence of all the class representatives in each tree. This assures that none of the trees are biased in their classification process.

The next phase creates the actual decision trees and passes them with the appropriate data to begin the processing. The number of decision trees used depends on the parallelization capacity of the hardware architecture being used. Each decision tree identifies m predictor variables out of M total variables from the training instances (m<M). The best predictor variable is identified out of the m available variables and a decision split is performed on it. This phase marks the beginning of the tree creation. The process of predictor variable selection and tree splitting is carried out until exhaustion and the final decision tree is created. The process of selecting m predictors from M total attributes is carried using the Random splitter selection method where m=M. The Breiman suggest one of the three possible ways of selecting m namely, \( \frac{1}{2} \sqrt{m}, \sqrt{m} \) or \( \frac{m}{2} \). Each of the decision trees provide different set of rules, hence they are aggregated to obtain the final classification rules. Since aggregation is required in the last phase, the trees do not perform attribute pruning. This classifier works on the basic principle that several weak classifiers can combine together in operation to become a strong classifier.

IV. DATASET DESCRIPTION

Experiments were carried out on seven standard datasets namely glass1, glass5, E-coli, Yeast1, Yeast5, vehicle0 and KDD [25]. KDD was obtained from the UCI Repository [21] and all the other datasets were obtained from the Keel Repository [22]. Each of the dataset was selected based on its size and the imbalance level contained in it. It could be observed from Table-1 that the size of the dataset and their imbalance levels are varied gradually from low to moderate. KDD was selected as a representative for huge dataset with high level of imbalance. The datasets contain two classes, as the required application of anomaly detection requires binary classes. The KDD dataset is also converted to binary classification dataset and is used for the process.

| TABLE-1. Dataset Description |
|---|---|---|---|---|
| Data Set | No of attributes | No of instances | No of Classes | Imbalance Level | Data Size |
| Glass1 | 9 | 214 | 2 | 1.82 | 1926 |
| Glass5 | 9 | 214 | 2 | 22.78 | 1926 |
| E-coli | 7 | 336 | 2 | 15.8 | 2352 |
| Yeast1 | 8 | 1484 | 2 | 2.46 | 11872 |
| Yeast5 | 8 | 1484 | 2 | 32.73 | 11872 |
| Vehicle0 | 18 | 846 | 2 | 3.25 | 15228 |
| KDD | 26 | 345719 | 2 | 3274 | 8998694 |

V. COMPARISON ANALYSIS

A comparative analysis as carried out between Naïve Bayes and Random Forest classifiers to identify the appropriate classifier that can handle the specific requirements put forward by the current application namely, speed, handling imbalance and prediction accuracy. Figure-1 represents the accuracy comparison of Naïve Bayes and Random Forest classifiers. It could be observed from the figure that the Random Forest classifier operates effectively compared to the Naïve Bayes classifier. The datasets are arranged in increasing order of their sizes, and it could be observed that in the initial stages of operating on less number of data, both Naïve Bayes and Random Forest operate with similar accuracy, as the data size increases, Random Forest exhibits a huge efficiency in terms of accuracy of classification. Hence it could be concluded that Random Forest classifier operates effectively in terms of accuracy of prediction.

VI. RESULT AND DISCUSSION

Results obtained from the Random Forest classifier pertaining to various metrics are discussed in this section. Figure-2 shows the F-Measure obtained from the Random Forest classifier. Though fluctuations exist in the F-Measure values, it could be observed that most of the data sets exhibit...
F-Measures at acceptable levels, while large dataset vehicle0 and KDD exhibit very high F-Measure levels >0.95.

**FIGURE 2: F-Measure**

![F-Measure Diagram]

**FIGURE 3: Classification Level**

<table>
<thead>
<tr>
<th>Classification Level</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD</td>
<td>0</td>
</tr>
<tr>
<td>Yes-ID</td>
<td>5</td>
</tr>
<tr>
<td>Yes-ID</td>
<td>3</td>
</tr>
<tr>
<td>No-ID</td>
<td>47</td>
</tr>
<tr>
<td>No-ID</td>
<td>4</td>
</tr>
<tr>
<td>Vehicle0</td>
<td>2</td>
</tr>
<tr>
<td>Vehicle0</td>
<td>42</td>
</tr>
</tbody>
</table>

**FIGURE 4: ROC**

![ROC Diagram]

**FIGURE 5: PR Curve**

![PR Curve Diagram]

Figures 4 and 5 present the ROC and PR plots for the datasets. Each point in the plots correspond to the aggregated value of a single data set. It could be observed from the ROC curve, that the Random Forest classifier exhibit very high true positive rates, while the KDD dataset has high false positive rates, while all other datasets exhibit very low false positive rates <0.2. Effective precision and recall rates exhibited by the Random Forest classifier can be observed from Figure. This depicts the efficiency of the classifier of huge and imbalanced data.

**VII. CONCLUSION**

This paper presents an effective in-memory processing technique that can be used for intrusion detection in large data. Usage of Spark has satisfied all the constraints and provided increased speed and efficiency to the detection process. It is also observed to handle imbalance effectively. Hence it could be concluded that Spark based Random Forest classifier provides a scalable system for intrusion detection. This approach enables to create automatic dynamic baselines for millions of data points per minute and thereby allows the organization to monitor a larger set of data rather than just the KPIs. It enables to create a Machine Learning based Anomaly Detection that helps eliminate the need for setting various thresholds and rules based on various KPIs. Using this system can detect issues earlier and respond faster which enables troubleshooting teams to resolve issues before they impact large numbers of internal or external customers.

**REFERENCES**


