



**KADIR HAS UNIVERSITY**  
**SCHOOL OF GRADUATE STUDIES**  
**PROGRAM OF ELECTRONIC ENGINEERING**

**LOG ANALYSIS WITH ANOMALY DETECTION**

**UYGAR ŞAHİN**

**MASTER'S THESIS**

**İSTANBUL - AUGUST, 2019**



# LOG ANALYSIS WITH ANOMALY DETECTION

UYGAR ŞAHİN

MASTER'S THESIS

Submitted to the Graduate School of Science and Engineering of  
Kadir Has University in partial fulfillment of the requirements for the degree of  
Master of Science in Electronics Engineering

İSTANBUL, AUGUST, 2019

DECLARATION OF RESEARCH ETHICS /  
METHODS OF DISSEMINATION

I, UYGAR ŞAHİN, hereby declare that;

- this master's thesis is my own original work and that due references have been appropriately provided on all supporting literature and resources;
- this master's thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- I have followed *Kadir Has University Academic Ethics Principles prepared in accordance with The Council of Higher Education's Ethical Conduct Principles*.

In addition, I understand that any false claim in respect of this work will result in disciplinary action in accordance with University regulations.

Furthermore, both printed and electronic copies of my work will be kept in Kadir Has Information Center under the following condition as indicated below:

- The full content of my thesis will be accessible only within the campus of Kadir Has University.

UYGAR ŞAHİN

28.08.2019



KADİR HAS UNIVERSITY  
SCHOOL OF GRADUATE STUDIES

**ACCEPTANCE AND APPROVAL**

This work entitled LOG ANALYSIS WITH ANOMALY DETECTION prepared by UYGAR ŞAHİN has been judged to be successful at the defense exam on 28.08.2019 and accepted by our jury as Master's Thesis

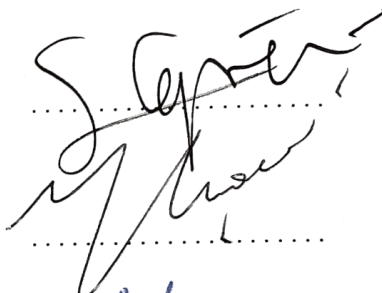
APPROVED BY:

Asst. Prof. Dr. Arif Selçuk Öğrenci (Advisor)  
Kadir Has University

Asst. Prof. Dr. Yalçın Şadi  
Kadir Has University

Asst. Prof. Dr. Figen Özen  
Haliç University

I certify that the above signatures belong to the faculty members named above.

  
.....  
Prof. Dr. Sinem Akgül Açıkmeşe  
Dean of School of Graduate Studies  
DATE OF APPROVAL: 28.08.2019

# TABLE OF CONTENTS

ABSTRACT .....	i
ÖZET .....	ii
ACKNOWLEDGEMENTS .....	iii
LIST OF TABLES .....	iv
LIST OF FIGURES .....	v
1. INTRODUCTION .....	1
2. PROBLEM DEFINITION .....	3
2.1 What is an Anomaly? .....	3
2.2 Type of Anomalies .....	3
2.2.1 Point Anomaly .....	3
2.2.2 Contextual Anomaly .....	4
2.2.3 Collective Anomaly .....	5
2.3 Anomaly Detection .....	7
3. METHODOLOGY .....	8
3.1 Anomaly Detection Algorithms .....	8
3.1.1 Supervised Methods .....	8
3.1.2 Semi-Supervised Methods .....	8
3.1.3 Unsupervised Methods .....	9
3.2 Steps of Data Analysis .....	9
3.2.1 Setting the Goal .....	10
3.2.2 Data Preparation .....	10
3.2.3 Data Transformation .....	11
4. DATASET .....	13
4.1 About the Dataset .....	13
4.2 Data Processing .....	13
4.3 Performance Evaluation Metrics .....	16
4.3.1 Precision .....	16
4.3.2 Recall .....	16

4.3.3 F1 Score . . . . .	17
5. ALGORITHMS AND EXPERIMENTS . . . . .	18
5.1 Log Clustering . . . . .	18
5.2 Isolation Forest . . . . .	18
5.3 Support Vector Machine . . . . .	19
5.4 Desision Tree . . . . .	19
5.5 Result of Experiments . . . . .	19
6. CONCLUSIONS . . . . .	21
APPENDIX A: LOG PARSING . . . . .	22
APPENDIX B: HDSF_Template.csv . . . . .	25
APPENDIX C: CREATION OF STRUCTURED DATA . . . . .	27
REFERENCES . . . . .	29

# LOG ANALYSIS WITH ANOMALY DETECTION

## ABSTRACT

Detection of anomalies in the data is an important data analysis job for server logs as they will reveal many benefits. Different types of methods can be used for anomaly detection: supervised, semi-supervised, and supervised anomaly detection. Similarly different algorithms exist for each category. In this work, four anomaly detection algorithms are utilized and their performance metrics are compared for a public Hadoop Distributed File System (HDFS) data. Among the others, the support vector machines are identified as the best method for anomaly detection.

**Keywords:** Anomaly, anomaly detection, log analysis, SVM, supervised and unsupervised methods

# ANORMALLİK TESPİTİYLE KÜTÜK ÇÖZÜMLEMESİ

## ÖZET

Sunucu kütükleri için veride anormallik yakalama getireceği faydalar sebebiyle çok önemli bir veri işleme görevidir. Anormallik yakalama için farklı türde yöntemler kullanılabilir: gözetimli, yarı gözetimli ve gözetimsiz. Benzer şekilde, her bir tür için farklı yöntemler bulunmaktadır. Bu çalışmada, herkese açık bir Hadoop Dağıtık Dosya Sistemi (HDFS) verisi için dört adet anormallik yakalama yöntemi kullanılmış ve başarımları kıyaslanmıştır. Ötekilerin yanında SVM anormallik yakalamada en başarılı yöntem olarak ortaya çıkmıştır.

**Anahtar Sözcükler:** Anormallik, anormallik yakalama, kütük çözümlemesi, SVM, gözetimli ve gözetimsiz yöntemler



## ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Asst. Prof. Dr. Arif Selçuk Öğrenci for the continuous support of my thesis study and research; for his patience, motivation and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Master's thesis study.

I would like to thank to my family for excellent support. I would like to express my special thanks to my wife, for limitless patience and support during this thesis.

## LIST OF TABLES

Table 4.1	HDFS Dataset Information . . . . .	13
Table 5.1	Results for the Training Set . . . . .	20
Table 5.2	Results for the Test Set . . . . .	20

## LIST OF FIGURES

Figure 2.1	Type of anomalies . . . . .	4
Figure 2.2	A simple example of point anomalies in a 2-dimensional data set.	4
Figure 2.3	Contextual anomaly. . . . .	5
Figure 2.4	Contextual anomaly in HDFS log data. . . . .	5
Figure 2.5	Collective anomaly in ECG output. . . . .	6
Figure 2.6	Collective anomaly in HDFS log data. . . . .	6
Figure 2.7	Conceptual taxonomy of anomaly detection. . . . .	7
Figure 3.1	Anomaly detection approaches. . . . .	9
Figure 4.1	Implementation Flow . . . . .	14

## 1. INTRODUCTION

The information age has witnessed the widespread use of computers for a variety of jobs in every sector. Enterprise computing systems are designed to meet high levels of service requirements. Those requirements can only be met by an appropriate cooperation of hardware and software components. Servers are dedicated hardware units that serve computing platforms. Usually, servers are built to work in a fault tolerant manner where the software components (both the operating system and the application software) are also designed with high expectations. Despite the advances in technology, faults and errors are inevitable due to several reasons. Firstly, hardware components suffer from fatigue and random errors. Secondly, software may contain bugs. Beyond these factors, modern computer systems are under attack of hackers by means of several methods. As a consequence, all computers track the events occurring in their hardware and software components including the network. Those events are recorded as digital logs which can be investigated so as to make a forensic analysis.

Nowadays, applications and servers produce log data in an enormous amount. As the Internet grows and communication between computers constitute an integral part of server activities, log data is structured into several categories such as processing, operating system, database, networking, etc. Logging data is a normal daily routine to register many events in the server and applications [1]. Therefore, a problem in applications and servers can be found by analyzing these log data. A lot of information is available in this log data. But this analysis is difficult even impossible to do manually. On the other hand, real time event recognition is essential to prevent data loss or hacking activities. However, real time log data analysis is almost impossible in online systems with the data stream rates of today. Therefore,

there is a need for a structured methodology to do this analysis quickly [2].

Abnormalities of data in the logs can be an effective indicator for taking action as a critical event may have happened. Hence, anomaly detection becomes an important necessity in the computing industry. Anomaly detection supplies correct and understandable evaluation of the data. Anomaly detection is used to identify unusual patterns in the rest of the data. There are many different names of anomalies in the literature such as outliers, exceptions, peculiarities, etc. All these terms can be used interchangeably as the common point is deviation from the normal operation. However, the term outlier is most commonly encountered and used in the literature. There are three different types of anomaly in the literature. Firstly, the point anomaly is generally called outliers anomaly. In point anomaly, a single instance of data is anomalous if it's too far off from the rest. Second one is contextual anomaly that is common in time-series data. Third type of anomaly is collective anomaly. This type of anomaly appears as a set of data instances collectively indicate an anomaly [3].

Based on the extent to which the labels are available, anomaly detection techniques can operate in one of the following three modes: supervised, semi-supervised and unsupervised [4]. Supervised techniques assume that data instances are labeled as normal or abnormal, that is, data is divided into two classes. On the other hand unsupervised techniques don't necessary have labeled data instances. They directly learn from unlabeled data instances. Semi-supervised techniques need labelled data instances for the normal class. Those three techniques used in machine learning are also applicable for anomaly detection which is the main topic of this thesis. Different anomaly detection approaches will be investigated for log data analysis. The next chapter will define the anomaly types. Next, the methods of anomaly detection will be highlighted and the real life dataset to be used, will be explained. The thesis will conclude after the results of the experiments are analyzed.

## **2. PROBLEM DEFINITION**

### **2.1 What is an Anomaly?**

Anomalies are unexpected behaviors or previously unseen patterns in the outcomes of a process. Regarding server systems, an anomaly can be identified by an unusual data stream that does not appear in usual, normal operation. Abnormalities can occur due to many reasons such as hardware faults, energy surges, software bugs, and last but not least by malicious acts of hackers. Detection of those abnormal behavior in the systems can be utilized for prevention of malware, network intrusion, industrial damage, and faults in general. Video surveillance, manufacturing monitoring systems, financial institutions tracking transactions, network analyzers etc. are some of the systems where anomaly detection is applied. Detection of anomaly is within the research area of many disciplines such as statistics, machine learning, information theory, and data mining [5, 6].

### **2.2 Type of Anomalies**

Anomalies are divided into three general groups as can be seen in Figure 2.1. Each of them will be explained below.

#### **2.2.1 Point Anomaly**

This type of anomaly is also called a global outlier. Point anomaly is an individual data instance which differs from the rest of the data. For example, an instant change in credit card spending may indicate the possibility of fraud or that the credit card is being stolen. It is a point anomaly to make an expenditure in a distant place than the

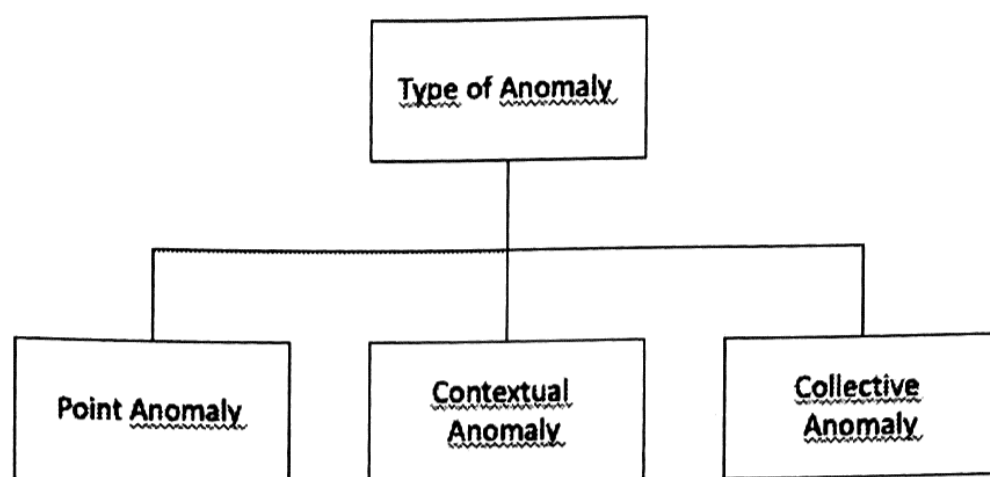


Figure 2.1 Type of anomalies

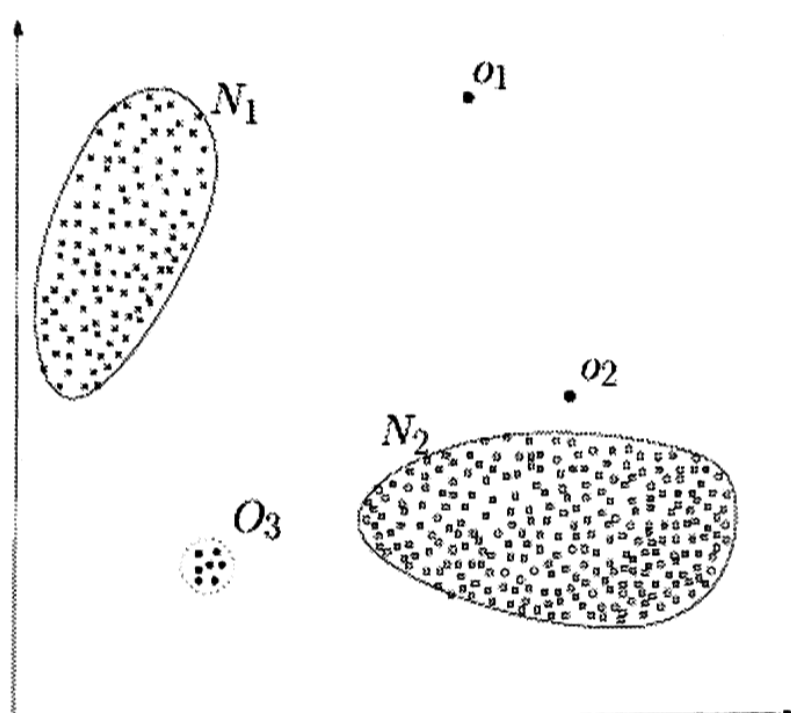


Figure 2.2 A simple example of point anomalies in a 2-dimensional data set.

routine expenses are made from credit card. Figure 2.2 displays a generic example of point anomaly. Point anomalies can be identified and fixed quickly provided that there exists enough data about the past history. Moreover, the clusters (types of normal behavior) have to be separated from each other. A major risk in point anomaly detection is the so called false positive where an abnormal event actually represents a valid change in the behavior.

### 2.2.2 Contextual Anomaly

The abnormality is defined with respect to the context where the data appears. These type of anomalies are commonly observed in time series data. In Figure 2.3 a periodicity can be seen in the context. In this graph, point t1 is an example of

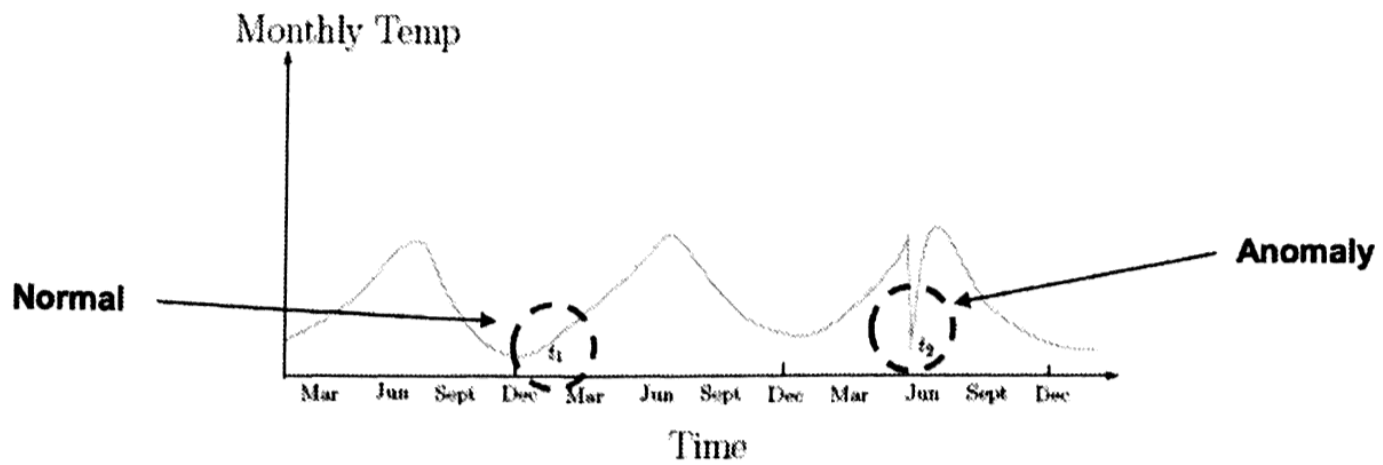


Figure 2.3 Contextual anomaly.

```

19 INFO dfs.FSDataset: Deleting block blk_-8213344449220111733 file /mnt/hadoop/dfs/data/current/subdir39/blk_-8213344449220111733
19 INFO dfs.FSDataset: Deleting block blk_-6899869435641005946 file /mnt/hadoop/dfs/data/current/subdir12/blk_-6899869435641005946
19 INFO dfs.FSDataset: Deleting block blk_-8191677345482862686 file /mnt/hadoop/dfs/data/current/subdir43/blk_-8191677345482862686
19 INFO dfs.FSDataset: Deleting block blk_-919439116365725304 file /mnt/hadoop/dfs/data/current/subdir30/blk_-919439116365725304
143 INFO dfs.DataNode$DataXceiver: Received block blk_-1608999687919862906 src: /10.251.215.16:52002 dest: /10.251.215.16:50010 of
19 INFO dfs.FSDataset: Deleting block blk_459350181734707719 file /mnt/hadoop/dfs/data/current/subdir26/blk_459350181734707719
19 INFO dfs.FSDataset: Deleting block blk_4740256457847790996 file /mnt/hadoop/dfs/data/current/subdir26/blk_4740256457847790996
19 INFO dfs.FSDataset: Deleting block blk_4827511386537849489 file /mnt/hadoop/dfs/data/current/subdir26/blk_4827511386537849489
19 INFO dfs.FSDataset: Deleting block blk_-5679405126602834227 file /mnt/hadoop/dfs/data/current/subdir26/blk_-5679405126602834227
19 INFO dfs.FSDataset: Deleting block blk_-6029280107803421476 file /mnt/hadoop/dfs/data/current/subdir26/blk_-6029280107803421476

```

Figure 2.4 Contextual anomaly in HDFS log data.

normal behavior but point  $t_2$  is an anomaly, because it is different from the periodic context as it deviates abruptly. As can be seen in Figure 2.4, the 'received block' log may not be abnormal in the HDFS log data used in the thesis. However, if this context occurs after 30 deletions, this indicates an anomaly. Hence the existence of an anomaly is heavily dependent on the context [7].

### 2.2.3 Collective Anomaly

Collective anomaly is a group of anomalous values in the data set. If a collection of related data instances is anomalous with respect to the entire dataset, but not among themselves (as individual values) then this is called a collective anomaly. Breaking rhythm in ECG and web attacks are some examples of collective anomaly as can be seen in Figure 2.5. Collective anomaly in HDFS log data is presented in Figure 2.6 . While appending to an "invalid set" may not be an anomaly, several similar sequential appends to an "invalid set" may define a collective anomaly in HDFS log data [8].



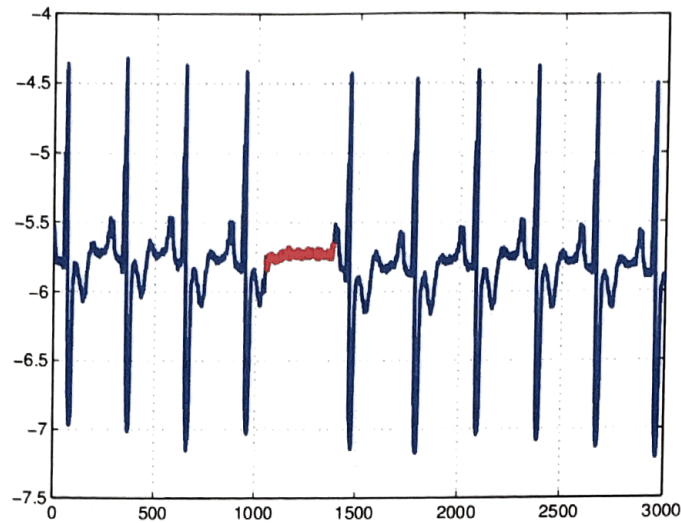


Figure 2.5 Collective anomaly in ECG output.

```

INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-7731904648499906111 is added to invalidSet of 10.251.202.209:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_5974111262061307773 is added to invalidSet of 10.250.9.207:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_681039239152659399 is added to invalidSet of 10.250.14.38:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_543049469610993836 is added to invalidSet of 10.251.39.179:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-5911878273886750645 is added to invalidSet of 10.250.11.53:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-2807193698972165323 is added to invalidSet of 10.251.199.150:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-3987477245521920562 is added to invalidSet of 10.250.6.214:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-7969924975975364418 is added to invalidSet of 10.251.43.210:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-1969860458329121257 is added to invalidSet of 10.250.15.101:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_5013424830919192724 is added to invalidSet of 10.251.203.179:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-5114399346914160800 is added to invalidSet of 10.251.123.99:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_8648769525804190870 is added to invalidSet of 10.251.89.155:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_8899521512761756651 is added to invalidSet of 10.251.107.50:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_-8072054884795354988 is added to invalidSet of 10.251.126.227:50010
INFO dfs.FSNamesystem: BLOCK* NameSystem.delete: blk_8093422814437206016 is added to invalidSet of 10.250.15.101:50010

```

Figure 2.6 Collective anomaly in HDFS log data.

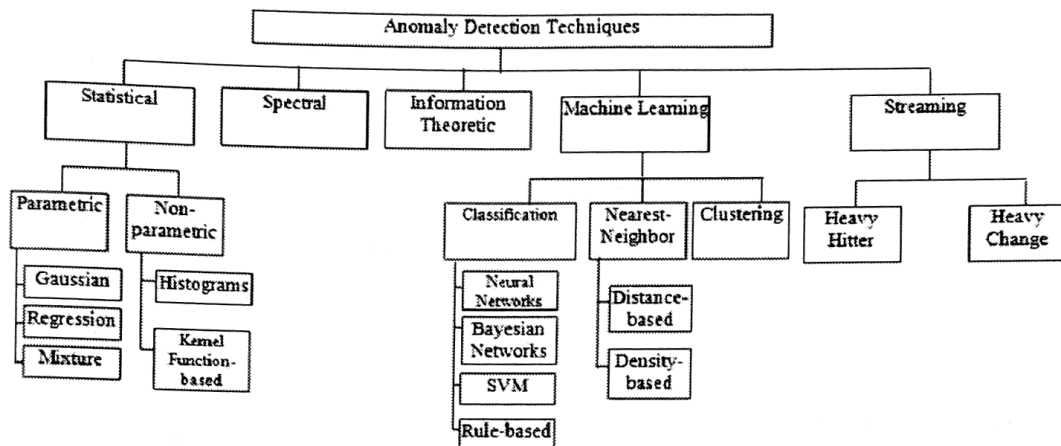


Figure 2.7 Conceptual taxonomy of anomaly detection.

### 2.3 Anomaly Detection

Anomaly detection can be defined as finding the different patterns in a data set. Such data points are called anomalies in machine learning, and outliers in statistics. These nonconforming patterns can be difficult to detect. Anomaly detection is used in a wide variety of applications such as fraud detection on credit cards, fault detection of server systems, health care, cyber-security, and enemy activities on borders. The taxonomy of anomaly detection is given in Figure 2.7 where the majority of efforts have been spent in statistical and machine learning algorithms.

## 3. METHODOLOGY

### 3.1 Anomaly Detection Algorithms

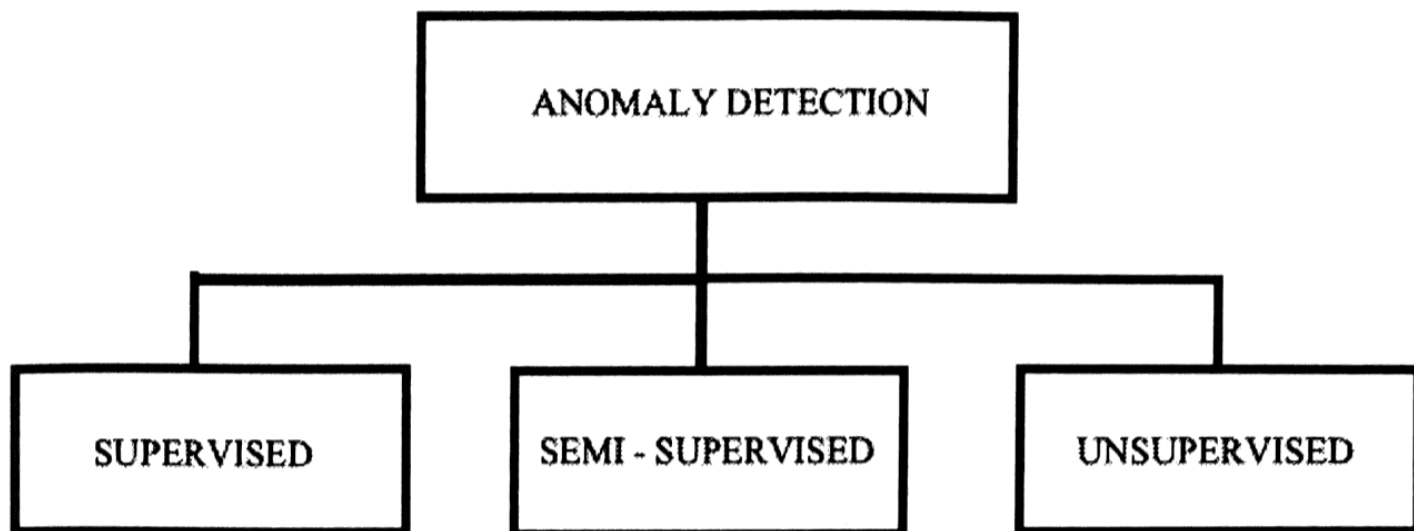
Anomaly detection can be regarded as a subset of a machine learning task which deals with a large data set. Depending on the existence of a "teacher" signal in the data, the learning methods can be classified as follows (Figure 3.1).

#### 3.1.1 Supervised Methods

Supervised anomaly detection methods require a labeled dataset. Labeled dataset means that each instance is defined as normal or abnormal dividing the whole data into two classes. Any unknown data instance is compared to the model to determine if it belongs to the normal class or the abnormal class. A supervised learning algorithm analyzes the training data and produces an inference function, which is called a classifier or a regression function [9]. Logistic regression, decision tree, and SVM (Support Vector Machine) are examples of commonly used supervised anomaly detection methods [10]. One of the advantages of these methods is that they ensure reliable results when using a suitably large training data set. Another advantage is related to the testing process. The success of a model trained with a labeled data set can be predicted. The major disadvantage of supervised methods is that labelling is sometimes deficient to detect anomalies [11].

#### 3.1.2 Semi-Supervised Methods

Semi-supervised anomaly detection method is halfway between supervised method and unsupervised method. Semi-supervised method assumes that the training dataset



**Figure 3.1** Anomaly detection approaches.

consists of labeled instances for the normal class, but the training dataset does not require labeled anomaly instances. This means that semi-supervised methods will only use normal instances to detect anomalies [12].

### **3.1.3 Unsupervised Methods**

Unsupervised anomaly detection is one of the hardest types of anomaly detection because there is no labeled dataset. The data set is not labeled at all, only the inputs are present where there is no information about the output. This method is used in real life applications because it is more applicable than the supervised method as for many cases the different types of anomalies cannot be defined before they actually occur. Many researchers work on unsupervised methods such as log clustering, PCA (Principal Component Analysis), and invariant mining to develop algorithms [13, 14].

## **3.2 Steps of Data Analysis**

Several steps of data analysis are required to perform an efficient and effective machine learning task. They can be listed as follows:

### **3.2.1 Setting the Goal**

For the given data set, the aims of the anomaly detection process and the targets for performance have to be identified in order to accomplish the subsequent tasks efficiently. This includes analysis of the problem domain, specification of anomaly types and their occurrence frequencies, and determining parameters of the model which is a possible generating process for the stochastic system. For the machine learning task, the labeled log data of a Hadoop file system has been selected where multiple algorithms are used to develop an anomaly detection system.

### **3.2.2 Data Preparation**

The possibly big data exhibits numerous errors and inconsistencies which have to be corrected. Sometimes the correction is impossible where data have to be discarded or repaired depending on the need. Data preparation is an essential step for an accurate training of the machine learning system. This step is divided into subtasks as follows. Firstly, data have to be collected from the available sources. This task specifies the amount of data to be collected which should be as large as possible because machine learning algorithms perform better if there are more data. An important decision has to be taken if data are segmented as this decision is effective on the models generated later. Depending on the goals of the anomaly detection process, all the attributes have to be included in the data set which seem to have an impact on the outcome.

Then, data preprocessing will be carried out to transform the raw data (server log data in our case) into a form that is suitable to be processed by the system. Formatting of raw data will allow standardization of the attributes so that data can be transformed into information. Next, data will be cleaned where inconsistencies are identified and resolved. Missing data are either removed or interpolated. This requires extreme attention as missing or no-care values may affect system performance heavily. Lastly, data records in a standard format are sampled to reduce

the processing complexity. This can be done in two ways, either by down sampling records to remove repeating elements in the data set, or by down sampling attributes by principal component analysis where uncorrelated attributes (attributes without effect on outcome) are removed.

### **3.2.3 Data Transformation**

Collected and cleaned data will be processed further to fit the requirements of the algorithms to be employed. This includes steps such as data scaling (normalization), feature selection and feature generation (combining attributes or deriving new attributes based on available ones).

Anomaly detection can be regarded both as a supervised or unsupervised process. For our case, the data set is manually labeled, that is, each row (or groups of rows) of data represents a normal or anomalous state. Hence supervised machine learning algorithms can be employed for anomaly detection as we have both the input and the desired output. In this research, we will apply multiple supervised learning techniques such as decision trees, isolation forest, and support vector machines. Also the log clustering method will be utilized. Further details of the architecture will be explained in subsequent sections.

The novel approach to be employed in this work is to apply unsupervised methods such as clustering and supervised methods for forming a decision support system for anomaly detection. The clustering algorithm will divide the samples into a predefined number of clusters based on their similarities. The algorithm (k-means for benchmarking) will be run for different number of clusters and each cluster will be analyzed for the proportion of anomalies. In case, the anomalous data samples are predominantly clustered (more than a threshold value such as 90 percent) in one class, then this information can guide us to label the points clustered here as anomalous. Furthermore, autoencoder network can be used as a neural network that aims to learn its inputs as its outputs. That is, the desired outputs for an

autoencoder is the input supplied. The network can be trained to mimic the inputs at the output. Usually there will be three hidden layers of neurons where the first and third hidden unit will have a larger number of neurons (between  $n$  and  $2n$  for  $n$  attributes at the input) and the second hidden layer will have smaller count of hidden units so as to represent the essential features of the input space. The output error will be used as an indicator for anomaly: If the error for  $z$  certain sample is larger than the threshold value (usually taken as a multiple of the average error) then, that specific input sample will be considered as an anomaly because the network could not encode it automatically.

In short, the work will include both supervised and unsupervised methods to form a combined score for the samples. As usual, the data set will be divided into training, validation and test sets where ratios of 70, 15 and 15 per cent will be used respectively. The ultimate aim is to increase accuracy above 90 percent whereas the false alarm rate will be kept below one percent.

## 4. DATASET

### 4.1 About the Dataset

This study is related to anomaly detection in the log data of a Hadoop server. Because of data privacy and security issues, it is quite difficult to obtain real world data in spite of the fact that there are thousands of live production systems generating log data [15, 16]. By means of an extensive literature search, a public log dataset has been identified (HDFS: Hadoop File System, a distributed filesystem) which would enable us to carry out analysis of anomaly detection using machine learning algorithms.

The HDFS dataset has been obtained from a production Amazon EC2 system and it is well suited for anomaly detection. The dataset has 11,175,629 log messages with 16,838 anomalies labeled by some domain experts which will be useful for identifying correctness of the algorithms employed [17, 18]. There are 575065 blocks in the HDFS log data of which 16836 are labeled as abnormal. The standard block size is 128MB on Amazon EC2 servers. The HDFS log data has a unique block ID for each block operation such as allocation, writing, replication, and deletion. Further details about the dataset are given in Table 4.1.

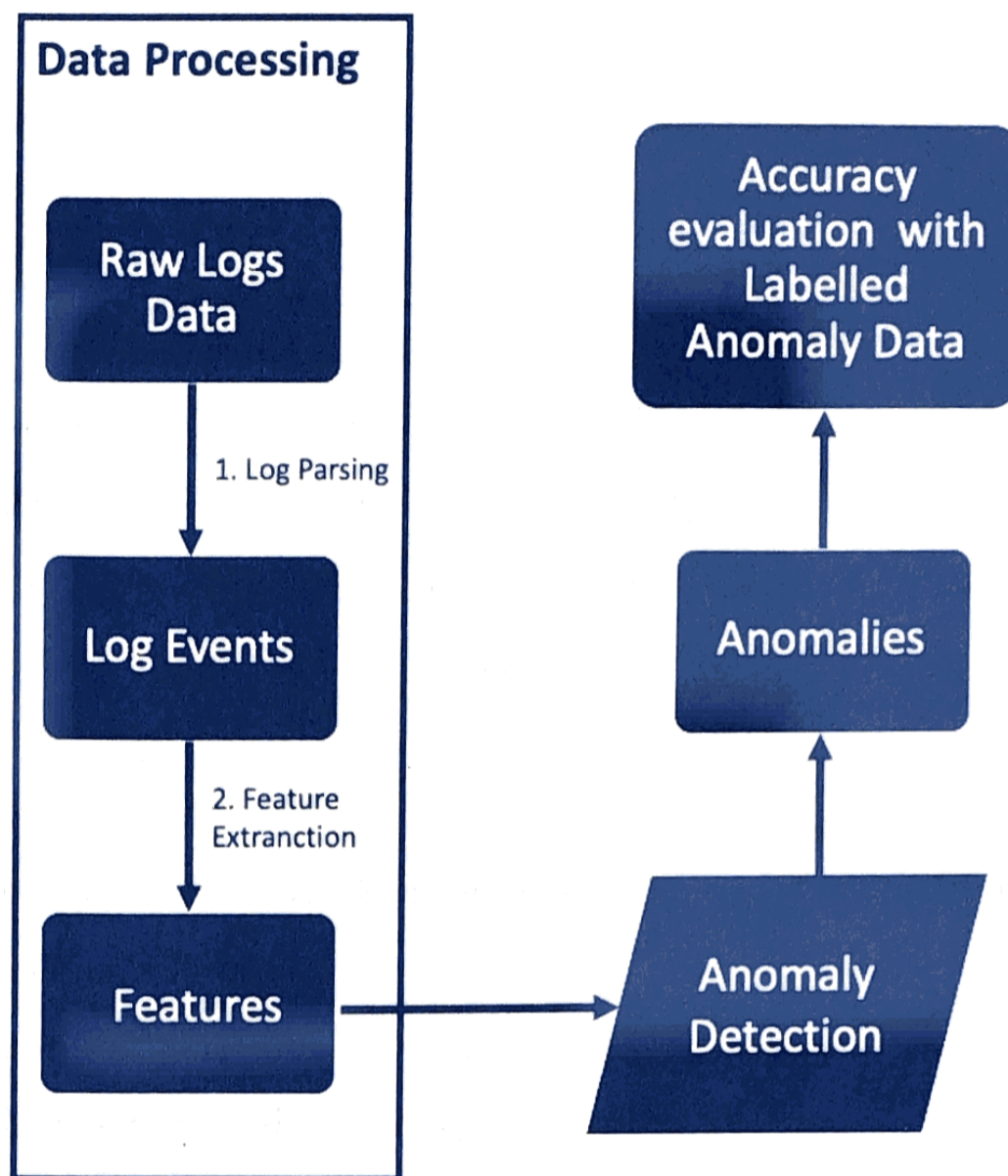
Application	Range	Dataset Size	Log Messages	Anomalies
HDFS	38.7 hours	1.47GB	11,175,629	16,838

Table 4.1 HDFS Dataset Information

### 4.2 Data Processing

The first step is data processing which means that long, raw log data are transformed into features that can be used by anomaly detection algorithms. The raw log data





**Figure 4.1** Implementation Flow

is unstructured data and it is used as input for the log parsing process. Log parsing process removes application specific details or unrelated data from raw data [19]. Output of log parsing is used as input to the feature extraction process. In this part, parsed log data are converted to structured data with numerical features. Output of this step is used as input for anomaly detection algorithm [17]. Anomalies are then cross-validated with the domain expert labeled list of anomalies for each dataset to identify false positives, false negatives, true positives and true negatives in order to derive precision, recall, and F1 score metrics as will be explained below. Figure 4.1 summarizes the data processing steps.

Application log records contain unstructured time series data. This log data need be to converted to structured data. It is required to consolidate anomalies into single events based on common criterion such as the block number for HDFS. Raw log data file consists of raw log messages. Each raw log message has two parts: a constant part and a variable part [19]. The constant part is a fixed plain text and it is not

contain runtime variables. The variable part contains runtime variables such as IP address, log type (INFO), and thread ID (145) as given below.

```
081109 203524 145 INFO dfs.DataNode$PacketResponder:
PacketResponder 2 for block blk_-9073992586687739851 terminating
081109 203524 145 INFO dfs.DataNode$PacketResponder:
Received block blk_-9073992586687739851 of size 11977 from /10.250.19.102
081109 203552 13 INFO dfs.DataBlockScanner: Verification succeeded for blk_-
9073992586687739851
```

After the log parsing process, data become ready to act as input to the feature extraction phase as follows.

```
081109 203524 : PacketResponder * for block * terminating
081109 203524 : Received block * of size * from *
081109 203520 : Verification succeeded for blk *
```

As the second step, once all log events for a particular block are isolated, the frequency of these events are counted in order to create an event count vector. For statistical analysis, the data must be numerical. Therefore, it is necessary to create event counts or a feature matrix which consists of feature matrix vectors. These vectors are called event vectors or feature vectors. Each feature vector gives the number of unique occurrences in the set of variables. This means, for block -9073992586687739851 the event count vector is like [ 2,2,3,0,0,0,0,0,0,0,...]. The event vector (numerical features) represent two packet responder events, two received block events and three verification succeeded events for block -9073992586687739851. The zeros in the event count vector mean that no events did occur for block -9073992586687739851. This step is repeated using the session window for log data sets. Session windows are based on identifiers, they are not based on timestamps. Identifiers are utilized to specify dissimilar paths in some log data. For example, information such as write, delete, copy, allocation in HDFS logs is saved with block\_id.

Thus, descriptors, groups in which each session window has a unique identifier, can be created. After that, the event count vector is created for input for the anomaly detection algorithms [20].

### 4.3 Performance Evaluation Metrics

This study will compare the accuracy of anomaly detection algorithms in HDFS log data. Three metrics are used to evaluate the performance of anomaly detection algorithms as described below [21].

#### 4.3.1 Precision

Precision shows us knowledge about a model's performance with relation to false positives. This means, precision is the ratio of anomalous events that have been correctly identified divided to the sum of true positives and non-anomalous events that have been incorrectly identified as an anomaly [22].

For example in email spam detection, a false positive means that a non-spam email (actual negative) is defined as spam email (predicted spam). This email will not be in the user's inbox, so the user won't see an important email if the precision is not high for the spam detection model. Equation 4.1 gives the formula of precision.

$$Precision = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (4.1)$$

#### 4.3.2 Recall

Recall show us knowledge about a model's performance with relation to false negatives. Recall gives how much of the real anomalies are detected.

For example, in fraud detection, if a fraudulent transaction (Actual Positive) is predicted as non-fraudulent (Predicted Negative), the result can be bad for the bank. Equation 4.2 shows the formula of recall.

$$Recall = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (4.2)$$

### 4.3.3 F1 Score

F-measure or the F1 score represent the harmonic mean of the precision and recall values. F1 score is a measure of a test's accuracy. F1 score is the balance between precision and recall. When F1 score approaches 1, it gives the best result and when it approaches 0, it gives the worst result. Equation 4.3 shows the harmonic mean calculation.

$$F1 = 2 \times \frac{\textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}} \quad (4.3)$$

## 5. ALGORITHMS AND EXPERIMENTS

Four algorithms have been utilized and their performance have been compared for anomaly detection. The HDFS log data with 12 million lines are parsed in 2 days, 2 hours and 12 minutes using an Apple Mac Air computer with 8GB memory, 1.7 GHz Intel Core i7 processor and SSD disk. The algorithms are implemented using the Loglizer package [23], a Python library, on the structured HDFS log data obtained after parsing.

### 5.1 Log Clustering

Log clustering calculates the similarity value between two log sequences and implements the agglomerative hierarchical clustering technique to group the similar log sequences into clusters [24]. At the beginning of the agglomerative hierarchical clustering, each log sequence belongs to its own cluster. Then, the closest pair of clusters is selected and merged into a single cluster. To decide which pair of clusters should be merged, the distance metric between the clusters should be defined properly [25].

### 5.2 Isolation Forest

The isolation forest algorithm is among the most common algorithms. The algorithm is based on the fact that anomalies are few and occur at different data points. As a result of these features, anomalies are susceptible to a mechanism called isolation. This method is very practical and is dissimilar from other methods[26]. It introduces the use of isolation as a more effective and efficient method to detect anomalies rather than the commonly used basic distance and density measures. In addition, this

algorithm has a low linear time complexity and a small scale memory requirement. It builds a performing model with a minimum number of trees using small subsamples of fixed size, regardless of the size of the original data set.

### **5.3 Support Vector Machine**

Support vector machine (SVM) is a supervised classification algorithm. SVM tries to find the best line separating the two classes. The algorithm allows the line to be drawn between classes to pass the across elements farthest away. SVM can also classify linear and nonlinear data, but it is mostly used to classify the data in a linear fashion [27].

### **5.4 Decision Tree**

Decision tree is more suitable for outlier detection. This is one of the advantages with respect to other classification methods. In a decision tree, every node crumbles the feature space from its parent node into two or more separate subspaces and the root node splits the complete feature space. The tree building process selects at each node that split point, which divides the given subspace and the training data best according to some impurity measure [28].

### **5.5 Result of Experiments**

The performance of four anomaly detection approaches are compared in this study. These are log clustering, isolation forest, SVM and decision tree. Table 5.1 show a summary of anomaly detection algorithms for the training set of the data. For the HDFS log data, the decision tree algorithm performs best followed by the SVM algorithm.

After the training, the result of the test is shown in Table 5.2. SVM algorithm (Precision:0.814, Recall: 0.577, F1:0.675) beats the others.

Model	Precision	Recall	F1-measure
Log Clustering	1.000	0.438	0.610
Isolation Forest	0.984	0.460	0.627
Decision Tree	1.000	0.508	0.674
SVM	0.988	0.508	0.671

**Table 5.1** Results for the Training Set

Model	Precision	Recall	F1-measure
Log Clustering	0.793	0.505	0.617
Isolation Forest	0.781	0.467	0.584
Decision Tree	0.801	0.534	0.641
SVM	0.814	0.577	0.675

**Table 5.2** Results for the Test Set

As a result of the comparison, it can be stated that the SVM algorithm was found to be more successful than the other methods in detecting anomalies in HDFS log data.

In this study, Python code in Appendix A is used for the initial parsing of HDFS raw log data. After the first log parsing process, Python code in Appendix C is used for creating structured log data. The feature template in Appendix B is used to create structured log data.

## 6. CONCLUSIONS

The primary goal of this work is to combine supervised and unsupervised methods to form a decision support system for anomaly detection in server log data. A real life big data set obtained from the logs of a HDFS system is utilized to perform anomaly detection where several actions are regarded as anomalous such as repeated failed login attempts, ping attacks, etc. The processing of the big data is followed by the application of supervised and unsupervised machine learning efforts to learn the behaviours of the system. The results of the algorithms will be used as a combination to form a decision about the current state of the server. The detailed analysis of logs is still required even if the system creates an alert as those systems can never be trusted fully. The actual reason of an anomaly can be identified by analyzing the stream log data generated by the system as online and inline. As a future work, the algorithms can be tested on various similar server datasets so as to make a generalization.



## APPENDIX A: LOG PARSING

```
import sys
import re
import pandas as pd
import numpy as np
import datetime as dt

data_pattern = r"(\d+)\s+(\d+)\s+(\d+)\s+(\w+)\s+(.*?\:)\s(\b.*)
(\bblk_?\d+)(.*$)"
regex_obj = re.compile(data_pattern, re.VERBOSE)
hdfsfile = open("./HDFS.log", "r")
##### Data Frame's Columns define #####
date=[]
Time=[]
pid=[]
level=[]
component=[]
content=[]
eventId=np.nan
eventTemplate=np.nan
block=[]
label=np.nan

##### Main Log to CSV #####
start_time = dt.datetime.now()
for strLineRead in hdfsfile:
```

```

#— remove leading and trailing whitespace—
strLineRead = strLineRead.strip()

#— split the line into fields —
parsed_log = ""
parsed_log = regex_obj.search(strLineRead)

if parsed_log:

    date.append(parsed_log.group(1))
    Time.append(parsed_log.group(2))
    pid.append(parsed_log.group(3))
    level.append(parsed_log.group(4))
    component.append(parsed_log.group(5))
    content.append(parsed_log.group(6)+parsed_log.group(7)
+parsed_log.group(8))
    block.append(parsed_log.group(7))

#print (hdfsfile.readlines(5)[1].strip())
hdfsfile.close()
df=pd.DataFrame({'Date':date,
                 'Time':Time,
                 'Pid':pid,
                 'Level':level,
                 'Component':component,
                 'Content':content,
                 'EventId':eventId,
                 'EventTemplate':eventTemplate,
                 'Block':block,
                 'Label':label
                 })

```

```

df[['Date', 'Time', 'Pid', 'Level', 'Component', 'Content', 'Block',
'Label', 'EventId', 'EventTemplate']].t
o_csv('./HDFS_parsed.csv', encoding='utf-8', index=False)
print("First_log_parsing_end:{}".format(
end_time=dt.datetime.now() - start_time))

```

Output:

	Block	Component	Content	Date	EventId	EventTemplate	Label	Level	Pid	Time
0	blk_38865049064139660	dfs.DataNode\$PacketResponder:	PacketResponder 1 for block blk_38865049064139...	081109	NaN	NaN	NaN	INFO	148	203815
1	blk_-6952295868487656571	dfs.DataNode\$PacketResponder:	PacketResponder 0 for block blk_-6952295868487...	081109	NaN	NaN	NaN	INFO	222	203807
2	blk_7128370237687728475	dfs.FSNamesystem:	BLOCK* NameSystem.addStoredBlock: blockMap upd...	081109	NaN	NaN	NaN	INFO	35	204005
3	blk_8229193803249855061	dfs.DataNode\$PacketResponder:	PacketResponder 2 for block blk_82291938032499...	081109	NaN	NaN	NaN	INFO	308	204015
4	blk_-6670958622368987959	dfs.DataNode\$PacketResponder:	PacketResponder 2 for block blk_-6670958622368...	081109	NaN	NaN	NaN	INFO	329	204106

## APPENDIX B: HDSF\_Template.csv

EventId ,RegContent

E1,(Adding an already existing block blk\_.\*\$)

E2,(Verification succeeded for blk\_.\*\$)

E3,(.\*? Served block blk\_.+ to .+)

E4,(.\*? Got exception while serving blk\_.+ to .+)

E5,(Receiving block blk\_?\d+ src: .\*?\sdest:.)

E6,(Received block blk\_?\d+ src: .\*?s\s of size.+)

E7,(writeBlock\sblk\_?\d+\sreceived\s exception\s.\*)

E8,(PacketResponder\s\d+\sfor block blk\_?\d+\sInterrupted.)

E9,(Received\sblock\sblk\_?\d+\s of\s size\s\d+\sfrom\s.+)

E10,(PacketResponder\sblk\_?\d+\s\d+\sException.+)

E11,(PacketResponder\s\d+\sfor\sblock\sblk\_?\d+\sterminating)

E12,(.\*?: Exception\s writing\sblock\sblk\_?\d+\sto\s mirror\s.\*)

E13,(Receiving\s empty\s packet\sfor\sblock\sblk\_?\d+)

E14,(Exception\s in\s receiveBlock\sfor\sblock\sblk\_?\d+.)

E15,(Changing\sblock\sfile\s offset\s of\sblock\sblk\_?\d+\sfrom\s\d+\sto\s\d+\s meta\sfile\s offset\s to\s\d+)

E16,(.\*?: Transmitted\sblock\sblk\_?\d+\sto\s.+)

E17,(.\*?: Failed\s to\s transfer\sblk\_?\d+\sto\s.+)

E18,(.\*? Starting\s thread\s to\s transfer\sblock\sblk\_?\d+\sto\s.+)

E19,(Reopen\sBlock\sblk\_?\d+)

E20,(Unexpected\s error\s trying\s to\s delete\sblock\sblk\_?\d+.\*?\sBlockInfo\s not\s found\s in\s volumeMap.\*)

E21,(Deleting\sblock\sblk\_?\d+.\*?\sfile\s.+)

E22,(BLOCK\\*\sNameSystem.allocateBlock:.\* blk\_?\d+)

E23 , (BLOCK\\*\sNameSystem . delete : .\* blk\_ . ? \d+ \sis \sadded  
\ssto \sinvalidSet \sof \s . + )

E24 , (BLOCK\\*\sRemoving \sblock \sblk\_ . ? \d+ \sfrom \sneededReplications  
\sas \sit \sdoes \snot \sbelong \sto \sany \sfile . )

E25 , (BLOCK\\*\sNameSystem . addStoredBlock : \sblockMap \supdated :  
\s . \* ? \ sis \sadded \sto \sblk\_ . ? \d+ \ssize \s \d+ )

E26 , (BLOCK\\*\sNameSystem . addStoredBlock : \sRedundant \saddStoredBlock  
\srequest \sreceived \sfor \sblk\_ . ? \d+ \son \s . \* ? size \s \d+ )

E27 , (BLOCK\\*\sNameSystem . addStoredBlock : \saddStoredBlock  
\srequest \sreceived \sfor \sblk\_ . ? \d+ \son \s . \* ? size \s \d+  
\sBut \sit \sdoes \snot \sbelong \sto \sany \sfile . )

E28 , ( PendingReplicationMonitor \stimed \sout \sblock \sblk\_ . ? \d+ )

E29 , (BLOCK\\*\sask \s . \* ? \sto \sreplicate \sblk\_ . ? \d+ \sto \sdatanode . + )

E30 , (BLOCK\\*\sask \s . \* ? \sto \sdelete \s+blk\_ . ? \d+ )

## APPENDIX C: CREATION OF STRUCTURED DATA

```
df_template=pd.read_csv('./HDFS_templates.csv')
chk_size=1000
count=0
first_time = dt.datetime.now()

for gm_chunk in pd.read_csv("./HDFS_parsed.csv", chunksize=chk_size):
    start_time = dt.datetime.now()
    count+=1
    for row_df_indx, row_df in gm_chunk.iterrows():
        #print(gm_chunk.loc[row_df_indx, 'EventId'])
        for row_df_temp_index, row_df_temp in df_template.iterrows():
            if re.match(row_df_temp[1], row_df.Content):
                #print(row_df.Content, row_df_temp[0])
                gm_chunk.loc[row_df_indx, 'EventId']=row_df_temp[0]
                gm_chunk.loc[row_df_indx, 'EventTemplate']
                =row_df_temp[1]
                break

gm_chunk.to_csv("./HDFS_structured.csv", index=False,
header=False, mode='a')
#print(gm_chunk.head(2))
print("{}_chunk_stop_{:}:".format(
count, end_time=dt.datetime.now() - start_time))

print("{}_last_time_{}".format
```

```
(last_time=dt.datetime.now() - first_time))
```

Output:

Date	Time	Pid	Level	Component	Content	Block	Label	EventId
0	81109	203518	143	INFO	dfs.DataNode\$DataXceiver: blk_-1608999687919862906 Receiving block src: ...	blk_-1608999687919862906	NaN	E5 (Receiving block blk_?d+
1	81109	203518	35	INFO	dfs.FSNamesystem: NameSystem.allocateBlock: /mnt/hadoop/m...	blk_-1608999687919862906	NaN	E22 (BLOCK\^sNameSystem.alloct
2	81109	203519	143	INFO	dfs.DataNode\$DataXceiver: blk_-1608999687919862906 Receiving block src: ...	blk_-1608999687919862906	NaN	E5 (Receiving block blk_?d+
3	81109	203519	145	INFO	dfs.DataNode\$DataXceiver: blk_-1608999687919862906 Receiving block src: ...	blk_-1608999687919862906	NaN	E5 (Receiving block blk_?d+
4	81109	203519	145	INFO	dfs.DataNode\$PacketResponder: PacketResponder 1 for block blk_-1608999687919...	blk_-1608999687919862906	NaN	E11 (PacketResponder's'd+lsfc

## REFERENCES

1. Patrick Kostjens *Anomaly Detection in Application Log Data*, Utrecht university, Master Thesis, 2016.
2. Aarish Grover, *Anomaly Detection for Application Log Data*, San Jose State University, Master Thesis, 2018.
3. Ahmed M., *Collective anomaly detection techniques for network traffic analysis*. Annals of Data Science, 2018.
4. Varun Chandola, Arindam Banerjee and Vipin Kumar, *Anomaly detection: A survey*, ACM Computing Surveys (CSUR), 2009.
5. Raghavendra Chalapathy, Sanjay Chawla, *Deep Learning For Anomaly Detection: A Survey*, 2019.
6. Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar *DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning*, ACM Conference on Computer and Communications Security (CCS), 2017.
7. Hayes, Michael A., and Miriam AM Capretz, *Contextual anomaly detection framework for big sensor data*, Journal of Big Data, 2015.
8. Sherenaz Al-Haj Baddar, Alessio Merlo, and Mauro Migliardi, *Anomaly Detection in Computer Networks: A State-of-the-Art Review*, 2014.
9. Oded Maimon, Lior Rokach, *Introduction to Supervised Methods, Chapter 8*, Data Mining and Knowledge Discovery Handbook, 2005.
10. S. B. Kotsiantis, *Supervised Machine Learning: A Review of Classification Techniques*, 2007
11. Erik G. Learned, *Introduction to Supervised Learning*, 2014
12. Xiaojin Zhu, *Semi-Supervised Learning Literature Survey*, 2006
13. Markus Goldstein, Seiichi Uchida, *A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data*, 2016
14. Risto Vaarandi, *A Data Clustering Algorithm for Mining Patterns From Event Logs*, Proceedings of the 2003 IEEE Workshop on IP Operations and Management, 2003.
15. D. Borthakur, *The hadoop distributed file system: Architecture and design*,



- Hadoop Project Website, 2007.
16. Pinjia He, Jieming Zhu, Shilin He, Jian Li, Michael R. Lyu. *Towards Automated Log Parsing for Large-Scale Log Data Analysis*, IEEE Transactions on Dependable and Secure Computing (TDSC), 2018.
  17. Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu, *Experience Report: System Log Analysis for Anomaly Detection*, IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), 2016.
  18. Dataset, <https://github.com/logpai/loghub>
  19. Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu, *An Evaluation Study on Log Parsing and Its Use in Log Mining*, 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2016.
  20. Pinjia He, Jieming Zhu, Zibin Zheng, Michael R. Lyu *Drain: An Online Log Parsing Approach with Fixed Depth Tree*, IEEE International Conference on Web Services (ICWS), 2017.
  21. Accuracy, Precision, Recall or F1, <https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9>
  22. <https://www.intel.ai/precision-and-recall-for-time-series/#gs.113ry0>
  23. loglizer (Python Library), <https://github.com/logpai/loglizer>
  24. Dan Gunter, Brian L. Tierney, Aaron Brown, Martin Swany, John Bresnahan, Jennifer M. Schopf *Log summarization and anomaly detection for troubleshooting distributed systems*, 8th IEEE/ACM International Conference on Grid Computing, 2007.
  25. Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, Xuewei Chen, *Log Clustering Based Problem Identification for Online Service Systems*, International Conference on Software Engineering (ICSE), 2016.
  26. Fei Tony Liu, Kai Ming Ting, Zhi-Hua Zhou, *Isolation Forest*, ICDM '08. Eighth IEEE International Conference on Data Mining, 2008.
  27. Bouchra Lamrini, Augustin Gjini, Simon Daudin, François Armando, Pascal Pratmarty, Louise Travé-Massuyès, *Anomaly Detection Using Similarity-based One-Class SVM for Network Traffic Characterization*, 29th International Workshop on Principles of Diagnosis, At Warsaw, Poland, 2018

28. Matthias Reif, Markus Goldstein, Armin Stahl, Thomas M. Breuel, *Anomaly Detection by Combining Decision Trees and Parametric Densities*, 19th International Conference on Pattern Recognition, 2018

## Contact

+90 532 6032703 (Work)  
uygarsahin@gmail.com

www.linkedin.com/in/uygarsahin  
LinkedIn)

## Top Skills

AIX  
Storage Area Networks  
ITIL

## Languages

Turkish (Native or Bilingual)  
English (Professional Working)

## Certifications

Python Programing from A to Z  
Oracle Linux: System Administration  
Ed 3  
IBM Certified System Administrator -  
AIX 7

# Uygar Sahin

Senior System Administrator at KKB Kredi Kayıt Bürosu  
Barbaros Mah., Istanbul Province, Turkey

## Summary

### Specialties:

- IBM Power Systems
- System virtualization configuration and administration
- Vmware,
- IBM AIX administration and Upgrading
- Oracle Linux System Administration
- IBM Spectrum Scale ( GPFS ) implementation
- Storage administration
- IBM Storwize Family
- IBM DS8K
- EMC VMAX, Unity, isilon
- Hitachi VSP G series
- Software Define Storages
- Storage Virtualization configuration and administration
- SAN Design and configuration
- Brocade Gen5, Gen6 FCIP router
- Cisco, MDS series
- Disaster Recovery Design and configuration
- Python/Korn Shell/Perl Scripting
- ITIL v3 Foundation of Service Management Certification

---

## Experience

KKB Kredi Kayıt Bürosu  
Senior System Administrator  
December 2017 - Present  
Istanbul, Turkey

Storage-SAN-Backup Systems Administrator

Akbank  
Senior Storage Administrator  
December 2016 - December 2017 (1 year 1 month)  
Gebze

- Between Prod site and DR site line monitoring script (Perl script)

- IBM SVC remote copy consistency group state daily monitoring script (with Python)

## IBM

### STG Lab Services Consultant

June 2015 - December 2016 (1 year 7 months)

Istanbul, Turkey

- Spectrum Control implementation and Configuration
- Spectrum Scale implementation ( GPFS )
- Spectrum Virtualize configuration
- Spectrum Virtualize IP replication implementation
- DS8K configuration

## Akbank

### Storage Specialist

January 2013 - June 2015 (2 years 6 months)

Istanbul, Turkey

#### UNIX - Storage Administrator in AKBANK

- Daily disk operation for AIX, LINUX, Windows servers ( create file system, add disk etc.. )
- Storage Systems performance monitoring and reporting with TPC and Stor2rrd.
- Performing migration of AKBANK storage environment to a fully virtualized infrastructure with IBM SVC
- Monitoring AKBANK SAN infrastructure with BNA( Brocade Network Advisor)
- Management of AKBANK SAN infrastructure.( Brocade DCX 8510 )
- AKBANK storage and SAN infrastructure maintenance
- VIO, dedicated AIX server hba card maintenance.
- Create Replication architecture and testing. ( IBM SVC replication and TPC-R)
- Management of Storage System like EMC VNX 5700/7500, EMC VMAX 40K, EMC VMAX 20K, HDS USP/ USP-V, IBM DS8K
- EMC VPLEX ( Metro Cluster ) implementation to AKBANK infrastructure.

## ISNET

2 years 8 months

### System Engineer

May 2012 - January 2013 (9 months)

Unix - Storage Administrator in ISBANK

## System Specialist

June 2010 - May 2012 (2 years)

Unix - Storage Administrator in ISBANK

## Kadir Has University

Graduate Assistant

October 2008 - June 2010 (1 year 9 months)

Courses Taught:

Signals & Systems

Circuits Lab.-I

Electronic Circuits Lab. I

Digital Communications

Communication Systems

Microprocessors

## Profilo Telra Elektronik San. Ve Ticaret A.S.

Intern

August 2006 - September 2006 (2 months)

One month training at R&D Department. Intervening immediately to eliminate of the product defect and failure during testing and production and working for the development of existing products.

## Ford Otosan A.S.

Intern

July 2002 - August 2002 (2 months)

One month training at Body Department. Developing robot programming and making the installation of the program, programming of welding robot which is in the production line and introduce a solution according to existing failures.

## Turk Telekom

Intern

July 1997 - August 1997 (2 months)

One month training at Cable Tv Department. Resolving of a cable tv system failure and updating the system after required processes.

---

## Education

Kadir Has Üniversitesi

3.57, Electronic Engineering · (2008)

Kocaeli Üniversitesi

3.08, Industrial Electronic · (2001 - 2003)

Macka Akif Tuncel Technical High School

4.16, Electronic · (1997 - 2001)