POLITECNICO DI TORINO

Master's Degree Course In
Ingegneria Informatica (Computer Engineering)

Master's Degree Thesis

Anomaly Detection In The Enterprise Context Through Log Analysis

Academic Supervisor
Prof. Paolo Garza

Candidate
Dr. Alessandro Bozzella

Mediamente Consulting Srl
Business Unit
Technological Infrastructure
Corporate Supervisor
Dr. Vincenzo Scinicariello

Academic Year 2019-2020 December 2020

† To my maternal grandmother

Acknowledgements

I would like to thank, first of all, who accompanied me during the curricular internship and the thesis in the company process (despite the obvious difficulties caused by the dramatic historical period in which we are living), allowing the realization of the latter because without their valuable technical suggestions useful for dealing with the problems that have arisen I would not have written these pages.

Hence my thoughts go to all the components of Mediamente Consulting Srl that first virtually and then in presence have welcomed me with extraordinary availability and patience, making me feel comfortable during my first working experience in this area.

A special thanks goes then to the professor Paolo Garza, that guided me with his outstanding preparation and humaneness during all the academic path not only with the curricular internship and the thesis but also with his course of Big Data: Architectures And Data Analytics followed during my master's studies.

Last but not least and with no little emotion I have to say my biggest thank you to my family and relatives (the list here is very long, starting from my maternal great-aunt, uncles, paternal grandmother and so on) to whom I dedicate this work, aware of the fact that without their sacrifices I would not have been able to be where now I am.

Finally (the best for last!) I want to mention and thank my girlfriend and adventure mate Alina (including her mother and her stepsister) with whom I shared from the beginning the desire to attend the master's degree course at the Politecnico di Torino and that gave me the firmness and the tenacity not to surrender so that the dream came true.

Contents

List Of Tables

List Of Figures

Part I

Preamble, Motivations And Goals

Chapter 1

Introduction

This introductory chapter briefly gives a panning for what comes next, introducing the required knowledge domain and the corporate with whom the investigation has been conducted.

1.1 Background

With the advancement of technology, computer systems have undergone a progressive evolution: indeed they have become systems spread around the world composed of a large number of calculators or a relatively small amount of machines with very high performances. This new characterization of IT systems is becoming the favorite choice for the IT industry as it provides the basic technological structure for a large variety of Internet services; furthermore usually these IT systems are required to operate throughout the year. In this sense, the need to provide a functioning service at any time will become crucial and any type of malfunction can result in a system failure and a consequent considerable loss of information.

Nowadays the usage of technologically evolved systems is a must for the development of the services offered by a company: indeed the handling of the technological infrastructures that supports these systems reached a very high level of complexity such that there has been an enormous increase in the job hiring in the IT department. The professional role covered by these professional figures requires the capability of managing a variegated IT ecosystem that ranges from hardware to the application level and at the same time demands also the efficient and fast resolution of the encountered problems. Hence in the environment mentioned above it is necessary to optimize and automate as much as possible the maintenance operations of IT infrastructures

to allow insiders to focus only on the more complex aspects of corporate IT management. For this reason, over time the so-called "engineered systems" have aroused particular interest: installations in other words that integrate hardware and software in favour of a greater efficiency and simplicity of the final product.

1.2 Overview

The detection of anomalies can be considered a valuable instrument of an advanced technological system, whose purpose is to promptly discover any anomalies and thus playing a central role in the management of faults for this type of system. A rapid detection of an anomaly allows technicians to solve the problem as soon as possible, in such a way as to reduce the time in which a certain service is unusable. All computer systems nowadays generate a large amount of logs where they enter information deemed important during their operation: such log files can be used as a data source for system-wide anomaly discovery; for this reason the detection of anomalies through the system logs is very recurrent both in the academic and business field. The log files of individual computers are often analyzed line by line in order to disclose anomalies thanks to exploitation of key terms or regular expressions; unfortunately with the advent of the new systems described above this procedure is no longer feasible due to the increase in the architectural complexity of the latter, the high number of logs that are produced and the different malfunctions toleration mechanisms implemented by each of them.

Starting from the general considerations made so far, this work focuses in particular on the application of appropriate techniques of machine learning (with whom has a strong relationship and which is a branch of the artificial intelligence that deals with improving, through a set of methodologies, the understanding and thence the learning by machines). Machine learning with a wide range of algorithms is able to identify recurrent patterns, classifying data and learn by themselves to absolve specific assignments: in recent years it had a considerable use but nevertheless for many people and also for many experts remains an unexplored topic.

In particular the conducted study aims to detect anomalies through the application of a real case study: the analysis of a specific log file called **customer's_name-logs-rdbms**, produced by an Oracle database software deployed on a cluster provided to a final customer of the company. Furthermore, the study was tackled keeping in mind also the intention of laying

the foundations for the future realization of an instrument capable of, aside from being able to understand the state (normal or abnormal) of the system, reacting autonomously to "abnormal events".

In this sense, the research work is carried out within the business unit of Technological Infrastructure of the Mediamente Consulting Srl company, is born as an integration of the previous work on the "intelligent monitoring" already started in the company and also in the field of university research in cooperation with the Politecnico di Torino thanks to a previous master's thesis project. If the aforementioned project analyzed the possibility of predicting possible system faults through the application of predictive models based on numerical data from the Enterprise Manager monitoring software, with the study in question instead the focal point is on the possibility of knowing with the available data whether the status of the system has anomalies or not through the detection of anomalies thanks to the analysis of system logs. Specifically the activity conducted by the business unit of Technological Infrastructure relates in particular to the sphere of Oracle technologies and products, the company's area of specialization. For these reasons and for the fact that there was the need of identifying the most used system log in the area of troubleshooting the largest part of customer problems, the log file of a relational database management system has been decided to consider: this is represented by a customer's Oracle Database instance. Furthermore the reason for limiting the analysis to a real and precise context is to be found in the interest of evaluating the validity of the analysis considering realistic values. In reference to this, the framework is made up of several parts: log collection, log parsing, feature extraction and anomaly detection. The first phase is essential to fetch the material to be analyzed, while the second aims to process the log file in such a way as to put it in a structured and well-defined form in order to facilitate the work of the subsequent phases. After these first two phases, there is the feature extraction, which consists in extracting numerical feature vectors from textual fields of the log file. The fourth and last phase is substantially the most important of all as it constitutes the central part of this work: as its name suggests, it consists in identifying any anomalies present in the analyzed log file. Therefore, the numerical data obtained in the previous phase serve as a starting point for applying thereafter some machine learning algorithms: here the choice fall on using algorithms belonging to both supervised and unsupervised learning in order to see which of the two types of learning was the most suitable to carry out the assigned task. For this purpose, three widely used performance metrics were used to evaluate the accuracy of the anomaly detection methods

both supervised and unsupervised, varying also a series of parameters and the validation strategy adopted (for what concerns the supervised analysis). The objective of the analysis is, in summary, to evaluate the effectiveness of the models examined, determining whether they are able to predict with a reasonable level of accuracy the current state of the system under examination (normal or abnormal) in terms of performance metrics and to provide references for future developments.

1.3 The Enterprise Context

The research project arises from the encounter between the intention to investigate the possibilities of machine learning and the proposal of the company Mediamente Consulting Srl. Mediamente Consulting Srl is a consulting company operating in the IT sector with the declared intent of directing the strategies of companies through a path of technological development and innovation, using the concepts and tools of business analytics and the knowledge of a team of proven experience. Among the keystones on which the company's business is based is therefore the continuous research towards the innovation. In this regard, her recent past in the Incubator of Innovative Enterprises of the Politecnico di Torino is noteworthy, being awarded the "Startup Of The Year" in 2016 for growth merits. Over the years, the company has assisted in the formation and development of multiple areas of specialization: Technological Infrastructure, Data Integration and Management, Corporate Performance Management, Advanced Analytics, Business Intelligence. The work was carried out at the same time in conjunction with the IT Management activity conducted by the Technological Infrastructure unit, whose objective is to provide design, management and monitoring services for the technological and application infrastructure.

1.3.1 The Business Unit: Technological Infrastructure

In order to understand more clearly the context and the reasons for the work exhibited, a more precise description is presented about the role of the business unit within the company in which the master's thesis was made. The activity carried out by the Technological Infrastructure team essentially consists in providing consultancy regarding the handling of elaborated IT systems. Specifically, the services provided span multiple operational fields: from the deployment of the infrastructure to the performance analysis, up

to the maintenance and periodic updating of software systems. One of the strengths of the proposed service is undoubtedly to be identified in the mastery of the products and technologies of the Oracle family, on which the company has invested heavily in terms of knowledge, to the point of earning an excellent reputation with its customers.

The master's thesis work is closely related to the Application Management Services (also known by the acronym AMS) carried out by the company. Generally, the AMS service can be described as a well-defined process characterized by the use of related to each other tools to pinpoint, solve and document how good is the service provided by a structured business transaction to guarantee that they meet or exceed the performance measures of end users in relation to how fast a transaction is completed or how the information is delivered to the end user [1].

1.4 Document Structure

The master's thesis work is organized into three parts: the first part does not require any thorough technical competence, wanting to be a gentle exposition of the sphere of knowledge in which the study can be placed in association with the description of the collaborating company with which the research was done; in second place there is the delineation of the problem with the related parameters and formulation of a case study and the purposes for which the work itself aims.

The second part gives the analytical means (beyond which are mandatory different notions in programming and machine learning not provided here) needed to fully understand each single piece of the "value chain" shown here and then focuses on the chosen programming language as well as the employed software libraries.

The third and last part represents the main one, making initially a theoretical recall of the machine learning models employed for the anomaly detection and then finally apply for each of them the tests through which are produced the numerical values representing different metrics used as a means of comparison among the utilized models and as an indication of the results of the single model with the related considerations. At last general conclusions are drawn, highlighting the weaknesses and strengths of the entire process, ending later with some reflections on future developments.

Chapter 2

Problem Definition

The second chapter was intended to illustrate the problem faced along with the conceptual scheme of the workflow followed for addressing the problem itself and to obtain the proposed solution in the subsequent chapter; in the end there is the definition of the parameters of the problem with the correlated real case study considered.

2.1 Description Of The Problem

The monitoring of IT systems, intended as a synthesis of support and problem solving activities, is one of the most valuable points of the company business. The need to provide a quality service, thus strengthening its own position in the sector, requires the use of adequate resources and, considering the volume of customers and the heterogeneity of the problems to face, it can be understood how this can represent a critical aspect. From here the need to seek solutions to improve, in the first place, the efficiency of the service proposed and, consequently, its quality.

To make an improvement to the service, it is advisable to understand, first of all, what is meant by problem and what are the way to operate and the tools available to deal with it.

The term problem, referring to a computer system, means a series of causes that contribute to determining a malfunction condition of the system, corresponding to a deviation, in negative, of its parameters from normal values, up to causing a more or less serious degradation of performances. In most of the cases, the occurrence of a problem is reported by the user using the system or detected automatically by a monitoring software (for example, by sending a report following the exceeding of a certain threshold). However,

often it is difficult to identify the causes of the problem, allowing only an intervention a posterior, that is, once the problem has occurred and/or has been perceived by the user.

The research work focused precisely on this aspect: investigating the possibility of extending the actual domain knowledge of the company (represented by the previous master's thesis works as stated in the first chapter) thanks to the anomaly detection through the log analysis for building in future a support tool for monitoring and troubleshooting, capable to allow a timely assessment of the problems and therefore permit a proactive intervention.

Figure 2.1 shows the conceptual scheme of the workflow followed for addressing the problem. This scheme aims to highlight the passage from a general type of problem to the proposal and therefore the design of a solution and how this was then applied to a very specific context, defined according own parameters. The solution is defined as a set of choices, both of a technical nature (e.g., software products and libraries) and related to the analytical phase (e.g., the type of analysis, the adopted algorithms, etc.), to set up an architecture capable of achieving the intended objective.

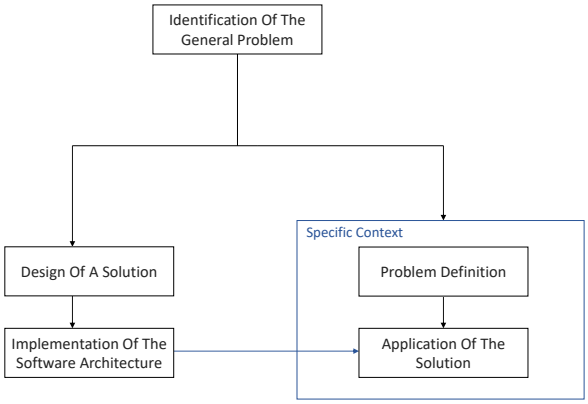

Figure 2.1: Conceptual Workflow Diagram

2.2 Problem Parameters Definition

The definition of the state of a system that consents the characterization of the parameters of a problem obviously cannot ignore the fact that every system is different from the other: indeed each system is observed and

evaluated according to its characteristic parameters, a concept that can be translated into the analysis of logs files or any other type of information generated by the system itself or by components associated with it. In other words, the presence of the data is mandatory, numeric and not, which allow to understand the state in all its aspects.

Recent computer systems often exploits the interaction among various components, both hardware and software, which leads to a greater complexity in the analysis of such systems.

2.2.1 Case Study

The case study presented concerns only a single system and, in the specific, the instance of a relational database (relational database management system, RDBMS) of a customer. The choice to consider a real case was substantially dictated by the need of evaluating the actual validity of the proposed solution. A special environment would undoubtedly have guaranteed greater control, having the ability to carry out targeted tests. However, in this case the results obtained would not be of any guarantee with respect to have a real context as a "scenario".

Specifically, the case under study refers to the discovery of abnormal behaviours in the log file relating to a customer's Oracle Database instance (for privacy reasons the name of the customer is omitted). What the investigation wants to emphasize is the possible presence of errors related to routine operations executed on a RDBMS: remaining within the limits of the shareable information, this can be accomplished thanks to the checking of the value of a specific attribute present in the log file in question that signals if an error is happened or not respectively through an error code or the NULL value. Every error (that appears in this log file) has a standard format characterized by a fixed part common to all and a variable one: the first portion is **ORA-** while the second reports a number that can vary over an extended range of possible numbers, pointing out that there is a complex diversification of the errors [2]. For instance, the error codes that ranges in a certain interval can identify only a certain class of errors, thus allowing a simple diagnosis of the type of problem and a faster solution of them; as a concrete example a very recurrent type of error is **ORA-00028**, asserting that the session of a certain user for a specific database has been dispatched because another user with higher privileges has discarded it. Another also frequent is **ORA-00030**, reporting that this user session ID does not exist anymore due to a probable disconnection from that session.

As regards the type of information to be analyzed, it should be noted that these are not numerical values: as a matter of fact they are three alphanumerical attributes, describing the date of occurrence of the event (**event_date**), the (eventual) error code (**ora_errors**) and the corresponding message for that record (**message**) detailing the event itself (the entire content of the log file is not reported in this document for the reasons of privacy just mentioned above). These attributes are not the only ones to be present in the log file but are the most significant for the analysis to be conducted: not by chance the feature selection was carried out since fields such as **host**, **path** and so on represent personal characteristics related to the machine but in this scenario the data source comes from a single customer machine and hence have been discarded being useless. A further point to be clarified concerns the possible values assumed by the attributes of interest: these can contain not only information regarding the originated error but also the description of a successful performed transaction (**event_date** will have the same meaning, **ora_errors** will have the NULL value as stated before and **message** will outline the details of the performed action).

For the sake of completeness in the following there is a brief description of what a log file is together with the type of information that usually stores and of the architecture on which the instance in question is hosted and the Oracle Database Appliance (in short ODA) along with further notes on the release of the instance itself. Finally there will be a formalization in greater detail of the faced problem, depicting what the expected input and output should be.

Log File

A log file (having a .log extension) is a file that contains a list of events which have been "logged" by a computer. Better said, it is a computer generated data file that contains information about usage patterns, activities and operations within an operating system, application, server or another device that decided to record this type of information.

Oracle Database Appliance

The Oracle Database Appliance is an Oracle engineered system that aims to offer a simple, optimized and cost-effective solution for databases and applications Oracle. The ODA is distributed as an integrated system of software features, computing, networking and storage. In summary, the ODA is composed by one or more Oracle Linux servers connected to each other

and by a storage unit, allowing the execution of a single instance database or in cluster mode. The technical specifications, of course, vary depending on the model considered and the type of need for which it is used.

Oracle Database

The target analyzed is the instance of a relational database, in this case an Oracle Database 12c Enterprise Edition 12.1.0.2.0 - 64bit Production [3]. The system version is by no means a secondary aspect: the choice of using one version rather than another implies the possibility to use, or not, certain features [4]. In addition, Oracle also allows the integration of other features, regardless of the type of edition. Depending on the services and products used, there is a correlated access only to certain information that may also differs in terms of quality. Therefore the possibility of being able to use or not certain functionalities it undoubtedly influences the management operations of a database that could be more or less complex.

Expected Input

Figure 2.2 is a simple representation that summarizes the attributes of interest present in the log file to be analyzed in order to provide a clearer view of what is the input data source available.

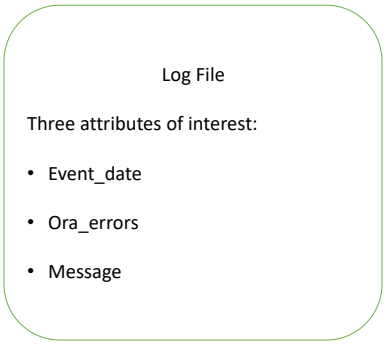

Figure 2.2: Expected Input

Expected Output

Figure 2.3 illustrates the expected output in conjunction with three performance metrics used for the evaluation of the obtained results (for more details see the paragraph 6.1 of chapter 6) in a similar way to what has been done for the expected input: this outcome is generated after the execution of the proposed framework in this document.

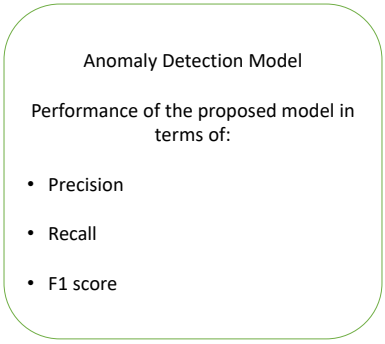

Figure 2.3: Expected Output

Part II

Building Blocks, Tools And Software

Chapter 3

Proposed Solution

The present chapter outlines the "value chain" followed, describing it not only from a theoretical point of view each single piece but also the conducted calibration of some specific parameters of a portion of exploited instruments, allowing to lay the foundations for chapter 5 and chapter 6.

3.1 Framework Overview

The analysis of a log file aimed at detecting potential anomalies is a process that consists of four principal steps: log collection, log parsing, feature extraction and anomaly detection (see [5] for a complete reference). Figure 3.1 graphically depicts the various phases. In this diagram is highlighted the flow of data through the various blocks. The components indicated for each block refers to the context under consideration. However, the possibility of adapting the proposed architecture to different contexts cannot be excluded: this will involve obviously another kind of source and/or analysis. In the following paragraphs there will be an elaborated exposition of each phase with the related instruments used, trying at the same time to clarify the reasons that led to these choices.

3.2 Log Collection

Before proceeding with the fetching of the information constituting the data source, the system log with the highest percentage of usage in the field of troubleshooting the most frequent problems has been chosen among the available ones with a subsequent feature selection.

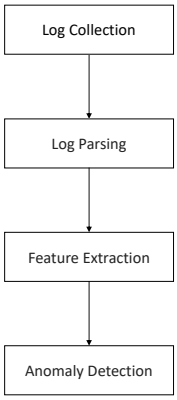

Figure 3.1: Framework Of Anomaly Detection

At the moment almost every system record information about its current state and other utilities information during its execution (CPU load, number of threads in usage in case of a multi-threaded system, etc.). The entry point of the work starts here: because the relevant data (the log file) were present on Elasticsearch (belonging to the ELK suite explained in a while), what has been done is the creation of a Python script (details on it, including the versions used, will be given in the next paragraphs) in order to create a client connection with one of the two network nodes on which Elasticsearch resides (it is worth to notice that this has been possible connecting to a Virtual Private Network for accessing the LAN of a specific customer of the company). After that a query in charge of requesting the data has been performed, the data selection has been conducted thanks to the employment of a regular expression aimed at discarding all the records that contain only symbols in the **message** attribute of the log file (the reason behind this choice is straightforward: records containing only symbols in the **message** attribute were not useful for the type of conducted analysis, representing only outliers that negatively affect the quality of the analysis itself); lastly the data formatting has been performed: the downloaded data has been organised in order to have a better visualization of them (the tab character has been used, permitting a fixed spacing among the field of the several attributes).

In summary the salient parts are:

1. Creation of a client connection.

2. Execution of the query.
3. Data selection.
4. Data formatting.

3.2.1 Elastic Stack

Elastic Stack is an open-source suite of products designed for research, analysis and display of data in real time, without placing restrictions on the origin and format of the same data. Elastic Stack provides a set of software components, each of which performs one precise function and may or may not interact with others. Initially, the product was known as ELK Stack, from the initials of the three fundamental components of which it is made: Elasticsearch, Logstash and Kibana. Over time, other modules have also been integrated (e.g., Beat), which have helped to simplify the use of the product and enrich its functionalities. Although the stack consists of a number of programs and functionality, attention will be paid exclusively to the components of interest to the objectives of this discussion.

Figure 3.2 shows the basic elements belonging to Elastic Stack, also providing an indication of the hierarchical level they cover within the framework. The parts referred to are the following:

- Beats.
- Logstash.
- Elasticsearch.
- Kibana.

A brief description of the functions performed is provided for each component in general and limited to the specific case.

Beats

The Beats are agents whose main function is to send generic information, whether related to a system or to a software application, therefore without particular format constraints, towards Logstash or Elasticsearch.

Although they were not actually used in the course of this work, however they are mentioned as they represent an excellent tool for collection of heterogeneous data (e.g., system logs), which, in view of a future development, could be integrated into predictive analysis.

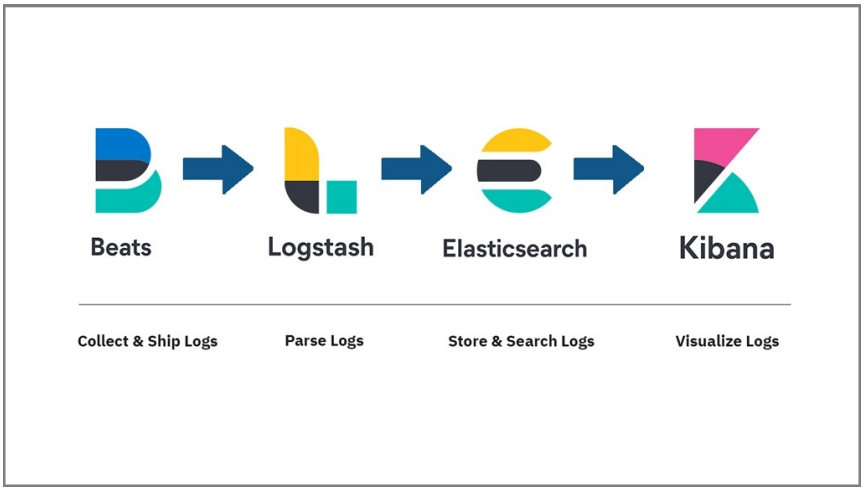

Figure 3.2: Main Components Of Elastic Stack

Logstash

Logstash is the component that collects data from one or more sources, eventually apply filters on them and send them in output to a generic stash (i.e., a repository), which is usually represented by Elasticsearch. Logstash carries out its function by defining a pipeline consisting of the following stages:

1. Inputs.
2. Filters.
3. Outputs.

The input module takes data from a source that can be of various kinds (e.g., file system, Beat, database, TCP socket etc.), thanks to the support of numerous plugins, that allow the management of the events generated by the sources themselves.

Elasticsearch

Elasticsearch represents, in a certain sense, the heart of Elastic Stack, that is the research and analysis tool, as well as the collector of the data extracted from the various sources. In essence, it is a search engine based on the Lucene library. Through HTTP web interfaces and the use of JSON for document formatting, allows a simple interaction with the user or other software and

the possibility of performing full-text searches effectively. Furthermore, Elasticsearch operates in an environment distributed, an aspect that guarantees scalability and resilience characteristics, in a completely transparent way to the applications with which it interacts.

For the purposes of this work, this is the only architectural component of the Elastic stack through which the visible communication (in terms of written lines of code) is happened; the established architecture consists of a single Elasticsearch cluster consisting of two nodes (i.e., two separate machines), each of which performs his own instance and is in communication with the other one. The information about the systems to be monitored is stored within the cluster in the form of JSON documents[1]. In the case study, this information corresponds as already expressed several times to the log file of a database instance and the access to its relative content is made thanks to the execution of an appropriate query (please refer to the Log Collection paragraph for more details about it), which return the result requested for the desired purposes (e.g., visualization, analysis on data).

Kibana

Kibana is the platform that allows you to consult and view the information of interest based on the filters that are set. The data can be represented according to different types of graph, based on the information that you want to analyze. It is also possible to create dashboards, that is, screens in which to collect the information deemed most important in order to facilitate consultation. For this reason, this functionality turns out to be remarkable utility for those who monitor systems.

From the perspective of this work and as future developments of the same, Kibana is therefore configured as the interface between the system and the user through which could be projected the trends of the most recurrent log events (defining appropriate alarm thresholds) in order to give a support to the technician in the identification of anomalous behaviors of the system under consideration.

[1]Elasticsearch is associated with the category of document-oriented databases. A document, in the terminology of Elasticsearch, is a JSON data structure, composed of key-value pairs, and can be seen as the row of a table in a relational database.

Output Of Phase 1

The output obtained at the end of this intermediate step constitutes the dataset on which the whole analysis below is based.

An annotation to make until now is: due to the intrinsic nature of the available data and their relative homogeneity, the data transformation belonging to the category of data preprocessing was not necessary.

A second comment on which is important at the same level to dwell is the observation of the existence of an imbalance about the presence of what is a "normal event" and an "abnormal event" (as it should be most of the events that take place are successful): this is caused by the imbalance of the two possible values that the key attribute among the three available can assume (please refer to the subparagraph 2.2.1 of chapter 2 for a clarification) and the consequent structural choice in not conducting any balancing operation. The reason is very simple: this preference linked to the composition of the dataset wants to demonstrate the effort of this work in wanting to standardize it as much as possible to the real world and to offer a framework applicable to an extracted information as it is without any human intervention, but paying a price from the point of view of the accuracy of the resulting model compared to the situation of having a balanced dataset.

3.3 Log Parsing

Usually a log file doesn't contain any type of structure but it is characterized only by text that doesn't follow any semantic rule in order to ease the understanding. For this reason the purpose of this phase is to generate a set of generic events put in a structured way: this is realized converting the log file taken in input into another. In other words for each record the specific parts are removed because are not useful for the analysis, while the recurrent parts that can appear also in other records constitute the template of what can be called an event (further details are showed here [6] and here [7]). Figure 3.3 explains better the concept: the 4th log message (Log 4) is parsed (mapped) as "Event 4" with an event template "TMON started with pid = *, OS id = *". In literature according to the state of the art regarding the existing log parsing methods available, there are two types of log parsing methods: the ones based on grouping techniques and the ones based on techniques guided by the intuition. The first type relies on the concept of group or cluster (the concept comes from the world of machine learning and it is strictly correlated): indeed inside a log file are calculated initially the distances between

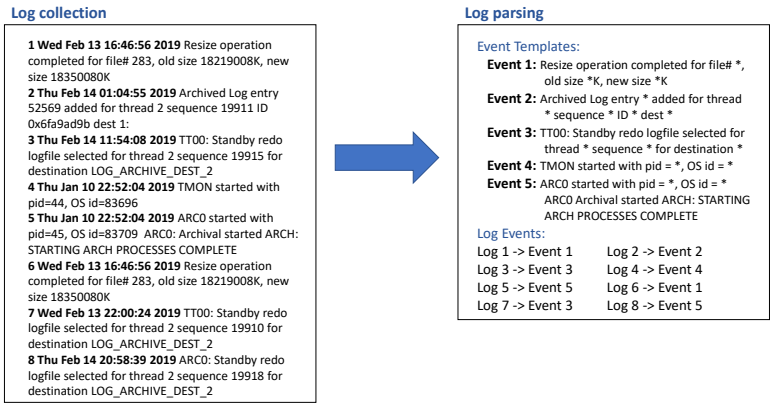

Figure 3.3: Log Parsing

the records and after that are applied clustering procedures so as to insert every record in the right cluster and originate a generic event representative of the latter. The second type instead is characterized by counting the occurrence of each word for every single position of the record: finally the most used words (the ones with the highest count) are taken into consideration: among these, some will be selected having the same role of those selected with the previous methodology.

The rationale behind the selection of the appropriate heuristic-based log parsing algorithm among the existing ones has been done in terms of benchmark results (evaluated across 16 different logs): indeed in literature all the log converters have been evaluated reporting the parsing accuracy as discriminating element, defined as the percentage of accurately parsed log messages; the choice fell on the IPLoM algorithm, representing the best compromise in terms of parsing accuracy and complexity.

3.3.1 IPLoM

Before going into the logical details of the algorithm some basic definitions are detailed that facilitate the understanding of what comes next.

Basic Definitions

- Event Register: contains the tracking of what happened on a machine (it is a synonym for a log file).

- Occurrence: identifies a specific record in the event register with its related information.
- Symbol: represents a single word inside the textual part of an occurrence surrounded by empty spaces.
- Occurrence Size: the number of each symbol appeared in the literal part of an occurrence.
- Occurrence Group: a subset of the event register characterized by having its records with the same textual part.
- Group Delineation: a generic line written in natural language that does not contain any type of parameter but asterisks, representative of all the elements in an occurrence group.
- Fixed Symbol: a symbol within the literal field of an occurrence that doesn't change expressed in natural language.
- Variable Symbol: it is the exact opposite of a fixed symbol and it is represented by an asterisk.

The IPLoM Algorithm

The IPLoM (Iterative Partitioning Log Mining) algorithm is a grouping algorithm: it is based on a loop that at each iteration divides in sub groups a set of logs. The final outcomes are the sub groups and the corresponding group delineation that best fits. Figure 3.4 depicts the entire steps. In addition to that there are two simplifications about the log file that can be passed in input to IPLoM:

1. An occurrence in the log has a textual description of itself in addition to the other fields.
2. The format of the characterization of the occurrence is not known in advance or is unclear.

The algorithm is subdivided in four principal phases, trying to determine all the existing group delineations in an event register and makes available as well a function able to remove those occurrence groups that do not reach a certain value chosen by the user.

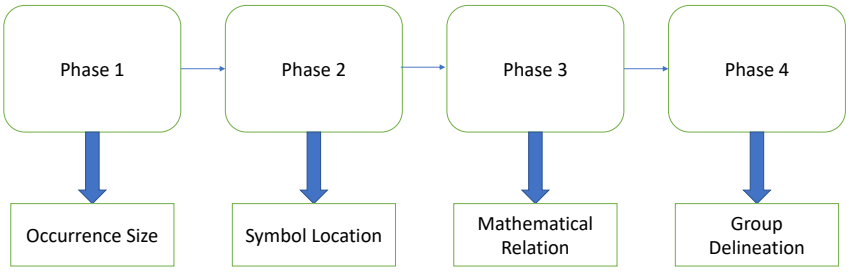

Figure 3.4: IPLoM Overview

Division By Occurrence Size

Occurrences belonging to the same group delineation are likely to have the same occurrence size: this is the initial splitting criterion. At the end there will hence be an high probability of obtaining groups with the same occurrence size due to the inherent property of many elements of these groups of having the same occurrence size. Below Figure 3.5 clarifies the concept.

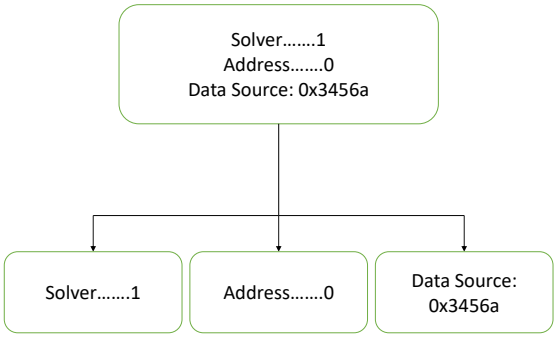

Figure 3.5: IPLoM Step-1: Division By Occurrence Size

Division By Symbol Location

The splitting methodology of this phase aims to find the symbol location in each occurrence group with the lowest number of different words because in this case in that position is highly probable that there will be the same words: so in this case the occurrence group is further divided using these recurrent values in that symbol location; surely each new partition will contain only one of those for that symbol position. Moreover the algorithm also here provides a specific threshold in order to guarantee a good performance: indeed every group that has a value lower than this will be discarded. Figure 3.6 can ease the comprehension.

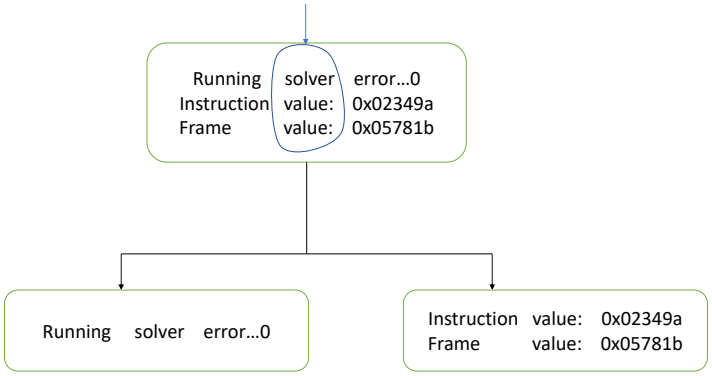

Figure 3.6: IPLoM Step-2: Division By Symbol Location

Division By Search For Mathematical Relation

The last partitioning step is designed to find any type of bijective relationship (that is at the same time injective and surjective) between two unique symbols coming from a group of unique symbols. Initially for each symbol location of a partition is calculated the amount of unique symbols; later the most recurrent symbol numbering among all the symbol locations will be selected, which has the constraint of having to be greater than 1. The choice of selecting the most recurrent symbol numbering is given to the fact that it probably represents the number of instances belonging to the group. Whether it is true what has been said, this will imply a bijective relationship between

the symbols in the symbol positions that have this symbol counting and occurrences having these symbol values in the corresponding symbol locations are further split. Figure 3.7 gives an idea of how it works the subdivision.

Unfortunately the bijective relationship may not even be a 1 to 1 relation: in fact often can be a 1 to N, N to 1 and N to N. An N to N relation is discarded because for obvious reasons is not possible to split a message of this type. For what concerns the 1 to N and N to 1 relations there are specific techniques able to handle both cases (refer to this paper [8] or the other one [9] for more technical details).

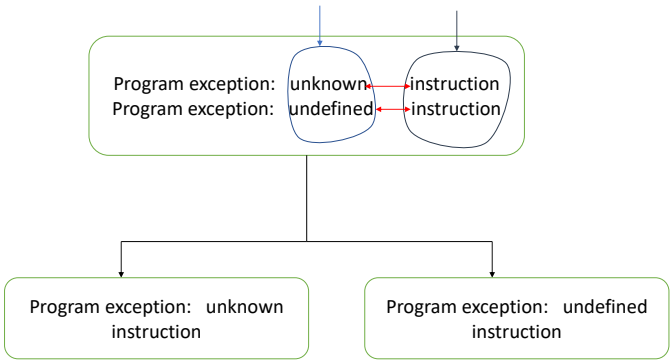

Figure 3.7: IPLoM Step-3: Division By Search For Mathematical Relation

Selecting Group Delineations In Every Occurrence Group

In this last section there is no more division to adopt: indeed the hypothesis is that every occurrence group is different from another and each one is able to uniquely identify its members.

Here the purpose is to discern between fixed and non recurrent values inside a group delineation (made up of a textual line were the non recurrent terms are substituted by asterisks and the recurrent ones are expressed in natural language) by counting the number of single symbols for every symbol location of an occurrence group: if a symbol position has more than one value then it is non recurrent, otherwise it will be considered recurrent in the group delineation. Figure 3.8 portrays the last phase.

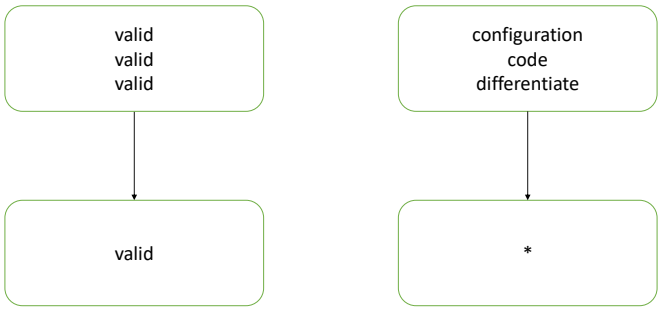

Figure 3.8: IPLoM Step-4: Selecting Group Delineations In Every Occurrence Group

Parameters Tuning

With reference to the present work, the sensitivity analysis performed to evaluate the stability of IPLoM in terms of a good structuring of the obtained result using different values for the parameters (a full explanation of these things is given in publications already cited above, like this one [8] or the other one [9]) showed that the best values for its parameters are respectively the following:

- **File Support Threshold:** 0.
- **Partition Support Threshold:** 0.
- **Upper Bound and Lower Bound:** 0.7 and 0.03.
- **Cluster Goodness Threshold:** 0.1.

Note that these values are the consequence of different executions of the software and not of a rigorous mathematical procedure that allows to predict a result.

Output Of Phase 2

The output obtained by the execution of this algorithm are two csv files: one (called **log file name.log_structured**) containing exactly the same

content of the input log file but in addition for each record there is an irrelevant **LineId** attribute, an associate **EventId** attribute (it is important to notice that one or more records can have the same identifier due to the fact that correspond to the same template) and the **EventTemplate** attribute representing the generic log event without the specific record parameters; the other one (called **log file name.log_templates** having the **EventId**, **EventTemplate** and **Occurrences** attributes) contains less records than the structured one with the list of all the group of event templates (including the occurrence for each of them) extracted by the input log file.

3.4 Feature Extraction

The main purpose of this phase is to codify in a numeric way the logs in order to apply on them some machine learning algorithms used for detect anomalies.

This is accomplished by using various techniques adopted for dividing the logs into various groups: for each of them there is an analogous vector that counts the number of times every record is present in that group. The "final product" issued by joining all these vectors is a matrix Y where the generic row $Y_{m,n}$ counts the number of times the record n is present in the m-th group.

To put in order what was said, the input of feature extraction are the logs coming from the previous phase and the output is a counting matrix.

The adopted windowing technique in this work is the sliding window, explained in the next sub paragraph.

3.4.1 Sliding Window

The rationale behind this windowing technique is very simple: there are two attributes, window size and step size. The window size defines the duration of the temporal interval to capture the records in a log file and then grouping them in a set. The step size instead is often smaller than the window size and indicates the amount of time (it is very often a multiple of the window size) for which the sliding window must be shifted. For obvious reasons there are sets having some records in common because there is an overlap among different windows.

Different from fixed windows, sliding windows consist of two attributes: window size and step size, e.g., hourly windows sliding every five minutes. In

general, step size is smaller than window size, therefore causing the overlap of different windows. The number of sliding windows, which is often larger than fixed windows, mainly depends on both window size and step size. Logs that occurred in the same Sliding Window are also grouped as a log sequence, though logs may duplicate in multiple sliding windows due to the overlap. Figure 3.9 synthesizes what has been said in this sub paragraph.

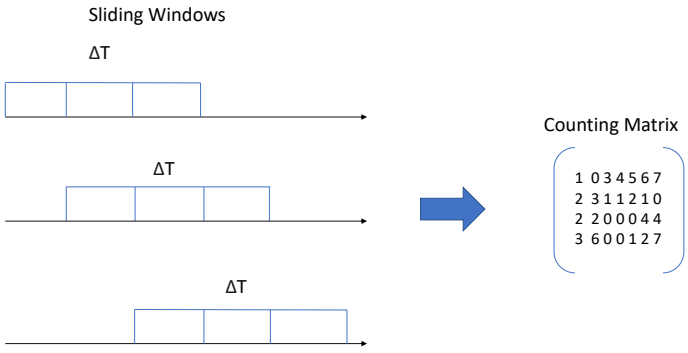

Figure 3.9: Feature Extraction Using Sliding Windows

Output Of Phase 3

The output here is a very big matrix that contains the occurrence of each record in each set. Specifically each row of this matrix will contain the number of times that a record appeared in a specific set.

3.5 Anomaly Detection

Therefore, the numerical data obtained in the previous phase serve as an entry point for applying thereafter some machine learning algorithms: here the choice fall on using algorithms belonging to both supervised and unsupervised learning in order to see which of the two types of learning was the most suitable to create an adequate model for detecting anomalies. In particular classification techniques are used, able to verify if a record was in a normal or abnormal state.

The chapter 5 is totally dedicated to this step since it is the last and most important of this process, reviewing the theory of the algorithms used for the specific kind of learning employed.

Output Of Phase 4

In this final stage the results are exclusively numerical: these are represented by the performance metrics used as quality indicators of the machine learning models utilized, produced by applying or not some particular validation strategies (the k-fold cross validation strategy of the sub paragraph 6.3.1 to be precise). However exhaustive details are given in chapter 6, the experimental part of the assignment.

Chapter 4

Programming Language

This short chapter informs the reader on which programming language the choice fell and the reasons behind this, paying particular attention on the version of the used software libraries that contains specific functions, classes and methods suitable for the conducted research.

4.1 Python

The development of a software tool that allows the processing and analysis of data cannot be separated from evaluations, of a more technical nature, on the type of language and libraries to be used: these choices determine, to a certain extent, the possibilities of implementation of a solution.

The decision of adopting a peculiar programming language rather than another was essentially driven by two factors:

- Ease Of Use.
- Richness In Terms Of Technical Contents.

Python is an interpreted, high-level and general-purpose programming language that combines simplicity of use (given its clear and readable syntax) and expressiveness very well. It is a widely used language in the prototyping phase, precisely because it allows to create a product quickly and with a relatively little effort.

Another very decisive factor for the choice of the programming language is: the availability of libraries on machine learning. Python boasts of a considerable number of libraries and frameworks oriented to data analysis and

machine learning, consolidated over time thanks to the unified contribution of researchers and developers.

In this work two different Python versions have been used: the **2.7** version for what concerns the log collection and the log parsing phases and the **3.6** version regarding the feature extraction and the anomaly detection steps.

4.1.1 Software Libraries

During the analysis various libraries for Python were used, the main ones and common to all the four parts are listed below along with the corresponding version and the specific step of the process at which that version was used:

- SciPy (version 1.5.2 for all the steps).
- NumPy (version 1.16.6 for log collection and log parsing and version 1.19.2 for feature extraction and anomaly detection).
- Scikit-learn (version 0.20.3 for all the steps).
- Pandas (version 0.24.2 for log collection and log parsing and version 1.1.2 for feature extraction and anomaly detection).

SciPy

The SciPy [10] (Scientific Python) is a Python software library that offers numerous scientific functions used for several routine operations in various mathematical areas.

NumPy

NumPy [11] and [12] (Numerical Python) is another software library for Python having the advantage of being open source. It is largely used in science and engineering and thanks to its simplicity it is used by any type of user, from the beginner to the most experienced.

Scikit-learn

Scikit-learn [13] is one more open source software library born for the machine learning in Python, in particular for the supervised and unsupervised learning.

Pandas

Pandas [14] and [15] is a further library makes available data structures particularly suitable for data analysis in Python thanks to its simplicity.

Part III

Models, Data Analysis And Conclusions

Chapter 5

Machine Learning Models

The chapter presents an initial review on the state of the art of the existing types of learning to then focuses on the technicalities concerning the machine learning models in conjunction with their most important tuned hyperparameters for realizing the supervised anomaly detection and the unsupervised anomaly detection.

5.1 Learning Typology

Predicting the evolution of a certain phenomenon basically implies the development of a model that can describe it and, consequently, its application to data detected over time. Machine learning uses algorithms that allow a machine to acquire knowledge through observation of reality, therefore learning from data analysis, with the aim of being able to make assertions about a certain event. This is made possible from the ability to abstract a general model from the set of learning data, in able to address and solve correctly new occurrences of this typology of problem. Machine learning algorithms are usually divided into three broad categories, which take into account both the nature of the problem (e.g., classification, regression, clustering) and the type of information available (e.g., desired input and output data). Very briefly, the three categories referred to are:

- Supervised learning, which allows you to determine a rule based on the presence of a label.
- Unsupervised learning, in which the algorithm deduces a possible structure from the data based exclusively on the input data.

- Reinforcement learning, which involves interaction with a dynamic environment in order to achieve a goal and, more generally, a behavioral strategy, adapting to changes by distributing a reward or reinforcement (positive or negative).

In addition to that, for the sake of knowledge, there is also a classification for the so-called hybrid learning (which fall into a separate category):

- Semi-supervised learning that is characterized by using all types of available data and not just those labeled as it happens in the supervised learning.
- Self-supervised learning is an unsupervised learning which faces the considered problem as if it were a supervised learning assignment in order to apply supervised algorithms. Doing this the obtained solution will be useful (in terms of modeling) for the initial encountered problem.
- Multi-instance learning belongs to the supervised learning, where there are single unlabeled data and clusters of labeled data: in this case the existing labeled data will be exploited as a knowledge base for predicting the labels for new clusters together with those unlabeled.

As already stated in paragraph 3.5 of chapter 3, anomaly detection methods are of two possible types: supervised anomaly detection (that obviously adopts the supervised learning) and unsupervised anomaly detection (that uses the unsupervised learning).

5.2 Supervised Anomaly Detection

As the word suggests this task is guided by the presence of a label that indicates if the system under analysis is in a normal or abnormal state, going to constitute a typical classification problem. A part of these labels are used for training the models employed: consequently a big number of labels implies better performances in terms of accuracy of the model. In the following there is a review of three models belonging to the supervised family: Logistic Regression, Decision Tree, and Support Vector Machine.

5.2.1 Logistic Regression

As all the regression algorithm, Logistic Regression is a predictive procedure that is non linear and it is designed to show the relation between dependent

variable and one or more independent variables, estimating probabilities with a logistic function with the following formula:

$$y = \frac{e^{-(b_0+b_1 x)}}{1+e^{-(b_0+b_1 x)}}$$

Where:
-x are the input values.
-b_0 and b_1 are the coefficients of input values.
-y is the output value to be predicted.

Figure 5.1 describes it visually. This model permits to search values for coefficients which minimize the error in terms of probability. First of all, the probability of an input (X) belongs to a specific class (Y = 1) is defined as follows:

$$P(X) = P(Y = 1|X)$$

Using the logistic function, it can be expressed in following way:

$$P(X) = \frac{e^{-(b_0+b_1 X)}}{1+e^{-(b_0+b_1 X)}}$$

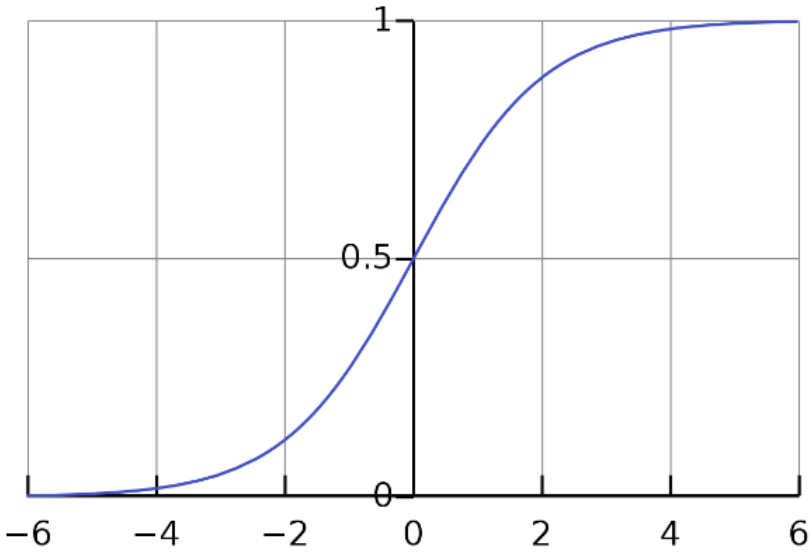

Figure 5.1: Logistic Regression

In this case the Binomial Logistic Regression was chosen among the existing ones, since permits to classify observations estimating the probability of a single observation that falls into a particular category (normal or anomaly).

Applying what has been said to the work that has been done (more details are here [5]), for any incoming group of logs will be created a vector on which will be applied a logistic function that has been trained in advance with the training data (the existing vectors with their corresponding labels). Later the logistic function will calculate the probability (whose value is between 0 and 1) for the new group of logs of being normal or abnormal. If this value is greater or equal than 0.5 then the group of logs will be considered abnormal; on the contrary will be considered normal.

5.2.2 Decision Tree

Decision Tree is another algorithm that belongs to the family of supervised learning algorithms: it can be used for solving regression and classification problems too. The goal is to create through its use a training model able to anticipate the value of a specific variable chosen as objective of the analysis by simply understanding selection rules thanks to the prior data (training data); in fact it is able to take a dataset of unknown data and extract a set of rules through which an individual can understand the problem and the results. This type of algorithm has the following advantages: low computation time, easy understanding of the results. The disadvantages instead are: could treat irrelevant data and it is prone to overfitting (the model fits too well to the training data losing in generality unlike the underfitting in which the model cannot acquire the underlying pattern of the data because there are few parameters in the model and a high classification discrepancy (high bias and low variance)).

In Decision Tree some division point are tried and tested using a regression cost function. The division with less cost is select. In the first division, all the features are considered and the training data are divided into group according to that division. For example, if there are 3 features, there will be 3 kind of divisions. For each one the accuracy of division is calculated and selected the best one. The cost function is the Sum Squared Error (SSE or Residual Sum Squared, RSS) between all data in the division:

$$RSS = \sum(y - prediction)^2$$

where y is the real observed value and prediction is the predicted value.

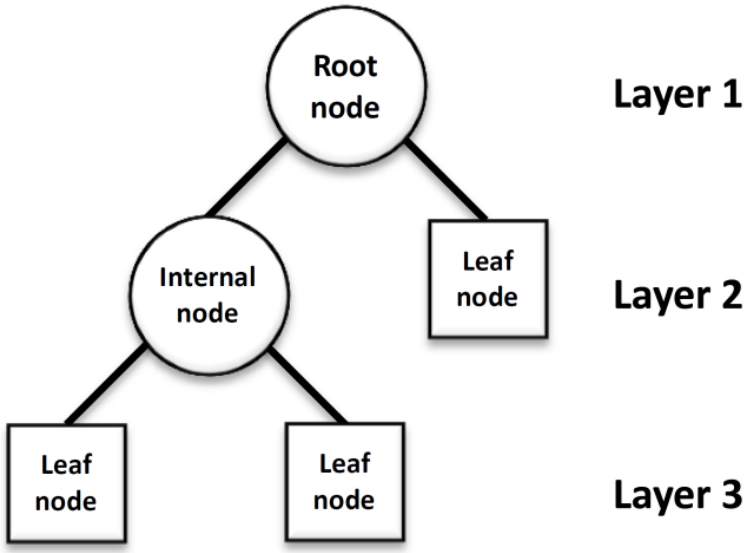

Figure 5.2: Decision Tree

For the anomaly detection, the tree is elaborated from the training data (represented by vectors with their labels), choosing with the usage of a metrics for each node (included the root node) the attribute that is more relevant for the analysis (more details are here [5]). When a new group of logs (converted in a vector) arrives, depending on the value of the predicate of every tree node the algorithm will cross the tree following a specific path until will reach a leaf having its corresponding status (normal or abnormal).

5.2.3 Support Vector Machine

In addition to the Decision Tree, among the algorithms capable of performing classification and regression tasks, there is also the Support Vector Machine or, more briefly, SVM: such an algorithm is based on the arrangement of the data in an n-dimensional space (where "n" is the number of feature of the considered dataset) and the calculation of a hyperplane that divides the elements belonging to different classes. The hyperplane calculation is based on two main factors: maximum margin, i.e. the maximum distance between hyperplane and the data in n-dimensional space, and accuracy which, briefly, indicates the precision with which the model predicts the belonging of a given to a class. The algorithm first tries to maximize accuracy and then the

maximum margin: however, if there are outliers in the dataset, the SVM is able to recognize and ignore them for prediction purposes. Figure 5.3 clarifies the concepts set out so far.

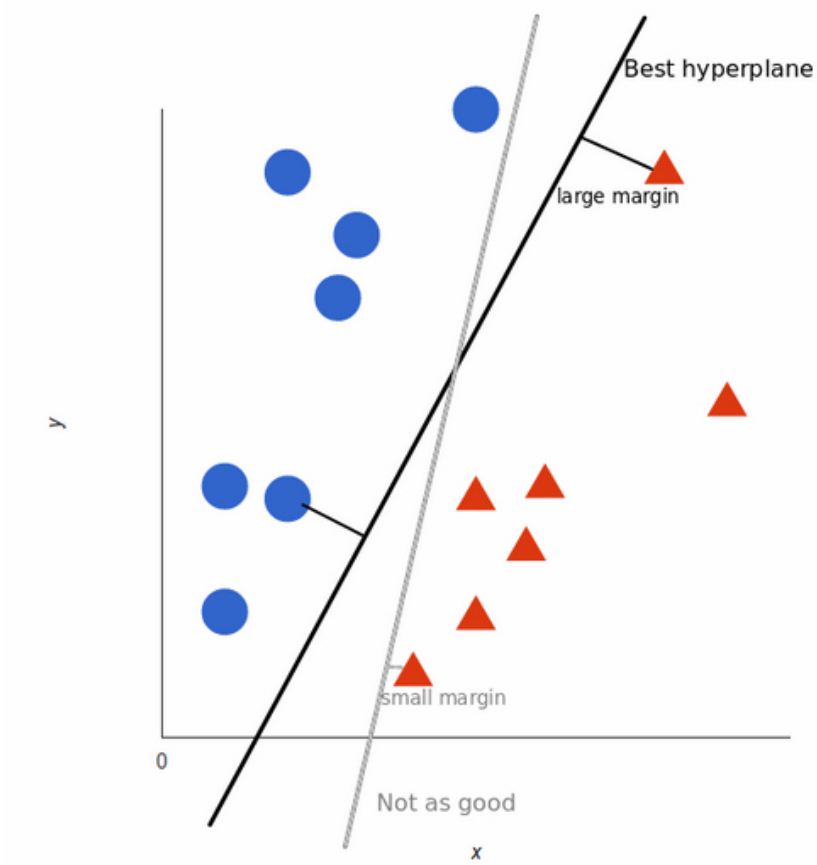

Figure 5.3: Support Vector Machine

Among the main features of the Support Vector Machine is the ability to identify automatically a hyperplane that divides the data into the respective classes of belonging also if these are not linearly separable: through its function called "kernel trick", the algorithm adds a dummy attribute (created on the basis of the other features already present) to increase the size of the space and thus make it possible to calculate a hyperplane. Another useful aspect made available by the algorithm is the ability to give weight to classes of belonging of the data: in many cases in fact (such as that of the research

developed here) the prediction on a specific class turns out to be more significant than on another and thanks to the different weights it is possible to address the results of the model if this did not identify the class clearly and clearly.

In order to optimize its performance, the SVM can be configured by means of some hyperparameters including the "constant C", the type of "Kernel" used and the parameter "Gamma". The first allows to manage the trade-off between errors in prediction and margin maximization: a model with a low C value tends to ignore points close to the hyperplane as the maximum margin considered for the decision increases; vice versa a very large value of C creates the so-called "hard-margin" for which a data comes always considered to belong to one class over another even if very close to the calculated hyperplane, which makes its identification more dubious. The "Kernel" parameter, on the other hand, indicates which type of formula must be used for the calculation of the hyperplane and can vary between linear, polynomial, sigmoid and RBF (radial basis function): each of these has different characteristics and processing complexity that they allow however, to create reliable models for data with even very different distributions. Finally, the Gamma parameter, strictly linked to the RBF type kernel, indicates, intuitively, how far the influence of a single point in the decisions of the pattern: High values values of it tend to create limits of "jagged" decision as they are very close to the points belonging to a class; in reverse small gamma values give more weight to points further away from the decision boundary than it will look more like a straight line. Too high gamma values generate the so-called problem of overfitting.

As well as for Logistic Regression and Decision Tree, also for SVM the vectors along with their corresponding labels are the training material (see [5] for a complete dissertion), composing the matrix coming from the output of the feature extraction phase: so in SVM if a new element will be placed above the hyperplane, will be considered as an anomaly; on the contrary if it will be above will be deemed normal.

5.3 Unsupervised Anomaly Detection

5.3.1 Principal Component Analysis

Principal Component Analysis (often known as PCA) is a methodology used to reduce the dimensionality of the available data. The followed principle is: starting from a very high number of dimensions, the goal is to project

into a new coordinate system the original data but using only a subset of the original components. Adopting this procedure, the objective to be achieved is to be able to find the components containing the highest variance among the data available, without losing essential information. Figure 5.4 can help with what has been said.

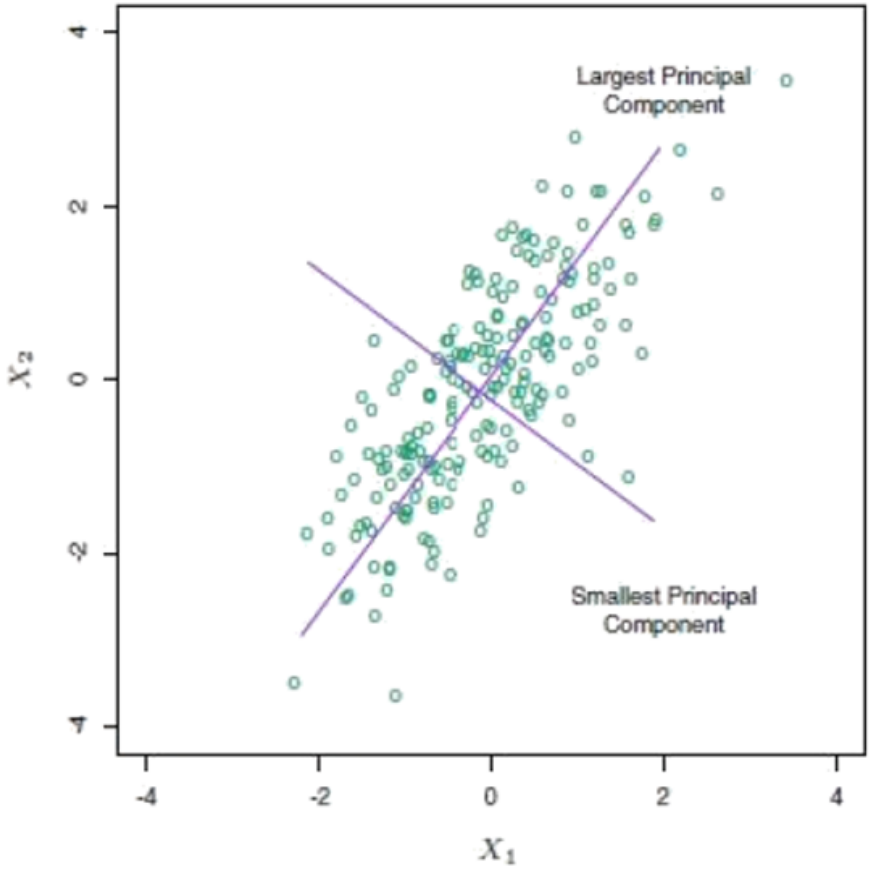

Figure 5.4: Principal Component Analysis

Focusing on the problem of the detection of anomalies, with the help of PCA every group of logs is put in the form of a vector: indeed the algorithms tries to find the recurrent schemes among the components of the vectors (see [5] for more stuff).

5.3.2 Hierarchical Clustering

Hierarchical Clustering is based on the hypothesis that data points next to each other are likely to have more features in common with respect to the

ones distant from each other. The behaviour is very simple: as the name of the algorithm suggests, it performs a hierarchical division of the input data that can be visualized through a suitable diagram called dendrogram representing a tree. Figure 5.5 portrays the architecture of a dendrogram (in that figure the dataset is made up of letters).

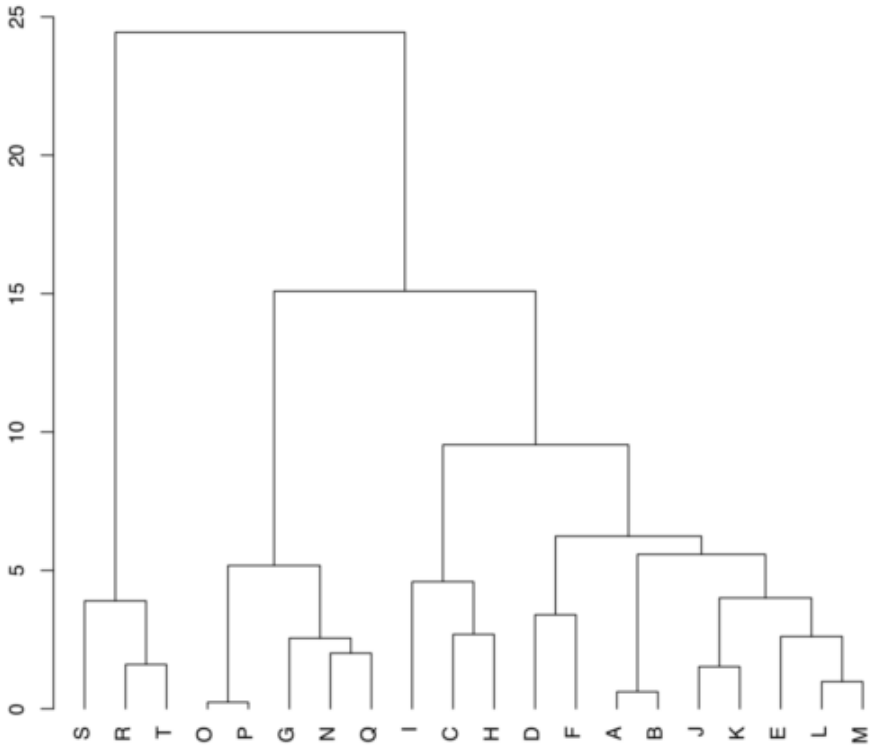

Figure 5.5: Hierarchical Clustering

Referring to the detection of anomalies, this algorithm (like the other ones) needs a training step that, in this case, is divided into two phases (more specifics can be retrieved here [5]).

The first phase consists in representing the logs of the previous phase in the form of vectors; therefore there will be a differentiation between normal and abnormal vectors using an agglomerative hierarchical clustering that exploits a bottom up approach through which inserts each vector in a different cluster and gradually merges clusters two by two. Hence at the end of this phase there will be two large clusters (one called normal and the other one abnormal) from which for each of them will be extracted a vector that best describes the others after the calculation of its centroid.

In the second phase every vector is reviewed through the computation of

the distances between itself and the salient vectors of each cluster produced in the previous phase: if the lowest distance is lower than a prefixed value, this vector will become a new element of the closest cluster (the one with the smallest distance). On the contrary a new cluster containing this vector will be constructed.

Thereafter the training there is the search for anomalies: if the lowest distance between an incoming group of logs (transformed in a vector) and the salient vectors of the last step of the training phase will be smaller than a fixed amount, the considered group will be marked as normal; on the other hand the incoming group of logs will be deemed abnormal or normal if the closest cluster is respectively abnormal or normal.

Chapter 6

Experimental Results

This chapter reviews from a numerical and technical point of view the various machine learning algorithms performed on the reference dataset, comparing them through the usage of the three most recurrent metrics in this field.

6.1 Evaluation Metrics

For the comparison of the results obtained by applying the machine learning models to the sampled data, different metrics have been used which allowed the evaluation of different aspects of the analysis carried out. In particular, if an unbalanced dataset is analyzed (like in this case), some specific evaluation parameters are used such as precision and recall: both values are calculated on the results of one single class and therefore allow to evaluate the goodness of a model regardless from any imbalance of the input dataset.

As shown below, the precision as the value which indicates how many samples, among all those classified by the model as belonging to one certain class, they have been correctly labeled; recall indicates, instead, how many samples there are been labeled correctly among all those belonging to a certain class present in the entire dataset: both parameters are considered optimal the closer they are to 1. F1 score instead indicates the harmonic mean of precision and recall (the harmonic mean can be expressed as the reciprocal of the arithmetic mean of the reciprocals of the given set of observations): it can assume all values between 0 and 1 and is optimal the higher its value is; as can be seen from the formula below, indeed, the f1 score is the higher the more the recall and the precision approach 1, which is the optimal value for both parameters.

Furthermore the accuracy of a model by definition indicates the relationship between the number of correct predictions and the total of predictions made: this parameter helps, in the algorithm optimization phase, to understand if the modification of the model brings improvements to the performances or not.

$$Precision = \frac{\#AnomaliesDetected}{\#AnomaliesReported}$$

$$Recall = \frac{\#AnomaliesDetected}{\#AllAnomalies}$$

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

6.2 Dataset

In machine learning in order to start using the models a splitting strategy of the dataset must be chosen for creating a training set and a test set. The training set is defined as that part of the dataset dedicated to the creation of a mathematical model, while the remaining part of the dataset is used to evaluate the model's performance previously generated. This aspect plays a very important role in the data selection phase as it actively affects the number of values that the model can have for the analysis: usually, most of the data in the dataset are dedicated to the training set, leaving only a small part in the test set of values useful for evaluating the predictive model.

Hence there is a training phase that involves the use of algorithms whose main function is to calibrate the parameters on the basis of the characteristic values of the model provided in input (i.e., the training set); the output generated by the training process is a model defined in its own characteristic parameters, which will constitute the crucial part of the entire predictive analysis: the prediction, in fact, of the future values (normal or anomaly in this case) of a variable.

Once the training process was completed, the model thus defined is tested on a portion of the dataset, in order to determine its performance or, in other words, the accuracy of the results generated. The test phase includes the following steps:

1. The data prepared for the testing phase (i.e., the testing set) are sent to the model.
2. The output generated by the model, ie the predictions on the test data, is compared with the real values of the starting set via an accuracy index.

During the research has been adopted this data division policy for all the three supervised methods: the first 80% of the data as the training data and the remaining 20% as the testing data because only having a "wide" knowledge of the past there is the possibility of having a good result in the testing phase.

Finally by default the window size and step size of sliding windows used as grouping technique for the feature extraction phase (for more details see the paragraph 3.4 of chapter 3) are set to six hours and one hour, respectively. Table 6.1 summarizes the cardinalities of the dataset after the splitting into training and test sets.

	Training Set	Test Set
Size	10262	2566

Table 6.1: Cardinality Of Training Set And Test Set

6.3 Predictive Models Application

In the application of the different models to the dataset the strong unbalances of the two classes (normal or anomaly) has been maintained in order to, as already said, simulate a context that is the closest to the real one.

6.3.1 Supervised Algorithms

The order of execution of the algorithms is the same as that used to explain their working mode in the previous chapter (both for the supervised and unsupervised ones).

Logistic Regression

The first algorithm among the models of this category that has been executed is the Logistic Regression that obtained the following values using these hyperparameters: C = 100, penalty = 0.01, class_weight = balanced.

Logistic Regression	Test Score
Precision	0.978799
Recall	0.731836
F1 Score	0.837491

Table 6.2: Logistic Regression Test Accuracy

Decision Tree

The second one is the Decision Tree that obtained the following values using these hyperparameters: criterion="gini", splitter="best", max_depth=None.

Decision Tree	Test Score
Precision	0.982332
Recall	0.734478
F1 Score	0.840514

Table 6.3: Decision Tree Test Accuracy

Support Vector Machine

The third and last one is the Support Vector Machine (the linear version has been used) that obtained the following values using these hyperparameters: penalty='l1', C=1, tol=0.0001, class_weight='balanced'.

K-Fold Cross Validation Technique

Cross-validation is a statistical validation technique used for the evaluation phase. It is particularly indicated when the number of samples are not too much: there is only one parameter called k through which decide the proportions for the division of a dataset in order to obtain a uniform distribution of the data. The choice of this parameter is not trivial: indeed it must be carefully chosen for positively impacting on the model performances.

In the following there is the application of the models with the help of this procedure, going to see the similarities and the advantages/disadvantages of not using it in terms of the evaluation metrics defined before.

Support Vector Machine	Test Score
Precision	0.978910
Recall	0.735799
F1 Score	0.840121

Table 6.4: Support Vector Machine Test Accuracy

Logistic Regression

The first model is the Logistic Regression that obtained the following values using the same hyperparameters used before, with k = 10.

Logistic Regression	Test Score
Precision	0.764691
Recall	0.867284
F1 Score	0.802764

Table 6.5: Logistic Regression Test Accuracy Using K-Fold Cross Validation

Decision Tree

The second one is the Decision Tree that obtained the following values using the same hyperparameters used before, with k = 10.

Support Vector Machine

The third one is the Support Vector Machine (the linear version has been used) that obtained the following values using the same hyperparameters used before, with k = 10.

6.3.2 Unsupervised Algorithms

In this sub paragraph the results related to the execution of the algorithms are reported in terms of the same metrics adopted for the supervised models, in order to guarantee a uniformity in the comparison.

Decision Tree	Test Score
Precision	0.819004
Recall	0.730864
F1 Score	0.771168

Table 6.6: Decision Tree Test Accuracy Using K-Fold Cross Validation

Support Vector Machine	Test Score
Precision	0.762919
Recall	0.880864
F1 Score	0.809321

Table 6.7: Support Vector Machine Test Accuracy Using K-Fold Cross Validation

Principal Component Analysis

The first tested model is the PCA that obtained the following values using the default values for what concerns the hyperparameters.

Hierarchical Clustering

The second model of this category is the Hierarchical Clustering that obtained the following values using the default values in reference to the hyperparameters.

6.4 Results Evaluation

This paragraph wants to be a very concise summary able to guide a technician to choose the best machine learning model for his needs. To do that the pros and cons of these two approaches are reported.

Supervised techniques turn out to be the clearest:

1. An important point in favor of this is given by the presence of the labels.
2. The analyzed supervised models reaches a high value for the precision, while the value of the other performance measures depends on the input data source and some parameters of the adopted grouping technique.

Principal Component Analysis	Test Score
Precision	0.643278
Recall	0.691124
F1 Score	0.623365

Table 6.8: Principal Component Analysis Test Accuracy

Hierarchical Clustering	Test Score
Precision	0.612467
Recall	0.625531
F1 Score	0.600345

Table 6.9: Hierarchical Clustering Test Accuracy

As regards the unsupervised procedures is clear that:

1. Generally the unsupervised algorithms perform worse than those supervised.

2. By providing equal values to the same parameters of the same grouping technique used for both supervised and unsupervised models, the obtained results will show a discrepancy in terms of performances between the supervised and the unsupervised methodologies.

Chapter 7

Conclusions

The final chapter concludes this work, paying attention to the reasons that led to follow a certain path and produce the present master's thesis work.

Secondly, a hint is given to what could be some of the possible future developments of this work.

7.1 General Considerations

The practical needs of the workplace, in which the monitoring of computer systems particularly complex is essential for the proper functioning of all infrastructures technological, led to the conception and development of the research described. In this sense the work carried out was born with the aim of exploring the potential of machine learning applied to system log files.

The application and the subsequent optimization of different machine learning models therefore have allowed to analyze the data source, identifying at the same time any critical issue present for it. In particular, a more practical interpretation focused on the practical aspect has been provided, trying, therefore, to bring the analysis carried out within the context under consideration. So far, in fact, the problem has been addressed with the aim of improving as much as possible the accuracy of the different employed models. In other words, the tests performed, according to the different configurations, were used to determine the algorithms and parameters such as to generate the best possible result.

The work carried out has therefore made it possible to identify the potential of machine learning applied in the context of monitoring IT systems in the enterprise context: on one side, the results obtained show ample room for

improvement in terms of accuracy for what concerns the unsupervised models, achievable with use of data collected in other ways and hence through an expansion of the dataset; on the other side they demonstrate the concrete possibility of exploiting such techniques illustrated to implement predictive maintenance policies and to allow the addressing of future research towards the specific study of some mathematical models, setting instead others aside.

Logs are crucial and constantly used everyday for detecting anomalies but with the progress of technology and the consequent increase in terms of complexity of the computing systems a manual inspection is not at all feasible. In this scenario, within this work has been adopted an automated log analysis tool and several anomaly detection methods in order to provide to an audience linked to the company environment a valuable tool capable of detecting the presence of any abnormal behavior.

7.2 Future Works

In order to improve the performance of the mathematical models studied and thus allow a more reliable classification of the samples, some improvements (to be defined) could be implemented in future versions:

- The first future development that could be realized is the integration of the previous research (conducted in the same company where the present work has been developed) and of the latter into a single tool that is easily usable by the final users (the company's technicians).
- A second possible enhancement in future may be a large-scale reproduction of the studied classification models in order to make them accessible to more than one user so as to identify further improvements and weaknesses for each of them.
- The third and last future advancement might be the implementation of an alerting system based on any anomalies envisaged by the model presented in the master's thesis.

As can be easily guessed, the aspects to be investigated are innumerable and the possibility of hosting new analysis methodologies is, without a doubt, one of the aspects more interesting of this assignment, together, clearly, with the ability to detect the evolutionary state of a system.

Bibliography

[1] Wikipedia contributors, *Application Service Management*. https://en.wikipedia.org/w/index.php?title=Application_service_management.

[2] Oracle, *Database Error Messages*. https://docs.oracle.com/database/121/ERRMG/title.htm.

[3] Oracle, *Oracle Database Documentation*. https://docs.oracle.com/en/database/oracle/oracle-database/index.html.

[4] Burleson, D., *Understanding Oracle Features And Options*. http://www.dba-oracle.com/art_so_oracle_standard_enterprise_edition.htm.

[5] Shilin He, Jieming Zhu, Pinjia He, Michael R. Lyu, *Experience Report: System Log Analysis For Anomaly Detection*. https://jiemingzhu.github.io/pub/slhe_issre2016.pdf.

[6] Pinjia He, Jieming Zhu, Shilin He, Jian Li, Michael R. Lyu, *An Evaluation Study On Log Parsing And Its Use In Log Mining*. https://jiemingzhu.github.io/pub/pjhe_dsn2016.pdf.

[7] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, Michael R. Lyu, *Tools And Benchmarks For Automated Log Parsing*. https://arxiv.org/pdf/1811.03509.pdf.

[8] Adetokunbo Makanju, A. Nur Zincir-Heywood, Evangelos E. Milios, *A Lightweight Algorithm For Message Type Extraction In System Application Logs*. https://web.cs.dal.ca/~makanju/publications/paper/tkde11.pdf.

[9] Adetokunbo Makanju, A. Nur Zincir-Heywood, Evangelos E. Milios, *Clustering Event Logs Using Iterative Partitioning*. https://web.cs.dal.ca/~makanju/publications/paper/kdd09.pdf.

[10] SciPy community, *SciPy Reference Guide*.
`https://docs.scipy.org/doc/scipy-1.5.2/scipy-ref-1.5.2.pdf`.
[11] NumPy community, *NumPy Reference*.
`https://numpy.org/doc/1.16/numpy-ref.pdf`.
[12] NumPy community, *NumPy Reference*.
`https://numpy.org/doc/1.19/numpy-ref.pdf`.
[13] Scikit-learn developers, *Scikit-learn User Guide*.
`https://scikit-learn.org/0.20//_downloads/scikit-learn-docs.pdf`.
[14] Wes McKinney & PyData Development Team, *Pandas: Powerful Python Data Analysis Toolkit*.
`https://pandas.pydata.org/pandas-docs/version/0.24.2/pandas.pdf`.
[15] Wes McKinney and the Pandas Development Team, *Pandas: Powerful Python Data Analysis Toolkit*.
`https://pandas.pydata.org/docs/pandas.pdf`.