Anomaly Detection in Log Files Using Machine Learning

Philip Björnerud

Computer Science and Engineering, bachelor's level
2021

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering
Abstract

Logs generated by the applications, devices, and servers contain information that can be used to determine the health of the system. Manual inspection of logs is important, for example during upgrades, to determine whether the upgrade and data migration were successful. However, manual testing is not reliable enough, and manual inspection of logs is tedious and time-consuming. In this thesis, we propose to use the machine learning techniques K-means and DBSCAN to find anomaly sequences in log files. This research also investigated two different kinds of data representation techniques, feature vector representation, and IDF representation. Evaluation metrics such as F1 score, recall, and precision were used to analyze the performance of the applied machine learning algorithms. The study found that the algorithms have large differences regarding detection of anomalies, in which the algorithms performed better in finding the different kinds of anomalous sequences, rather than finding the total amount of them. The result of the study could help the user to find anomalous sequences, without manually inspecting the log file.

Keywords

Anomaly detection, Log files, Data representation, Machine learning, Clustering
Acknowledgements

I am very grateful for the support given by my supervisors. I would like to thank Saguna Saguna and David Olsson Granlund for their help and guidance during the process of writing this thesis. I would also thank Mobilaris for providing the data to make this study possible.
## Contents

1 Introduction  
1.1 Background and Motivation ........................................ 1  
1.2 Collaboration with Mobilaris ..................................... 2  
1.3 Problem definition .................................................. 2  
1.4 Goal .................................................................. 3  
1.5 Objectives ................................................................ 3  
1.6 Methodology .......................................................... 3  
1.7 Delimitations ........................................................... 3  
1.8 Thesis structure ....................................................... 4  

2 Related work  
2.1 Data representation on logs .......................................... 5  
2.2 Machine Learning Techniques ...................................... 6  

3 Theory  
3.1 Machine learning ....................................................... 8  
3.2 Clustering ............................................................... 9  
3.3 Principal Component Analysis (PCA) .......................... 11  
3.4 Logs structure ........................................................ 12  
3.5 Anomaly data .......................................................... 12  
3.6 TF-IDF .................................................................. 13  
3.7 Sliding window ........................................................ 13  

4 Method  
4.1 Overview ................................................................. 14  
4.2 Data preparation and parsing .................................... 16  
4.3 Data representation .................................................. 17  
4.4 Approaches ............................................................. 21  
4.5 Implementation ....................................................... 24  

5 Evaluation  
5.1 K-means with Feature Vector Representation and PCA .... 28  
5.2 K-means with IDF representation and PCA .................. 31
5.3  DBSCAN with Feature Vector Representation and PCA   . . . . . . . 33
5.4  DBSCAN with IDF representation and PCA   . . . . . . . . . . . . 35
5.5  Discussion and analysis   . . . . . . . . . . . . . . . . . . . . . . 36

6  Conclusions and future work   39

References   40
1 Introduction

In today’s world, IT systems are complex and large, where log files play an important role in debugging and finding system errors, and sometimes analyzing the log files is the only way of finding the cause of the system failure. Large and complex systems tend to generate a large amount of log data and as the data grows larger, manually detecting errors and bugs becomes more difficult and tedious. The cause of an unhealthy system can be due to error by devices, human errors, or a bug in the system. A common method for analyzing a system failure is to use anomaly detection. Anomaly data might also indicate something interesting in the system, and detecting that data can play an essential role in revealing system failures or the health of the system [25].

1.1 Background and Motivation

Log analysis is an important process to determine the health of the system and to discover the cause of system failures. The logs are generated by applications, devices, and servers and contain important information about the system. Failures can occur during upgrades and data migration, and a common practice is to manually inspect the logs, which can be very tedious and time-consuming for the end-user. It often requires qualified knowledge of the system to manually inspect the log files, which is not always possible. Different services are created by different developers that can change over time. There are some common techniques for manually analyzing the log data. However, the accuracy and effectiveness of these techniques are limited. A common method for analyzing large log files is to keyword search for log entries [13]. An example of a key search at Mobilaris log files could be “error” and “warn”. Another common way for analyzing large log files is to find anomalous log entries that are previously known by experience. Both mentioned debug practices are inefficient and time-consuming if the log file is large and complex. Some cases of system failure could also occur because of a specific log sequence, it often requires special expertise to understand the behavior of the logs. Such expertise can be expensive to train. It’s also not applicable for an end-user to foresee all problems and numerous unknown problems can occur. The reason for considering the application of
machine learning is that machine learning does not rely solely on a user for finding anomalies.

### 1.2 Collaboration with Mobilaris

Mobilaris is a global leader in location-based intelligence with offices in Sweden and the United States. One of the Mobilaris products is the Mobilaris Mining Intelligence, which offers the underground industries a decision support system, by providing real-time situational awareness. This is done by tracking the underground employees, equipment, vehicles, and machines and presenting this in a 3D map. This research work is carried out in close collaboration with Mobilaris due to their interest in applying machine learning algorithms to data from the log files for detecting anomalous log sequences and see what possibilities it can bring to troubleshooting their systems. Mobilaris has supplied all of the data required for this study.

### 1.3 Problem definition

A reliable and cost-effective method for analyzing log files and detecting system failures can have an important role for a company. Machine learning doesn’t rely on individual expertise and the advantage of using machine learning algorithms can have an impactful outcome.

Anomaly data are also referred to as outlier, which is something differing from the majority of data and can be a sign of something bad or abnormal in the system. In this thesis, we will investigate the possibilities machine learning can have for analyzing the log files at Mobilaris, by finding anomaly data in the log files. There is no known most suitable machine learning algorithm for detecting anomalous sequences at Mobilaris log files, therefore different machine learning algorithms will be tested and evaluated. This study will also test different data representation models, which is an important factor for finding unusual structures in data, which will also be evaluated. In this study, Mobilaris will also provide real-world data for the log analysis. We will analyze the data in these log files to detect anomalous log sequences to enable system administrators to detect problems and system failures without having to inspect each log file manually.
1.4 Goal

The goal of this thesis is to use machine-learning algorithms to find anomalies in log entry sequences. The solution should help the system managers and operators to troubleshoot the system and to help the team at Mobilaris in finding bugs and causes of system failure and interruption.

1.5 Objectives

The main purpose of this thesis is to use machine learning algorithms that can find anomalous sequences for the Mobilaris logs.

- **Objective 1:** Is to parse and prepare the data from the received file.
- **Objective 2:** Is to represent the log entries as numerical values for the machine learning algorithms.
- **Objective 3:** Use machine learning algorithms for finding anomalous log sequences in Mobilaris log file.

1.6 Methodology

In this thesis, we conduct quantitative research where we conduct experiments using real-world data. This also involves doing exploratory research by analyzing data provided to us by Mobilaris for finding anomalous log sequences. To be able to answer and solve the problems, which is presented in section 1.3 we have first studied the related work in chapter 2. The machine learning algorithms and data representation are chosen from there. Because it is not known which method will be best for the Mobilaris situation, different methods will be tested out and compared. Additionally, some literature related to the problem is studied and suitable metrics will be selected to evaluate the result.

1.7 Delimitations

This study will not discuss and focus on the details of the machine learning algorithms, but rather the effectiveness in finding anomalies in Mobilaris log files with the algorithms. Mobilaris system is very large and uses numerous different
services, because of the time limit and the complexity of the system we will only focus on log files from one service.

1.8 Thesis structure

The remaining part of this thesis will be structured as follows.

- **Chapter 2, Related work:** The previous work, which has be done in the same area will be reviewed.

- **Chapter 3, Theory:** An overview on the theory will be presented.

- **Chapter 4, Method:** Overview of the method used in this study will be presented.

- **Chapter 5, Evaluation:** Presenting the result from our study.

- **Chapter 6, Discussion:** Discussing the result which is obtained from the result.

- **Chapter 7, Conclusions and future work:** Summary of this work and future work will be discussed in this section.
2 Related work

In this section we will describe related research topics. The topics will cover data representation on logs and different machine learning algorithms. The topics which the researcher are presenting are focusing on the methods used for anomaly detection.

2.1 Data representation on logs

There are numerous different types of logs and log messages that can be highly diverse. The content of the log messages can be numerical or textual and the values can consist of constants and variable parts. The constant part stays the same in each log message and the variable part does not[9]. An example of such a log message could be “level=error, msg=could not find mapping, ip=192.000.04”. Level and “msg” are considered as constant parts because they will never change. However, the IP is considered as a variable part because IP is not fixed.

Numerical and textual values are meaningful, and representing the numerical and textual values is an important part for machine learning algorithms. However, it does not exist any standard for representing the log data, which is the reason for studying several different methods.

Textual data is in some cases considered as valuable data and can have an important role in machine learning algorithms [20]. In cases where logs consist of textual data, some natural language techniques can be applied. One of the more common natural language techniques is the frequency-Inverse Document Frequency (TF-IDF)[23]. TF-IDF is also a common technique used for representing data in anomaly detection because it takes into account not only the frequency of the textual data in a file but also the frequency of how many different files the textual data appears in. [23].

In paper [24], the natural language processing technique word2vec has been very effective and efficient in representing textual data in low dimensions.

Additionally, many researchers use log vectorization methods for representing the log data [13][7], where log sequences are turned into vectors and logs are turned
into log events using log abstractions techniques[7], where the constant part of the log messages are used. Log events are representing generic log messages printed out from the same log-print statement in the code. In the paper[13], sequences have multiple log events, where log events are weighted with two methods. A modified IDF-based weighting and contrast-based weighting[13]. The paper [13] also proposed that each log event has different importance, depending on how frequently the log event is appearing in different log sequences. Therefore it was proposed to use the weighting methods. After each log event was assigned a weight value, the researcher represented a vector by log sequences with N-dimensional space, where N was the unique number of log events in the file.

2.2 Machine Learning Techniques

Anomalies in log files can be detected using a variety of machine learning techniques. This subsection will focus on various supervised and unsupervised learning techniques that have been used in related fields, rather than feature extractions. The theories behind some of the Machine learning algorithms is described in section 3.

2.2.1 Supervised learning

In the paper [9], three different supervised machine learning algorithms were investigated for anomaly detection in system logs. Logistic regression, decision tree, and support vector machine (SVM). Because the logs that were used in their experiment were labeled either as an anomaly or normal it was possible to use supervised learning. To detect anomalies in the data, an event count vector was constructed from each of the log sequences in the file with a label. Thereafter, a logistic regression training model was established, which was used to classify if the log data was an anomaly or not. The paper [9], investigated also the decision tree algorithm which is a tree structure diagram. Another algorithm that the paper [9] investigated was the SVM, which is also a classification algorithm. All of the three supervised algorithms used the training instances as event count vectors together with the labels. When detecting anomalies in the data it demonstrated that the decision tree choices were more logical in selecting anomalies than the other two
algorithms for the developer. All of the algorithms performed well in detecting anomalies, however, SVM achieved the best overall accuracy.

### 2.2.2 Unsupervised learning

In the paper [22] it was proposed to use an unsupervised learning algorithm instead of a supervised in anomaly detection since supervised learning algorithms rely on a labeled training set, which is expensive to use and produce. Some supervised methods also rely on a new training set if the system changes and uses new log messages, in which an unsupervised learning algorithm can be more adaptable. The paper [22] selected LogCluster as their algorithm. LogCluster is a data clustering algorithm that detects patterns from log files. For measuring the performance LogCluster was used in 92 days. The implementation processed 296,699,550 log messages, of which 1,879,209 were classified as anomalies. They also discovered that some of the anomalies would be considered normal log messages rather than anomaly log messages.

In the paper [4] a study was conducted where two unsupervised machine learning algorithms were evaluated and their accuracy was compared for detecting anomalies. Principal component analysis (PCA) and k-means were implemented in the study. A primary feature Message Count Vector from parsed logs was constructed[4]. The PCA implementation detected 11195 anomalies out of 16838. The k-means implementation performed slightly better, 14817 out of 16838 anomalies were detected.

In the paper[5], the DBSCAN algorithm was used, which is a density-based clustering algorithm. It was proposed to use DBSCAN for finding anomalies in monthly temperature data. The investigation showed that it has several advantages over statistical methods for discovering anomalies.
3 Theory

In this section, an overview of the theory will be given. This section is presenting different cluster techniques and different methods for representing textual data. The theory which is presented in this section will be used in the method.

3.1 Machine learning

Machine learning is a form of artificial intelligence that has in recent years been applied in many different fields, such as speech recognition, computer vision, bio-surveillance, and pattern analysis [14]. The concept of machine learning is defined in [3], “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”. The most common machine learning algorithms are supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning.

3.1.1 Supervised learning

Supervised learning [17] is a technique where it involves training a model with inputs that has a correct output labeled to it, which allows the model to learn over time. Common models in supervised learning algorithms are classification and regression models, the fundamental difference between them is that classification is about predicting a output, while regression is about predicting an quantity.

3.1.2 Unsupervised learning

In unsupervised learning, the model tries to learn itself without labeled inputs. Unsupervised machine learning is often used for discovering hidden patterns or data groupings, without the need of labels. Common tasks in unsupervised learning involve cluster detection and pattern recognition[17].
3.1.3 Semi-Supervised Learning

In semi-supervised learning [11], the model works with partially labeled data and unlabeled data. Some of the semi-supervised algorithms are a combination of unsupervised and supervised learning algorithms.

3.2 Clustering

Clustering is defined as an unsupervised learning technique, the fundamental concept of clustering is to group similar objects together[1]. Since the cluster algorithms are grouping objects basis on their similarity, different distance measures are used. One of the more common approaches for defining the measure similarities is the euclidean distance[1].

Grouping objects has been effective in many areas, such as medical science, prediction modeling, and anomaly detection. However, there exist various different clustering methods, the common ones are hierarchical, partitional, grid, density-based, and model-based clustering techniques. The reason for different clustering methods is because there does not exist an exact definition of what a cluster is[1], therefore different clustering techniques use different cluster definitions in their algorithm.

3.2.1 K-means

As previously mentioned, the K-means algorithm is an unsupervised clustering algorithm and one of the simplest clustering algorithms [1]. K-means is a non-parametric method of clustering, which groups similar objects into clusters. The number of clusters is not learned by the algorithm but set by the programmer manually and each of the objects in the data set will be assigned to a cluster. K-means has some disadvantages with an arbitrary size, density clusters, and the dependence on the programmer choosing the K clusters. Fig 3.1 illustrates a K-means plot with three clusters.

There are 6 essential steps, of the inner-working for K-means algorithm.

- 1. **Determine the number of K clusters**
2. Initialize K points arbitrary as the cluster center

3. Assign all objects to the closest cluster centroid

4. Compute centroid of each cluster

5. Centroid is the new cluster center

6. If not converged, repeat steps 3, 4, and 5.

3.2.2 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a density-based clustering algorithm. DBSCAN has the ability to create arbitrary shapes of clusters. Clusters are determined by the density of the data points. Areas with a high density of objects are indicated as clusters and areas with low density are indicated as potentially noise and outliers[15]. The algorithm DBSCAN was constructed for the reason to discover the clusters of arbitrary size and the noise in the data set[15]. DBSCAN algorithm requires to manually set two parameters eps, $\varepsilon$ and MinPts. The parameter eps sets the radius of a neighborhood around
the data point and the parameter MinPts sets the minimum number of neighbors within the eps radius. To locate the clusters, DBSCAN starts with initializing an arbitrary point and continues to form clusters until the eps and MinPts criteria are satisfied. The points that are not in a cluster will be classified as noise. Fig 3.2 illustrates a DBSCAN plot, where the dark purple data points represent noise points.

Figure 3.2: DBSCAN, dark purple points illustrates noise data

3.3 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a multivariate statistical technique, which purpose is to extract the most valuable information from the data set and reduce the size of the data set by keeping the only important information[2]. In order to achieve this, PCA computes new variables, which are linear combinations of the initial variables. It is a common technique for reducing dimensionality when you have a high-dimensional data set, and make the data more meaningful. PCA can filter frequently recurring patterns in data, which makes PCA a common method
for anomaly detection.

### 3.4 Logs structure

Log entries from a file are typically unstructured. Each log entry can normally be divided into two different parts, the constant part, and the variable part. The constant part will stay the same in each log message and the variable part will not. In most cases, the constant part is the message which is directly printed out by statements from the source code. A typically log message from one of the Mobilaris TVS services can be seen in Fig 3.3. A typical log entry from the service will print out a level, message, timestamp and a caller.

![Figure 3.3: Typical log structure](image)

### 3.5 Anomaly data

Anomalies are patterns in data that are not considered normal behavior[6]. Anomalies in data can be caused by various reasons, for example, malicious activity, cyber intrusion, or system failure [6]. Anomaly data can be divided into three different types. Point anomaly, contextual anomaly, and collective anomaly. A single distinguished data point, which is not part of the rest of the data, is considered a point anomaly. Detecting point anomalies are easier than detecting a contextual anomaly, and collective anomaly. Contextual anomalies are groups of data points that are distinguished from the other data points. Collective anomaly is groups of data points which is considered as an anomaly to the rest of the data and do not have to be considered as an anomaly because of the position of the data points.
3.6 TF-IDF

Term Frequency Inverse Document Frequency (TF-IDF) is a weighting algorithm. TF-IDF calculates the value for every word in a document, while it also takes into consideration that words that do not occur in many different files will be assigned a higher weight than words that occur more often in different files [12].

TF measures the frequency a word appears in a file and the mathematical formula is, \( TF = \frac{\text{Number of times } t \text{ appears in file}}{\text{Total numbers of terms in file}} \). While IDF measures the importance of one word. If the word appears in many different files, the weighing value will be decreased. The formula for IDF is, \( IDF = \log \left( \frac{\text{Number of files}}{\text{Number of files with term } t \text{ in it}} \right) \).

There exist numerous modified versions of TF-IDF where some of the definitions can change depending on the problem. Common areas for using TF-IDF are text mining, text categorization, text clustering, and machine learning.

3.7 Sliding window

The sliding window algorithm uses a fixed window size that is formed over some part of data and the window will slide over the data and capture portions of it. Fig 3.4 illustrates a sliding window with a size of three.

```
array= [a b c d e f g h]

[a b c]
  [b c d]
    [c d e]
      [d e f]
        [e f g]
          [f g h]
```

Figure 3.4: Sliding window of size 3
4 Method

In this section, the methodology for solving the problem from section 1.3 will be given. Section 4.1 will present an overview on the method, section 4.2 will present how we are preparing the data, section 4.3 will present how we are representing the textual data in numerical value and section 4.4 are presenting the machine learning approaches.

4.1 Overview

In this thesis, two different unsupervised machine learning algorithms will be used for detecting anomalous sequences in Mobilaris log files, K-means and DBSCAN. The reason for choosing unsupervised learning is because unsupervised learning can learn from structures without labels in the data and hidden problems can be found in log sequences. Detecting anomalies in the data file is divided into different steps, which can be seen in Fig 4.1. Anomalies in the log file were labeled with the help of the employees from the Mobilaris.
Figure 4.1: Method for finding anomalies
4.2 Data preparation and parsing

The log file received from Mobilaris was from one of their services, Tag Vibration Service (TVS). TVS determines if a tag is moving in the mine by measuring the time between blinks. Log entries from the file consist of constant and variable data, which is represented by textual or numerical data. Before using the machine learning algorithms it is important to prepare and parse the necessary data. The model for data preparation and parsing can be seen in Fig 4.2. The first step is to parse the data into abstracted log messages. Log entries do not follow a strict format, which makes it hard to extract different features from it for representing the data. Therefore a log abstraction techniques will be used, which have been suggested by the paper [10]. Another reason for using the abstracting method is because this thesis will investigate the anomalies in log sequences and not log entries as individuals.

The purpose of the abstraction technique is to generalize the log entries and not rely on the log entry format. The variable part in the log entry will be filtered out and the constant part will remain. The second step is to assign an id to each of the unique log entries. Fig 4.2 gives an example of a file of eight log entries. After the log abstraction, the variable part of the log entries will be filtered out, which was the timestamp and the caller part of the log entry, and the remaining parts of the log entries are the level and “msg”, which are constant parts. Each unique log entry will be assigned with an id, and in figure Fig 4.2 example, it existed five different kinds of unique log entries.
4.3 Data representation

Data representation is an important step in finding patterns in data. An example of a good representation of data is to minimize the distance of data that is similar to each other and maximize the distance of data that is not. After the parsing and log abstraction technique, we will have a new log file with log id’s that is representing the log entries instead. The purpose of this thesis is to find anomalies data in log entry sequences. Therefore a sliding window will be used for extracting log entries sequences, see Fig 4.3. Because of the time limit, a sliding window of size five will only be used. The purpose of using a sliding window is to capture all different sequences with the size of the sliding window. The sliding window size five was chosen for the reason that most anomalous sequences are around that size.

Fig 4.3 gives an example of a sliding window with a size of four. The first sequence will be Logtype1, Logtype1, Logtype2, and Logtype3. The second sequence will
be Logtype1, Logtype2, and Logtype3, and Logotype3. The id’s consist of textual data, which is not applicable for the machine learning algorithms that will be used later on in this thesis. Therefore, in this thesis, we will use two different techniques for representing the data numerically. The idea behind the data representation techniques is to group the multiple log entries from the sliding window into a single data point, which will be represented in a vector in N-dimensional space, N is the unique number of log entries in the log file. In figure example Fig 4.4 there exist five unique log entries in the log file, each sequence will then be represented in a five-dimensional vector, where each unique log entry will have a fixed place in the vector. The vectors will later on, be used in the machine learning algorithms as input.

Our approach to the problem does not consider the order of the log entries that appears in the window. This is because Mobilaris is using threads in their services, which will result in different orders for the same service request, and sometimes a clock drift problem can occur, which will cause failed synchronization.

![Sliding window for data representation](image)

Figure 4.3: *Sliding window for data representation*

### 4.3.1 Feature Vector Representation

The feature vector representation is a method for representing the textual log entries as numerical values in the log sequence. The dimensionality of the vector will be the same as the number of unique log entries in the entire log file. Each log will have its fixed place. The feature vector representation will count the frequency of each log entry appearing in the sliding window. In Fig 4.3 example we have five
unique log entries in the log file. As seen in the example Fig 4.4, the first sliding window sequence will have two logtype1’s and one logtype2 and one logtype3, which will generate a vector of \([2,1,1,0,0]\).

Advantages and disadvantages of using the feature vector representation as data representation for the machine learning algorithms,

**Advantages**

- Capture anomaly behavior around the frequency of some log entries in a sequence. Some log sequences can be considered as an anomaly if it is appearing multiple times in a sequence, which can be indicated as a bad system behavior.

**Disadvantages**

- The rareness of a log entry will not be represented. This is taking into consideration in the modified IDF data representation.

\[
\begin{align*}
\text{Log sequence} & \quad \rightarrow \quad v_1=[2,1,1,0,0] \\
\text{Log type1, Log type1, Log type2, Log type3} & \\
\text{Log sequence} & \quad \rightarrow \quad v_2=[1,1,2,0,0] \\
\text{Log type1, Log type2, Log type3, Log type3} & 
\end{align*}
\]

*Figure 4.4: Feature Vector Representation*

### 4.3.2 IDF-representation

The IDF representation is a weighting method for representing the log entries as a numerical value in the log sequence. The dimensionality of the vector will be the same as the number of unique log entries there are in the log file. In IDF
each log will also have a fixed place in the vector. However, instead of counting
the frequency of the occurrence of the log entries in the sliding window, IDF
will assign each log entry a weighting value if it appears in the sliding window.
Different log entries can have different importance. Therefore we will use a
modified IDF method, which is a common weighting method[13]. The equation
for the IDF is, \( IDF(t) = \log\left(\frac{\text{Number of files}}{\text{Number of files with term } t \text{ in it}}\right) \).
In the modified version, we will label each log entry as a term and
each sequence as a file. The equation will therefore be, \( IDF(t) = \log\left(\frac{\text{Number of sequences}}{\text{Number of times the log entry appears in different
sequences}}\right) \). Log entries that appear rarely will get a higher value than log entries
which appear often in sequences. Fig 4.5 illustrates the first two sliding window
scenarios from Fig 4.3 with an IDF weighting method. In the Fig 4.3 file there
will be a total of six sliding window sequences, which logtype1 will appear in two
different sliding windows, logtype2 will appear in three different, logtype3 will
appear in all, which is the reason for weighting value zero for logtype3 if you use
IDF equation.

Advantages and disadvantages of using the IDF data representation for the
machine learning algorithms

**Advantages**

- The value for each log entry will depend on the frequency of its appearance.
  Logs that are appearing rarely will have a higher value and will have a greater
  impact.

**Disadvantages**

- Difficult for using as an online analysis since all IDF weighting values have
to be updated when a new log entry appears.
4.4 Approaches

4.4.1 K-means with different thresholds and PCA for visualising

The vectors from the feature vector data representation and the IDF data representation were used for building two K-means models. The K-means algorithm is a widely used unsupervised learning algorithm. The algorithm was selected because of its simplicity. K-means cannot detect anomalies alone, which is the reason we are using a threshold for detecting the anomalous sequences in the data file. As previously mentioned, K-means is a non-parametric method of clustering, that groups similar objects into clusters[1], which is useful for getting an insight into the data. The number of clusters K is not set by the algorithm but set manually by the programmer. The method we used for finding the K value, was the elbow method[16]. It is a method used for finding the best number of clusters by looking at a chart and find where it forms an elbow. As an example in fig 4.6, three can be observed to be the optimal K value after applying this method. The values for plotting the chart were done by calculating the sum of squares(SSE) [19] result from each value of K and then plot it.

Identifying anomalies with only the k-means algorithm is not possible and a threshold will be used for finding anomalies in the K-means approach. In this thesis, three different threshold ratios were tested and compared. In each cluster
where the data points stayed out of the threshold ratio will be counted as an anomaly. To compare the data points to the threshold ratio, we calculated the distance for each data point to its centroid. The method used for calculating the distance was euclidean distance, when each of the distances was obtained they were compared with the threshold ratio.

PCA is a statistical method for reducing the dimensions. Due to the high dimensionality in our data we will use PCA to reduce the dimensions to two, which is for visualizing the data.

![Figure 4.6: Optimal K value](image)

### 4.4.2 DBSCAN and PCA for visualising

The vectors from the feature vector data representation and the IDF data representation were also used for building two DBSCAN models. DBSCAN is a density-based algorithm, which performs well with arbitrary shapes and sizes of the data. The algorithm was selected because it can label noise data by itself. Noise data is a data point that does not belong to any cluster and in our case,
we will label the selected noise data as anomalies. The number of clusters is also
decided by the algorithm. However, the programmer is required to set values for
two parameters, eps, and minPts. The parameters will set the rules for which
data points will belong to a cluster or which data points will be labeled as a noise
point. According to [8], the parameter minPts should be equal to the number
of dimensions plus one. The method for calculating the best eps value has been
suggested by the article [18]. The article suggested for finding the most suitable
epsilon is done by calculating the distance to the nearest n points for each point
and then plot it. The point where the line is starting to increase is the most optimal
eps value. The Fig 4.7 illustrates the plot for finding the best eps value, which is
0.007.

PCA will also be used here for reducing the dimensions for visualizing
purposes.

Figure 4.7: Optimal eps value
4.5 Implementation

The programming language that was used was Python and in this thesis we mostly used jupyter notebook, which is a script framework for Python. It was mainly used because jupyter notebook enables quick debugging and it is easy to run. Various Python and machine learning libraries were used, which was needed for this study. Python was selected because it is a powerful language for preparing the data and it is also designed with features, which is useful for data analysis and visualizing. Before abstracting the log entries we used a split() method from python, which breaks up a string at the specific separator and returns a list of the string. This method was used for extracting out the constant parts from the log entries. A code snippet from the implementation can be seen in Fig 4.8, where we extracted level, caller and the message from the log entry.

```python
with open('/Users/philipbjornerud/Downloads/tvs_tampered.log', 'r') as infile:
    for line in infile:
        if len(line.split()) == 0:
            continue
        if "level" in line:
            level=(line.split('level":"'))[1].split('"') [0]
            caller=(line.split('caller":"'))[1].split('"') [0]
            msg=(line.split('msg":"'))[1].split('"') [0]
            constantParts=level + msg
            data.append((constantParts))
```

Figure 4.8: Parsing

To be able to abstract the log entries, we used a built-in set() function to see how many unique log entries the log file had, and with that information, we could assign an id for every log type. Thereafter a sliding window was implemented for catching sequences of size five. Two simple methods were also created for assigning and calculating values for both the feature vector representation and IDF representation and the numerical sequences which were represented in a vector were written to a CVS file. Fig 4.9 illustrates the data from a CVS file, which has been constructed from a feature vector representation method with a sliding window of five. The rows in Fig 4.9 illustrate the vector values from the sequences and the column illustrates a specific log entry. In the TVS file, we had 29 different
log entries, which is the reason for the row length of 29.

Because we had 29 different kinds of log entries, every vector will have 29 dimensions. We implemented the PCA for reducing the dimensions from 29 to 2 for visualizing purposes and we used libraries from skikit learn, and the implementation for PCA can be seen in Fig 4.10.

A implementation was needed for calculating all distances between each data point and their centroid for the K-means with threshold approach. The reason for obtaining the distance was to see if the data point was within the threshold percentile or not. In Fig 4.11 we illustrate the method for calculating all the distances and how we compared it with the threshold percentile, which was done by using a build-in functions and euclidean distance. We first obtain the centroids coordinates, thereafter we loop through all the data points and calculate the
distance between the cluster centroid and the data point itself. The distance is calculated with cdist.

```python
# Obtaining the coods for each centroids
centroids = km.cluster_centers_
# points array will be used to reach the index easy
points = np.empty((0, len(fif2[0])), float)
# distances will be used to calculate outliers
distances = np.empty((0, len(fif2[0])), float)
# get points and distances
for i, centerElem in enumerate(centroids):
    # cdist is used to calculate the distance between center and other points
    distances = np.append(distances, cdist([centerElem], fif2[myclusters == i], 'euclidean'))
points = np.append(points, fif2[myclusters == i], axis=0)

percentile = 97.5
# getting outliers whose distances are greater than some percentile
outliers = points[np.where(distances > np.percentile(distances, percentile))]
normal = points[np.where(distances == np.percentile(distances, percentile))]
```

Figure 4.11: *Source code for calculating the distances for each data point to its centroid and the source code for comparing the distances to the threshold*

To run the machine learning algorithms K-means and DBSCAN, scikit learn libraries were used, and the parameters that were used in the algorithms were calculated as previously mentioned in the method.
5 Evaluation

The data file received from Mobilaris has 41294 log entries with 29 different kinds of log entries. Out of the 41294 there exist 1176 anomalous sequences with various sizes. In this thesis, we will use three different performance indicators, precision, recall, and F1-score. These indicators are considered as standard metrics for evaluating the test result[21]. Precision measures the proportion of the real anomalous log sequences in those predicted anomalous log sequences by the machine learning algorithm, meanwhile recall measures the proportion of the anomalous log sequences that are correctly predicted. A machine learning algorithm with a high recall infrequently mispredicts anomalous sequences as a normal log sequence. Machine learning algorithm with a high precision infrequently mispredicts a normal log sequence as an anomalous log sequence. F1-score is a metric that combines recall and precision and is used for better measurement. The following terminologies will be used in our standart metrics,

- **True positive (TP)**, will correspond to the number of anomalous log sequences which are correctly predicted. In our evaluation, if an anomalous log sequence is within a sliding window and the sliding window sequence is labeled as an anomaly by the machine learning algorithm, we will count it as true positive.

- **False negative(FN)**, will correspond to the number of anomalies wrongly predicted as normal. In our evaluation, if there is an anomalous sequence in the sliding window and the algorithm labels the window as normal, we will count it as a false negative.

- **False positive(FP)**, will correspond to the number of normal log sequences wrongly predicted as an anomaly. In our evaluation, if there is a normal sequence in the sliding window and the algorithm labels the window as an anomaly, we will count it as a false positive.

- **True negative (TN)**, will correspond to the number of normal log sequences correctly predicted as normal. In our evaluation, if there is a normal sequence in the sliding window and the algorithm labels the window
as a normal, we will count it as true negative.

The definition of the metrics are given below,

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{1}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{2}
\]

\[
F1 = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \tag{3}
\]

In this study, we will do two different evaluation tests with the three performance indicators. In the first one, we will count the total number of found anomalous log sequences, and in the second, we will count how many different kinds of anomalous log sequences we found. In the TVS file there exist a total of 1176 anomalous sequences, which has 174 different kinds of anomalous sequences that occur a different amount of times, which sum up to 1176. We will call the different kinds of sequences for the unique sequences. The reason for doing the two different evaluation tests is to get a better overview of the anomaly detection. Some log sequences might occur more often, than others, which can in some cases be harder to detect.

### 5.1 K-means with Feature Vector Representation and PCA

In this subsection, we will show the result from the K-means approach with feature vector representation. Table 5.1 illustrates the result from all anomalous log sequences and Table 5.2 illustrates only the unique ones found. The metrics recall, F1 and precision are shown and the three different thresholds were 99, 98, and 97.5. Fig 5.1, 5.2, and 5.3 shows the plots from the K-means algorithm with the thresholds used, PCA was also used for reducing the dimensions from 29 to 2. Each of the plots had three clusters and the blue star illustrates the centroid for that cluster. The red circles around the data points are the marked anomalies from the algorithm.
Table 5.1: Result from all log sequences, K-means Feature Vector Representation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>K-means with threshold percentile 99</th>
<th>K-means with threshold percentile 98</th>
<th>K-means with threshold percentile 97.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.34</td>
<td>0.61</td>
<td>0.87</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.51</td>
<td>0.75</td>
<td>0.93</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.2: Result from unique log sequences, K-means Feature Vector Representation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>K-means with threshold percentile 99</th>
<th>K-means with threshold percentile 98</th>
<th>K-means with threshold percentile 97.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.78</td>
<td>0.79</td>
<td>0.83</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.88</td>
<td>0.88</td>
<td>0.91</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.1: K-means with k=3 and threshold with percentile 99
Figure 5.2: K-means with $k=3$ and threshold with percentile 98

Figure 5.3: K-means with $k=3$ and threshold with percentile 97.5
5.2 K-means with IDF representation and PCA

In this subsection, we will show the result from the K-means approach with IDF data representation. Table 5.3 illustrates the result from all anomalous log sequences and Table 5.4 illustrates only the unique ones. The metrics recall, F1 and precision are shown and the three thresholds were 99, 98, and 97.5. Fig 5.4, 5.5, and 5.6 show the plots from the K-means algorithm with the threshold used, PCA was also used for reducing the dimensions from 29 to 2. Each of the plots had also three different clusters and the blue star illustrates the centroid for that cluster. The red circles around the data points are the predicted anomalies from the algorithm.

Table 5.3: Result from all log sequences, K-means, IDF Representation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>K-means with threshold percentile 99</th>
<th>K-means with threshold percentile 98</th>
<th>K-means with threshold percentile 97.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.31</td>
<td>0.31</td>
<td>0.87</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.48</td>
<td>0.48</td>
<td>0.93</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.4: Result from unique log sequences, K-means, IDF Representation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>K-means with threshold percentile 99</th>
<th>K-means with threshold percentile 98</th>
<th>K-means with threshold percentile 97.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.87</td>
<td>0.87</td>
<td>0.91</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.93</td>
<td>0.93</td>
<td>0.95</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Figure 5.4: *K*-means with $k=3$ and threshold with percentile 99

Figure 5.5: *TK*-means with $k=3$ and threshold with percentile 98
In this subsection, we will show the result from the DBSCAN approach with Feature vector representation. Table 5.5 illustrates the result from all anomalous log sequences and Table 5.6 illustrates only the unique ones. The metrics recall, F1, and precision are shown. Fig 5.7 shows the plot from the DBSCAN algorithm, PCA was also used for reducing the dimensions from 29 to 2. The DBSCAN algorithm created seven clusters. The purple data points are the predicted anomalous sequences from the DBSCAN.
Table 5.5: Result from all log sequences, DBSCAN, Feature Vector Representation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>DBSCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.43</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.60</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.6: Result from unique log sequences, DBSCAN, Feature Vector Representation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>DBSCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.90</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.95</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 5.7: DBSCAN feature vector representation
5.4 DBSCAN with IDF representation and PCA

In this subsection, we will show the result from the DBSCAN approach with IDF representation. Table 5.7 illustrates the result from all anomalous log sequences and Table 5.8 illustrates only the unique ones. The metrics recall, F1, and precision are shown. Fig 5.8 shows the plot from the DBSCAN algorithm, PCA was also used for reducing the dimensions from 29 to 2. The DBSCAN created six clusters. The purple data points are the predicted anomalous sequences from the DBSCAN.

Table 5.7: Result from all log sequences, DBSCAN, IDF Representation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>DBSCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.35</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.51</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.8: Result from unique log sequences, DBSCAN, IDF Representation

<table>
<thead>
<tr>
<th>Metrics</th>
<th>DBSCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>0.78</td>
</tr>
<tr>
<td>F1-score</td>
<td>0.87</td>
</tr>
<tr>
<td>Precision</td>
<td>1</td>
</tr>
</tbody>
</table>


5.5 Discussion and analysis

The aim of this thesis was to find anomalous sequences in their system with help of machine learning.

In approach K-means with feature vector representation, the model performed poorly in finding all anomalous sequences when the threshold percentile was 99. When lowering the threshold to 97.5 the recall and F1 score got higher, which can be seen in Table 5.1. F1 went from 0.51 to 0.93 and recall went from 0.34 to 0.87. In finding unique anomalous sequences the k-means approach with feature vector representation performed well with all percentiles and the F1 score and recall were high, F1 score went from 0.88 to 0.91 and recall went from 0.78 to 0.83. The algorithm also performed better in finding the unique sequences when the threshold percentile was 97.5.

In approach K-means with IDF representation, the model performed equally good as the k-means with feature vector representation in finding the total amount.
of anomalous log sequences with a threshold of 97.5. However, the algorithm model performed better in finding the unique sequences with the same threshold, which can be seen by the higher F1 score and recall. When the thresholds were 99 and 98, the feature vector representation had a higher F1 score and recall. The reason for better performance in finding unique sequences could be that the IDF representation is weighting the rareness of a log entry, which could be the reason for the better performance.

In approach DBSCAN with feature vector representation and IDF representation, both models performed poorly in finding the total amount of anomalous sequences, recall, and F1 score was low. F1 score for the DBSCAN with feature vector representation was 0.60 and recall was 0.43. F1 score for the DBSCAN with IDF representation was 0.51 and recall was 0.35. However, the algorithms performed well in finding the unique anomalous sequences.

From the result obtained, the algorithms K-means and DBSCAN performed better on finding the different kinds of anomalous log sequences than finding the total amount of them. As mentioned before in our data file we had a total of 1176 anomalous sequences of different sizes. Out of the 1176 anomalous sequences, there were 174 unique sequences, which occurred different amounts of times. The result suggests that our algorithms are good in detecting unique sequences, but might need more work in detecting the total amount. The reason for the suboptimal performance in finding the total amount of anomalous sequences is that some of the anomalous sequences appeared very frequently in the log file. Because of that, the machine learning algorithm labeled the sequences as normal.

In this thesis, we only used a sliding window of size five, which could be a reason for missing some of the anomalous sequences.

From the PCA cluster plots in the result, we can see which of the log sequences were labeled as an anomaly and which were labeled as normal. When observing the plots it seems that all log sequences were labeled as an anomaly, which is not correct. Most of the log sequences from the Mobilaris TVS service file were identical, which will make the majority of sequences plotted in the same area.

None of the algorithms labeled a normal log sequence appearing in a sliding
window as an anomaly, which is the reason for the perfect value on the precision metric. A high precision infrequently mispredicts a normal log sequence as an anomalous log sequence, which never occurred in our study. However, as mentioned before if an anomalous log sequence is within a sliding window and the sliding window sequence is labeled as an anomaly by the machine learning algorithm, we will count it as true positive. Meaning that if we have an anomalous sequence of size two, we will still have three other log entries in the window which might be considered as normal. It’s up to the end-user to determine which of the logs is an anomaly.

The K-means approach for identifying anomalies had some limitations and also the DBSCAN algorithm, the threshold that was used had to be set by the developer. The thresholds were designed to perform well for the Mobilaris TVS system, a different system will probably need to have different threshold values. The limitations with K-means and DBSCAN algorithms were that we had to manually set the parameters for the algorithms, which could affect the result.
6 Conclusions and future work

In this thesis, we investigated the possibilities machine learning can have for analyzing the log files at Mobilaris with anomaly detection in focus. From the result, we found that both the machine learning algorithms K-means and DBSCAN were good at finding unique log anomalies. The K-means approaches with low percentile threshold were the only algorithms, that performed well in finding the total number of log sequences. We also investigated how the data representation can impact the result, our investigation showed that the K-means approach that used IDF representation performed equally good in finding the total amount of anomalous log sequences with a threshold of 97 as the k-means with feature vector representation with the same threshold. However, the K-means with IDF representation performed better in finding the unique log sequences when the threshold percentile was 97.

The investigation gave us a good result, and the result could be useful for finding anomalous sequences in both finding the unique ones and the total ones. However, implementing this in real usages was a challenge due to all the manual work. The first step was to manually implement a parser for the log file, the second step was to parse the data into abstracted log messages, which was a challenge to generalize for the file.

One way to further improve the work of this thesis would be to test out different window sizes. In this thesis, only a size of five sliding window was used due to the time limit. Investigating in different window sizes could have an interesting outcome on the result. Due to the time limit, we focused on one service at Mobilaris instead of the whole system, a more generic solution could be applied, which would work on other types of services.
References


