Republic of Tunisia

Ministry of Higher Education and Scientific Research

University of SFAX

Higher Institute of Computer Science and Multimedia of SFAX

Department of Computer Science and Multimedia

End of Studies Internship Report

In pursuit of obtaining the Bachelor's degree in:

Computer Science

Branch: Big Data & Data Analytics

Log files analysis and features extraction to deduce the behaviour of logged systems

Introduced to

Higher Institute of Computer Science and Multimedia Of SFAX

Performed at

Primatec Engineering

By

Sirine Dhouib & Mariem Sfaxi

Academic year: 2021/2022

Dedications

I dedicate this humble work to:

My dear parents,

For whom I owe what I am today. Thank you so much for everything! You have been my source of inspiration, support, and guidance. You have taught me to be unique, and determined, to believe in myself, and always to persevere. Thanks to you, I learned the meaning of work and responsibility. I would like to thank you for your love, your generosity and for all the sacrifices you made for my education and studies .I am truly thankful and honoured to have you as my parents. Each of us has cause to think with deep gratitude of those who have lighted the flame within us. You, mom and dad, have been that spark for me when my light blew out. Thank you for your unwavering love and support along this journey I have taken. I love you both always and forever. Thanks to your prayers, support and sacrifices all along our curriculum. May God, the almighty preserve you and provide you with health and long life.

My deceased grandmother Rafia,

For all the affection she gave me and for her precious encouragement in every step of my life. Your prayers for success accompany me wherever I go. May God the Almighty grant you His infinite mercy and welcome you into His eternal Paradise.

My sister and brothers,

No dedication would be enough to express how I feel about you. Thank you for your everlasting love and warm encouragement throughout my End of Studies Project. Without you, I couldn't overcome my difficulties and concentrate on my studies. I will just say thank you for your advice and encouragement. You are the joy of my life May the Almighty God keep you and give you health, happiness and success.

My Best Friend Ousema,

I would like to thank you for standing by my side through thick and thin. Thank you for all your support throughout my journey. I will always appreciate all you have done for me, especially for helping me develop my technology skills, for the many hours of proofreading, and for helping me to master the leader dots. Thank you for always being there for me.You have been my best Cheerleader.

My family,

I would like to thank you for all the love and the support that you gave me. All words cannot express my affection and gratitude for your help and support, especially my aunt Fatma. May God give you a long and happy life.

All my teachers,

For their kindness, their advice, their encouragement and for their contribution to my solid training. Especially, Mr Mohamed Miladi. I would like to thank you for supporting me, guiding me and motivating me throughout all my studies and my End of Studies project.

My friends,

Thank you for offering me advice, and supporting me through this entire process.

All the people who have contributed in any way to the realization of this project,

Through their advice and motivations without Forgetting **My Partner Mariem.** I would like to thank you for your support and patience throughout our End of Studies Project. Thank you for giving all your best and also for having unity.

And lastly, I dedicate this piece of work to our **God** Almighty, My source of Inspiration, Wisdom, and Knowledge. Thank you, **God**, for giving me enough strength for this kind of End of Studies Project.

Sirine Dhouib

Dedications

I dedicate this humble work to:

My dear father,

My precious offer from God, in spite of the great responsibilities that you assume in your work as a father, you have always been close to me, to listen to me, to support me, to follow me and to encourage me. May this work lessen your suffering and bring you happiness.

My lovely mother,

The woman who suffered without letting me suffer, who never said no to my demands and who spared no effort to make me happy. You knew how to take care of me and make efforts for my education. No dedication can express all the respect and love you have for me, you have always trusted me. Please find in this work the consolation and the witness of patience.

My dear sister,

I reserve most of this work for you. You have always been a great help to me. I thank you for all the good you have done for me. May God protect you, give you luck, success and happiness.

Every member of my large family,

Thank you for your love and encouragement. May God give you a long and happy life.

All my teachers,

Who never ceased to advise, encourage and support me throughout my studies.

My friends,

May God preserve our friendship.

All the people who had the kindness and patience to satisfy my curiosity and to help me in my work by their precious advice, answers and recommendations.

Without forgetting **my partner Sirine** for her moral support, her patience, her effort and her understanding throughout this project.

Mariem Sfaxi

Acknowledgements

As a preamble to this End of Studies project, we thank God who has helped us and given us patience and courage during these years of study.

Also, it is with great honour and pleasure that we dedicate these few lines to thank all the people who encouraged us to accomplish this project.

To Mr. Tarek ZLITNI for the confidence he has given us by accepting to supervise our work, for his effort, his encouragement and his precious advice.

We address our thanks to all the personnel of the company and in particular

Mr. Firas FEKI, our supervisor at PRIMATEC ENGINEERING, for his availability, his support, his advice and the help he gave us during our internship.

We would also like to express our sincere thanks to the members of the jury, Mr Mohamed Tmar the President and Mrs Inès Kammoun Fourati member of the jury for the great honour of evaluating our work.

Finally, we would like to thank all our teachers of the Department of Computer Science and Multimedia of the Higher Institute of Computer Science and Multimedia of Sfax and all the people who encouraged us.

Abstract

As the server logs increase in size, it becomes difficult for human experts to manually examine error log messages and analyse the anomalies because of the high volume of log data. If the error message is rare or of low frequency, the system does not categorize it as essential and gets ignored which may lead to fatal errors. Server log analytics has proven to be optimum for active strategies and excellent performances of the system like preventive maintenance or complete shut-down. Improvements in analytical strategy are necessary for data analysts in handling a large system.

For this analytical process to yield good results, the input data need to be of good quality; therefore, research focuses on cleaning and pre-processing techniques. This research proposes consecutive logical steps to enhance the analysis of log messages. First, we purpose to extract sequences and patterns from the logs by optimizing window sizes without losing valuable information and combining them with forecasting techniques for predictive analytics. Second, we improve topic modelling for low-frequency messages through text analysis and language modelling.

The resulting proof of concept is not just visualizing the log data; instead, it provides insight into the logs through topics from the error messages. The experiments illustrate the effectiveness of the proposed steps and the approach for error log analysis.

Table of Contents

Abstract i

List of Figures v

List of Tables vi

General Introduction 1

Chapter 1: General Project Context 3

I. Introduction: 4

II. General Context of Work: 4

1. Host Company: 4

2. About Log Files: 5

3. Problematic and motivation: 5

3.1 Problem Definition: 6

3.2 Current State of Technology : 7

3.3 Current Practice : 8

3.4 Research Methodologies : 8

3.5 Data Acquisition and Pre-processing: 9

3.6 Generating the sequence of events: 9

3.7 Error language Model Techniques: 9

3.8 Research contributions and its significance: 10

Chapter 2: State of the art 11

I. Introduction: 12

II. Machine Learning: 12

1. Machine Learning For Anomaly Detection: 12

2. Machine Learning Algorithms: 13

3. Supervised Machine Learning: 13

3.1 Logistic Regression (LR) : 14

3.2 K-Nearest Neighbours (K-NN) : 15

3.3 Naive Bayes (NB): 16

3.4 Decision Tree (DT): 17

3.5 AdaBoost Algorithm: 18

4. Unsupervised Machine Learning: 19

4.1 K-Means: 20

5. Neural Networks and Deep Learning: 21

6. Natural Language Processing (NLP): 22
Chapter 3: Proposed Solution 25
I. Introduction: 26
II. The process of log analysis: 27
1. Data Log collection: 27
2. Log parsing: 27
3. Features extraction: 28
4. Pre-processing: 29
4.1 Natural Language Processing: 30
4.2 Data Splitting: 31
4.3 Machine Learning Structure and Models: 32
4.3.1 Logistic Regression Model: 32
4.3.2 K-Nearest Neighbor Model: 32
4.3.3 Naive Bayes Model: 32
4.3.4 Decision Tree Classification Model: 32
4.3.5 AdaBoost Classification Model: 32
4.4 Log Analysis Tools: 33
4.4.1 Python: 33
4.4.2 Anaconda Navigator: 33
4.4.3 Jupyter Notebook: 33
4.4.4 Flask: 33
4.4.5 Sklearn: 34
4.4.6 Gensim: 34
4.4.7 Natural Language Processing (NLP) : 34
4.4.8 Pandas: 34
4.4.9 Keras: 34
4.4.10 Matplotlib: 34
4.4.11 TensorFlow: 35
4.4.12 Bootstrap: 35
Chapter 4: Experimental Results and Discussion 36
I. Methodology: 37
II. Experimental Results: 37
1. Unsupervised Learning Results: 37
1.1 Word2vec: 37
2. Supervised Learning Results: 38

2.1 Naives Bayes: 38
2.2 KNN: 38
2.3 Logistic Regression: 39
2.4 Decision Trees: 40
....41
2.5 AdaBoost: 42
3. Machine Learning Metrics and Performance:....42
3.1 Accuracy:....43
3.2 Precision: 43
3.3 Recall:....43
3.4 F1 Score:....43
4. Comparison of Performance of Machine Learning Models: 43
4.1 Results of Machine Learning Performance and Discussion :....44
4.2 Conclusion :....46
III. Conclusion and Future Work:....46
General Conclusion....47
References....49

List of Figures

FIGURE 1: HOST COMPANY 4
FIGURE 2:GENERATING THE SEQUENCE OF EVENTS 9
FIGURE 3:ERROR LANGUAGE MODEL TECHNIQUES 10
FIGURE 4:EXAMPLE OF ANOMALY 13
FIGURE 5:FUNCTION OF LOGISTIC REGRESSION 15
FIGURE 6:EXAMPLE OF DECISION TREE 18
FIGURE 7:EXAMPLE OF ADABOOST 18
FIGURE 8:ADABOOST ALGORITHM 19
FIGURE 9:EXAMPLE OF FORMATION OF CLUSTERS 20
FIGURE 10:ARTIFICIAL NEURAL NETWORK 21
FIGURE 11:METHOD FOR FINDING ANOMALIES 26
FIGURE 12:PYTHON LOGO 33
FIGURE 13:ANACONDA LOGO 33
FIGURE 14:JUPYTER NOTEBOOK 33
FIGURE 15:FLASK LOGO 33
FIGURE 16:SKLEARN LOGO 34
FIGURE 17:GENSIM LOGO 34
FIGURE 18:NLP WITH PYTHON 34
FIGURE 19:PANDAS LIBRARY 34
FIGURE 20:KERAS LIBRARY 34
FIGURE 21:MATPLOTLIB LIBRARY 34
FIGURE 22:TENSORFLOW LOGO 35
FIGURE 23:BOOTSTRAP LOGO 35
FIGURE 24:WORD2VEC MODEL RESULT 37
FIGURE 25:NAÏVE BAYES MODEL RESULT 38
FIGURE 26:KNN MODEL RESULT 39
FIGURE 27:LOGISTIC REGRESSION MODEL RESULT 40
FIGURE 28:DECISION TREE MODEL RESULT 41
FIGURE 29:ADABOOST MODEL RESULT 42
FIGURE 30:CONFUSION MATRIX 43
FIGURE 31:MODELS PERFORMANCE 45
FIGURE 32:RECALL RESULTS OF MACHINE LEARNING MODELS ON DATASET 45

List of Tables

TABLE 1:FEATURES EXTRACTION.....28
TABLE 2:PERFORMANCE RESULTS OF MACHINE LEARNING MODELS ON DATASET.....45

General Introduction

Large systems generate an immense amount of log data each day, containing various sensitive and invaluable information. Data analysts use this information to improve the user's experience, including the system's security and confidentiality.

As the number of data doubles every day, it is becoming more challenging for data analysts to track errors on different servers and also drain the resources. The system analysts commonly use the grep command on the error log messages to find out the server failures or application failures, and in several cases, if the error message is irrelevant, it is difficult to figure out what to search for. The keyword as “FAILURE”, and “ERROR” and “FATAL” are not helpful for data analysts as these error messages may occur during system maintenance or actual application server failure and the terms are interchangeable. Error messages do not occur in a similar pattern and sometimes go through different stages and accumulate extra information, which makes analysing log data more challenging.

Finding the issues through error logs is a vital part of the server log analytics, and it takes a significant amount of time to search for the issue if the analysts do not have complete knowledge of the system; therefore, automated monitoring tools are needed to continuously and effectively monitor the server’s behaviour. There are some off-the-shelf apps available in the market for log analysis. However, there are some benefits and drawbacks to having them. Some are expensive and some are purely based on numerical data. After having a discussion with stakeholders, we found that all the software’s not aligned with the user requirements. The user wants to build a house system because the off-the-shelf cannot be applied to the stakeholder's Tibco log system. They mentioned that there is always an issue in analysing error messages, which became the motivation for this research. Whenever any error happens on the server, the analyst has to query the log files based on their experience however if the analyst has no experience or is new to the company. It would be hard to know what to search for and where to look for information. For example, ‘connection termination’ can happen for several reasons, if the analysis search for an error or a fatal error. The system will bring all the information logs and error logs. Which will take a significant amount of time.

The chapters of this project are organized as follows in five chapters, including the introduction.

Chapter 1 is related to the Introduction and General Project Context, and it has all the necessary information about the research work done in this End of Studies Project. Introduction includes the brief explanation of the server logs issues, log messages analytics system, and the current analytics system.

Chapter 2 is related to state of the art and gives essential information about background knowledge. It also walks us through the literature concerning relevant topics and provides comprehensive knowledge and understanding of the concepts, technologies, and algorithms that support this project. This chapter also talks about log analytics techniques, sequence mining techniques models, data mining, predictive analytics, and prescriptive analytics.

Chapter 3 discusses the proposed solution and method of development and describes the methodologies used during the development of this End of Studies Project. This chapter describes the contributions of research and all the steps for server analytics are described in detail. We have described all the algorithms used in sequence mining, text analysis, predicting the next event and log clustering technique.

Chapter 4 presents the experimental setup and discuss the derived results. The first section describes the data set used to perform the experiments, then the cleaning and pre-processing of server logs and sequence mining results, which also helps to reduce the time and efforts. Next section describes the results of predicting the next event and comparing the results with three predicting models. This chapter includes the language model for low frequency data for topic modelling and discussed the constraints of the experiments and the Future Work.

Chapter 1: General Project Context

I. Introduction:

The first chapter serves to put the work in the general project context, then, we first propose to present the host company "PRIMATEC ENGINEERING", then we expose the problematic, the need that gave birth to this project and to the work. This allows us to better understand the subject and to define its context.

II. General Context of Work:

This work is an End of Studies project, was carried out within the company PRIMATEC ENGINEERING as part of our training at the **Higher Institute of Computer Science and Multimedia of Sfax** during the **academic year 2021-2022** in the second year of a bachelor's degree in Computer Science.

1. Host Company:

To meet the challenges of an increasingly complex automotive context, **Primatec Engineering** relieves its customers to enroll end to end ECU testing services, from test environment development, test automation to issuing complete test reports and reviews.

Driven by **AI and ML technologies**, its intelligent testing solutions and automation service is comprehensive and scalable, delivered in a flexible business model that enables clients improve the product quality, accelerate service delivery, and increase our customers competitiveness.

Collective intelligence consists of combining the diversity of expertise of its teams and making a good synergy with its customers and partner's teams thriving efficiency and excellence. Agility allows **Primatec** to capitalize on its experiences and reinvent their selves with each mission with pragmatism and humility.

Primatec Engineering is continuously working on quality processes and apply them to its rules and activities to meet major certifications and standards compliance required by the automotive industry.

Figure 1: Host Company

2. About Log Files:

Current software applications often produce (or can be configured to produce) some auxiliary text files are known as log files. Such files are used during various stages of software development, mainly for debugging and profiling purposes.

The use of log files helps to test by making debugging easier. It allows following the logic of the program, at a high level, without having to run it in debug mode.

Nowadays, log files are also commonly used at customer installations for permanent software monitoring and/or ne-tuning. Log files have become a standard part of large applications and are essential in operating systems, computer networks, and distributed systems.

Log files are often the only way how to identify and locate an error in software because log le analysis is not affected by any time-based issues known as project. This is the opposite of an analysis of a running program when the analytical process can interfere with time {critical or resource {critical conditions within the analysed program.

Log files are often huge and can have a complex structure.

Although the process of generating log files is quite simple and straightforward, log le analysis could be a tremendous task that requires enormous computational resources, a long time, and sophisticated procedures. This often leads to a common situation, when log files are continuously generated and occupy valuable space on storage devices, but nobody uses them and utilizes enclosed information.

3. Problematic and motivation:

1. As the servers generate thousands of logs each day, it takes a significant amount of time and effort to review the server logs for log analytics and data mining. The logs' simple keyword search leads to outliers and false positives such as “Connection terminated” error is generated due to several reasons, and this error can propagate to another system.

In such types of cases, there is no well-defined mapping between log messages and observed behaviour. And it is very difficult to decide what and where to look for information. This research study targets these issues by analysing error logs through natural language processing.

2. The single type of alert or error generates different types of logs, including automatic hexadecimal numbers, timestamps, and plenty of text that takes a lot of space on disks;

therefore, the resources drained out extremely fast. This study handles the large data set, and performs data cleansing; and pre-processing, which become a vital part of log analytics.

3. Sometimes the experts may overlook infrequent and critical logs, leading to significant issues in the system. The infrequent errors/ unseen messages motivate this research for Text mining, Topic modelling, and natural language mining techniques.

4. Unrelated errors occur in the server logs due to other systems which affect the prediction of the next event, which motivates analysts for pattern mining. Information on the specific issues that happen at certain times can be extracted from historical error data, to help to understand the system's behaviour. These issues motivate this study to find the sequence of events from the server logs and forecast the next event by machine learning algorithms.

5. Traditionally, several experts analyse the error logs and find the solutions according to their expertise and knowledge database, and sometimes solving one issue, may lead to another issue on a related server. This issue occurs when the analysts do not have complete knowledge of the system. This research concentrates on visualization server logs to solve this issue.

In our project, we have concentrated on the server issue mentioned above, and the log analytics process for server logs is widely divided into the following main steps, the first step includes server log data cleaning and data pre-processing, and the second step uses pre-processed data to find some patterns in the historical data and extract hidden information, the fourth step is to create error language for error messages, and clustering error messages and next step includes analysing the extracted information and results.

3.1 Problem Definition:

The overall goal of this research is to invent and design a good model of the generic processing of log files. The problem covers areas of formal languages and grammars, finite state machines, lexical and syntax analysis, data-driven programming techniques, and data mining/warehousing techniques.

The following list sums up the areas involved in this research.

- Formal definition of a log file
- Formal description of the structure and syntax of a log file (metadata)
- Lexical and syntax analysis of log file and metadata information

- Formal specification of a programming language for easy and efficient log analysis
- Design of internal data types and structures
- Design of such programming language
- Design of a basic library/API and functions or operators for easy handling of logs within the programming language
- Deployment of data mining/warehousing techniques if applicable
- Design of a user interface

The expected results are both theoretical both practical. The main theoretical results are the design of the framework and the formal language for log analysis description.

3.2 Current State of Technology :

In the past decades, there was surprisingly low attention paid to the problem of getting useful information from log files. It seems there are two main streams of research.

The first one concentrates on validating program runs by checking the conformity of log files to a state machine. Records in the log files are interpreted as transitions of the given state machine. If some illegal transitions occur, then there is certainly a problem, either in the software under test or in the state machine specification, or in the testing software itself.

The second branch of research is represented by articles that just describe various ways of producing statistical output.

The following items summarize the current possible usage of log files:

- generic program debugging and profiling
- tests whether the program conforms to a given state machine
- Various usage statistics, top ten, etc.
- security monitoring

According to available scientific papers, it seems that the most evolving and developed area of log file analysis is the WWW industry. Log files of HTTP servers are nowadays used not only for system load statistics but they offer a very valuable and cheap source of feedback. Providers of web content were the first ones who lack more detailed and sophisticated reports based on server logs. They require to detect behavioural patterns, paths, trends, etc. Simple statistical methods do not satisfy these needs so an advanced approach must be used.

Some log files, especially small and simple, can be also analysed using common spreadsheet or database programs. In such a case, the logs are imported into a worksheet or database and then analysed using available functions and tools.

3.3 Current Practice :

Prior to a more formal definition, let us simply describe log files and their usage. Typically, log files are used by programs in the following way:

- The log file is an auxiliary output file, distinct from other outputs of the program. Almost all log files are plain text files.
- On the startup of the program, the log file is either empty or contains whatever was left from previous runs of the program.
- During the program operation, lines (or groups of lines) are gradually appended to the log file, never deleting or changing any previously stored information.
- Each record (i.e. a line or a group of lines) in a log file is caused by a given event in the program, like user interaction, function call, input or output procedure, etc.
- Records in log files are often parametrized, i.e. they show current values of variables, return values of function calls, or any other state information
- The information reported in log files is the information that programmers consider important or useful for program monitoring and/or locating faults.

The syntax and format of log files can vary a lot. There are very brief log files that contain just sets of numbers and there are log files containing whole essays. Nevertheless, an average log file is a compromise of these two approaches; it contains minimum unnecessary information while it is still easily human-readable. Such files for example contain both variable names both variable values, comprehensive delimiters, smart text formatting, hint keywords, comments, etc.

Records in log files are often provided by time stamps. Although it is not a strict rule, log files without timestamps are rarely seen, because time-less events are of less valuable information.

3.4 Research Methodologies :

This research focuses on developing a proof of concept for the server log analytics through text analysis and machine learning algorithms. Though there is a fair amount of research that has been done on log analytics, log analysis is still under research and off the shelf applications do

not work for all types of log data. In this research, the following steps has been proposed for log data analysis along with topic modelling and language modelling.

3.5 Data Acquisition and Pre-processing:

The first and foremost steps include data acquisition, data cleaning, and pre-processing error logs. The idea of converting log files to binary data improves the time of execution and reduces the loading time. The data cleaning and pre-processing include removing duplicates, symbols, and handling system security information as the quality of results based on the quality of input data.

3.6 Generating the sequence of events:

In this step, the sequence has been extracted from pre-processed data.

Figure 2 describes that the log data acquired from the server database converts the data into binary form for faster execution. Rolling window size has been used to slice the sequence and find the patterns.

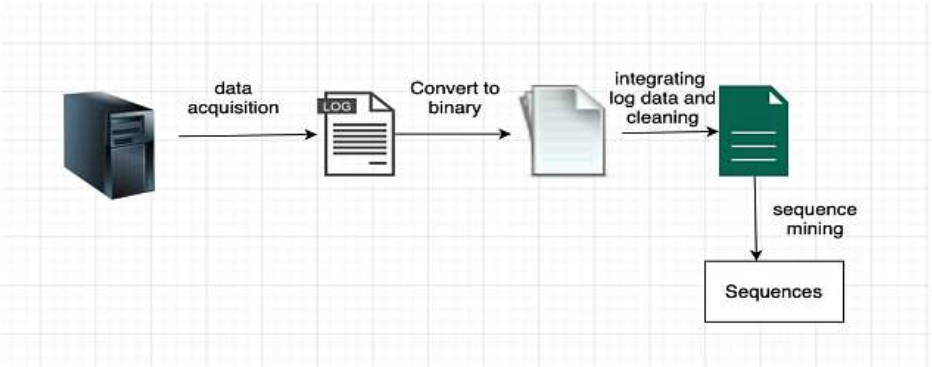


Figure 2: Generating the sequence of events

Figure 2 shows the data cleaning, pre-processing and sequences extraction from the error logs. The data acquisition, cleaning and pre-processing takes significant amount of time because the results of log analysis have direct correlation with quality of data.

3.7 Error language Model Techniques:

Error language model has been used for server logs and concentrated on low frequency data/ unseen data in the error messages.

Figure 3 describes the training log data set fed to the interpolation model with varying lambda and validate unseen data by held-out data and test the results with the testing set. We validate the results with the add-1 smoothing model, and linear interpolation outperforms.

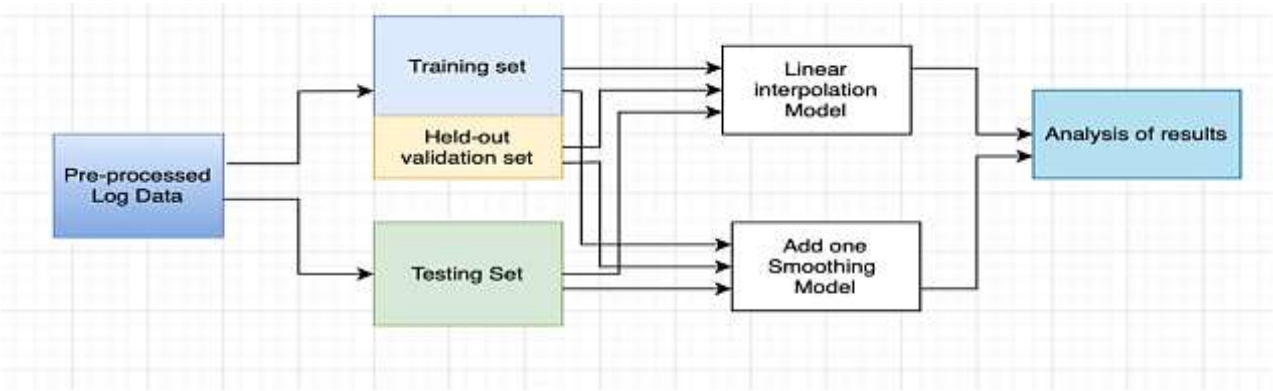


Figure 3: Error Language Model Techniques

Figure 3 shows the Error language model for low frequency log data analytics. The linear interpolation has been improved by varying the lambda. The results are compared with add one smoothing technique.

3.8 Research contributions and its significance:

This research has the following contributions towards the log analytics:

1. The consecutive logical steps for server log analysis to reduce the time and efforts for server's error log analytics, which improve the productivity of the analysts

2. Assigning Event ID - The proposed algorithm generate the Event Ids based on error message, which helps to remove the duplicate server log messages.

3. The methodology to extract low frequency/ unseen data from the server logs through linear interpolation technique in language model by varying lambda. This methodology provides the insight to low frequency data which is commonly get ignored.

4. The methodology to extract topics from error log messages and improving the visualization by varying K value. Combining trigram language model to LDA to improve topic visualization.

Chapter 2: State of the art

I. Introduction:

This chapter is the review of log analytics methodology from data acquisition to mining the results from logs, to gain the system's insight. The growing number of organizations generate millions of log entries from the application server to system servers that generate the company's business value, and log data has become important digital assets that generate the company's business value. It has always been a core task for business analytics and, if a data analyst ignores the log data or does not detect the abnormalities in the log files, it can lead to a breach in the system security, failure of the system, or downtime in production, which can impact the whole business. To ease up the management of exponentially growing log data sets and for log analytics, many research papers have been published with different methodologies.

II. Machine Learning:

Machine learning is a branch of Artificial Intelligence for data analysis that gives computers the ability to learn and improve from experience.

Machine learning algorithms are built on mathematical models, that search for patterns and relations in given data. These models are then capable of making decisions and predictions, even though they were not explicitly programmed.

Machine learning is widely used across different fields - from weather forecasting, finances, computer sciences, and image and language processing to tumour detection in biology. Utilization options for machine learning are almost endless and can be used in practically every aspect of our lives.

1. Machine Learning For Anomaly Detection:

Anomalies are events or items that differ from standard behaviour. They are rare and do not fit into normal patterns. Anomalies can indicate whether there is some kind of problem, for example, finance fraud, medical complication, or cyber-attack. Anomaly detection is a task to identify such events. Figure 1 a) illustrates an example of a point not fitting into the cluster and figure 1 b) illustrates an example of a spike in a time-series data.

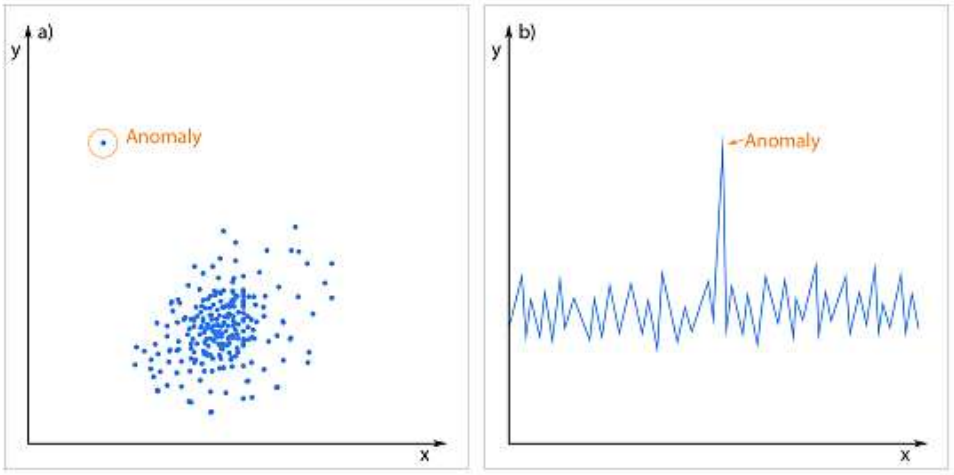


Figure 4: Example of Anomaly

There are dozens of machine learning algorithms. Each algorithm has its own specifications, due to which some algorithms are better at solving certain problems than others and vice versa.

This project deals with the implementation of these algorithms and compares them to others.

2. Machine Learning Algorithms:

In this study, a range of different machine learning algorithms unsupervised, supervised, and deep learning are used to analyse and interpret the channel outcrop statistics from our Dataset. These algorithms are described in further detail below.

3. Supervised Machine Learning:

The majority of existing machine learning algorithms are using supervised learning.

Supervised learning algorithms are developing machine learning models from labelled data. In other words, the process of learning from labelled data can be viewed as a teacher supervising the learning process. The learning process is iterative - it runs repeatedly until the algorithm accomplishes a sufficient level of performance.

The learning algorithm is mapping input variables (X) and an output variable (y) to find patterns. However, the goal is to create a mapping function that can predict the output variable as precisely as possible.

$$y = f(X) \quad (1.1)$$

Based on the problems the algorithm is dealing with, supervised learning can be divided into two groups - Regression and Classification.

Regression

Regression is a supervised learning algorithm used to predict continuous responses. For example, it can be used in finding the relationship between the price of a flat and its location, number of bedrooms, etc. Regression is one of the earliest machine learning algorithms and is still widely used. There are multiple types of regression methods, such as Simple Linear Regression (SLR), Multiple Linear Regression (MLR), Polynomial Regression (PR), Support Vector Regression (SVR), Decision Tree Regression (DTR), Random Forrest Regression (RFR), etc.

Due to the character of the data that this project deals with, none of the regression methods are used.

Classification

Classification is a supervised learning algorithm, which is grouping a given set of data points into classes. Sometimes classes are called also categories, labels, or targets.

A basic example of classification is spam detection in emails. A classifier trains to understand how given datasets are associated with labels spam and non-spam.

Spam detection falls under binary classification, as there are only two labels (classes). To deal with problems with multiple classes, such as whether an image is a dog, a cat, or a mouse, there are Multi-class classifiers.

There are multiple types of classification methods, such as Logistic Regression (LR), K-Nearest Neighbours (K-NN), Support Vector Machine (SVM), Kernel SVM (kSVM), Naive Bayes NB), Decision Tree Classification (DTC), Random Forrest Classification (RFC), etc.

3.1 Logistic Regression (LR) :

Logistic Regression (LR) is extending the Linear Regression model to be used for classification problems. Logistic Regression is not predicting exact number values (like 0 or 1), but it is using the logistic sigmoid function to generate a probability value - for example, a value between 0 and 1.

To compress output between 0 and 1, the function of logistic regression is defined as follows:

$$logistic(n) = \frac{1}{1 + \exp(-n)}$$

A visual representation of the Logistic function is shown in figure 5.

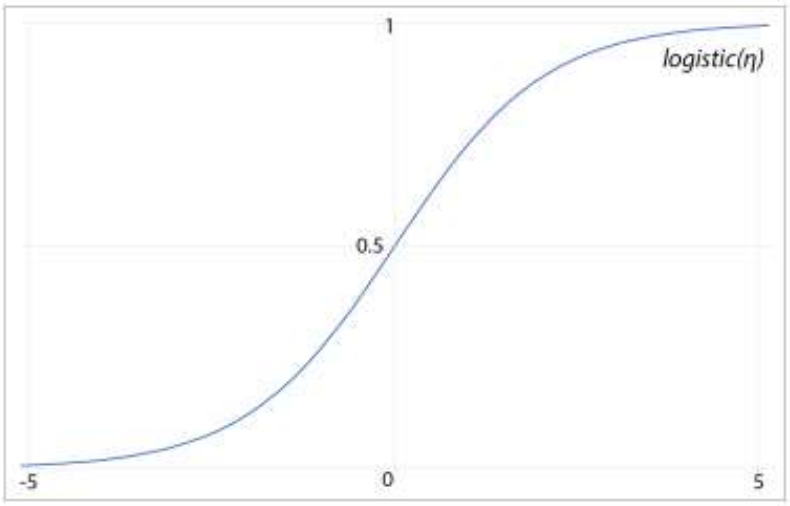


Figure 5: Function of Logistic Regression

As mentioned above, logistic regression is extending linear regression. The linear regression model is used for regression problems and is defined by the following equation:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + ... + \beta_p x_p^{(i)}.$$

As logistic regression is compressing outputs between values 0 and 1, function of logistic regression model is defined as:

$$P(\hat{y}^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + ... + \beta_p x_p^{(i)}))}.$$

Probability *P* will be a value between 0 and 1. As the output of the model needs to be either 0 or 1, thresholds have to be set up. Usually, the threshold is set up to be in the middle of values as follows:

$$p \geq 0.5, class = 1 p < 0.5, class = 0.$$

In case threshold is set up as above and prediction model returns value 0.67, it will be classified as 1. If returned value is for example 0.48, it will be classified as 0.
Implementation of Logistic Regression model in Python is described in the Section Chapter 4 Section 2.3.

3.2 K-Nearest Neighbours (K-NN) :

K-Nearest Neighbours (KNN) is simple, yet powerful classification algorithm. However, it can be used for regression problems as well. The KNN algorithm is based on assumption that similar data points lay close to each other.

The decision to which class should a node be assigned is based on the majority of votes of its neighbours. The number of neighbours depends on chosen value of parameter K. Nearest neighbours are then calculated by one of the following distance functions:

Euclidean function:

$$D(x,y) = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}.$$

Manhattan function:

$$D(x,y) = \sum_{i=1}^{k}|x_i - y_i|.$$

Minkowski function:

$$D(x,y) = (\sum_{i=1}^{k}(|x_i - y_i|)^q)^{1/q}.$$

However, these functions are accurate only for continuous variables. For classification problems is more suitable to use Hamming distance function:

$$D_H = \sum_{i=1}^{k}|x_i - y_i|.$$

$$x = y \Rightarrow D = 0.$$

$$x \neq y \Rightarrow D = 1.$$

There is no specified method to choose the best value of K, but the following recommendations should be taken into consideration:

- Large value of K means a more precise model.
- Choose an odd value of K for a problem with two classes - if K = 2 for such a problem, there might be a tie of what should be a class of the node.
- Multiple of a number of classes should not be chosen.

K = 1 is a special case of KNN when a node is assigned to a class of its closest neighbour.

Implementation of the K- Nearest Neighbours model in Python is described in Chapter 4 Section 2.2.

3.3 Naive Bayes (NB):

Naive Bayes (NB) is an algorithm calculating the probability of class for given data based on learned knowledge.

Bayes' Theorem

Bayes' Theorem is a mathematical formula describing conditional probability - probability of an event based on knowledge of other related conditions. For example, an internet search for "movie with a yellow car" brings up "Transformers". How the search engine knows this? Has it watched the movie? No it hasn't, but it learned it from similar searches of other people who probably were looking for it. Search e $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$, bability using Bayes' Theorem. Formula of Bayes' Theorem is define

Where A represents hypothesis and B represents observed evidence. $P(A|B)$ is called Posterior Probability, $P(B)$ is called Prior Probability and $\frac{P(B|A)}{P(B)}$ is called Likelihood Ratio.

Naive Assumption

Bayes' Theorem applied to the dataset will be defined as follows:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}.$$

A is replaced by class variable y and B is replaced by n-sized vector of features X, defined as:

$$X = (x_1, x_2, ...x_n).$$

Following on this, formula will be defined as:

$$P(y|x_1, ...x_n) = \frac{P(x_1|y)...P(x_n|y)P(y)}{P(x_1)...P(x_n)}.$$

After few adjustments, equation used by classifier will look like this:

$$\hat{y} = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y).$$

Implementation of Naive Bayes model in Python is described in Chapter 4 Section 2.3.

3.4 Decision Tree (DT):

The Decision Tree (DT) is a simple algorithm with a tree-like structure, where attributes of a dataset are represented by internal nodes of a tree, decision rules are represented by branched and outcome values are represented by leaves. Internal nodes are called Decision Nodes and are used to make decisions. They can have multiple branches going out of them. Leaves are called Leaf Nodes and are representing final output values. They do not have any branches but the one incoming towards them from the associated decision node. DT algorithm can be used for either regression or classification problems. They can be called Decision Tree Regression

(DTR) and Decision Tree Classification (DTC) algorithms respectively. As it can be seen in figure 5, the decision tree starts with the root node, expands into further branches, and forms a tree-like structure. Each branch represents a decision rule which can have two or more outcomes, for example, Yes/No, Red/Green/Blue, or numeric data as well. Implementation of the Decision Tree Classification model in Python is described in Chapter 4 Section 2.4.

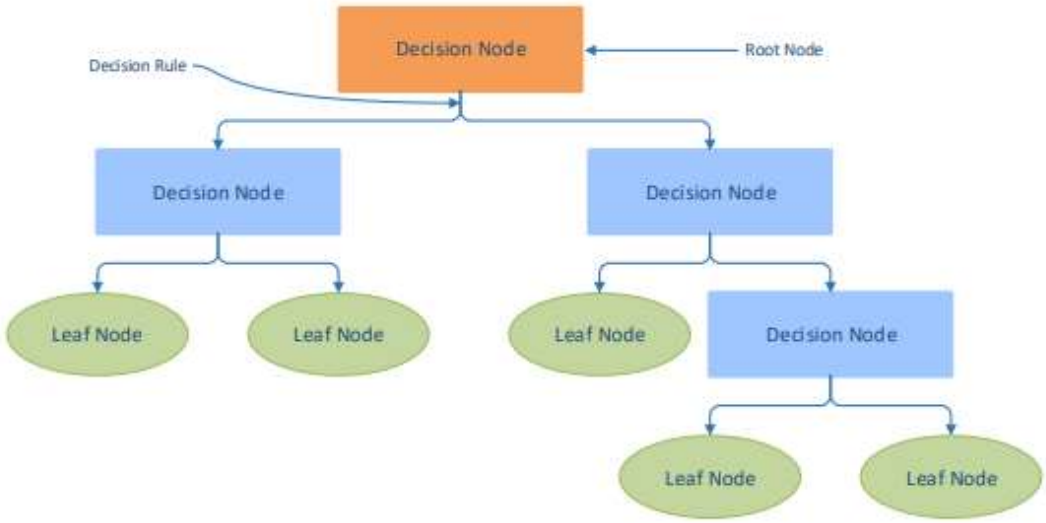


Figure 6: Example of Decision Tree

3.5 AdaBoost Algorithm:

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called Decision Stumps.

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level which means Decision trees with only 1 split. These trees are also called Decision Stumps.

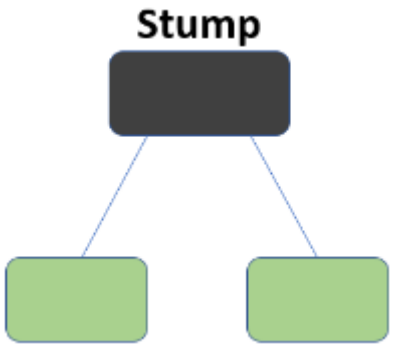


Figure 7: Example of AdaBoost

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.

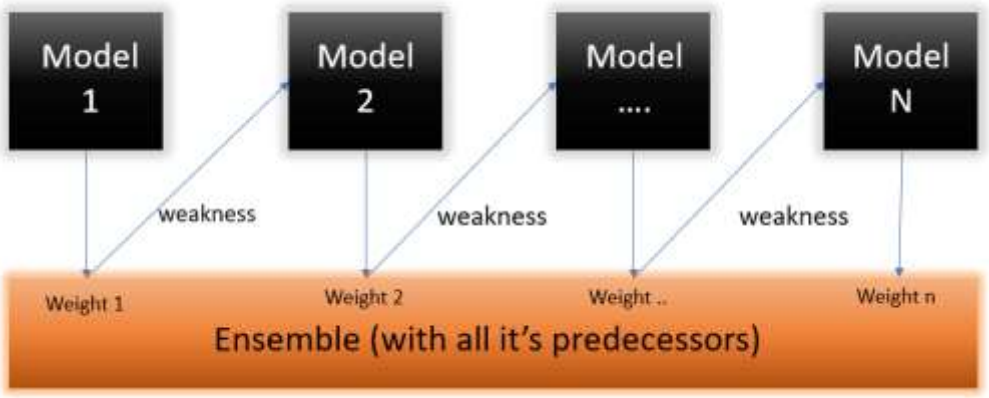


Figure 8: AdaBoost Algorithm

Implementation of the AdaBoost Classification model in Python is described in Chapter 4 Section 2.5.

4. Unsupervised Machine Learning:

In some cases, the training dataset is containing only input variables (X) and no output variables (y). In such cases, the goal might be to find certain patterns. To do so, unsupervised machine learning algorithms can be used. These algorithms can be used to find groups (clusters) based on similarities. Unsupervised machine learning algorithms are also often used for anomaly detection purposes. They can "group" normal behaviour data into one group and mark all data not fitting into this group as anomalies.

Unsupervised learning problems can be grouped into the following groups:

- **Association mining** is identifying associations in data points, for example, people that buy A might buy also B.
- **Clustering** is grouping data points based on similarities.

Association Mining

Association mining is algorithm that is searching and uncovering relations between input variables. These relations are formed into so-called association rules, which are then used to make predictions by calculating probabilities. Example of association mining is Priory algorithm.

Clustering

Clustering is the most common unsupervised learning method. It is searching for similarities in uncategorized datasets and grouping (clustering) them based on found patterns. Clustering algorithms are simple and quite effective. The number of clusters can be modified. It can help to improve the granularity of created clusters. Figure 9 illustrates a simple example of how certain data points can be formed into clusters.

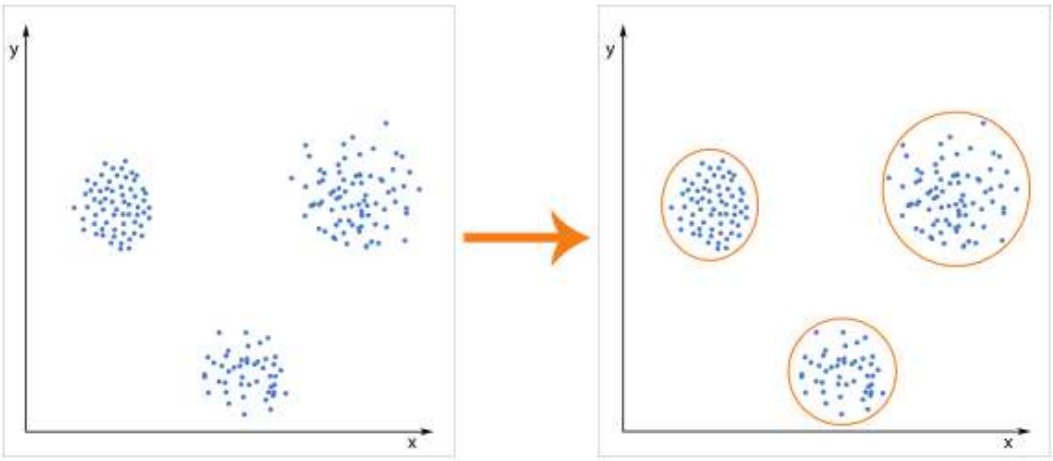


Figure 9: Example of formation of clusters

Most common clustering algorithms are K-Means clustering, Hierarchical clustering, Gaussian mixture models, Self-organizing maps, and Hidden Markov Models.

4.1 K-Means:

K-Means is a simple and one of the most popular unsupervised learning algorithms. It is creating groups (clusters) around centroids, which are basically defining the cluster. Centroids are also called means as they basically hold mean values of data points inside the cluster. Data points are assigned to clusters based on the distance between the data point and the centroid. It is assigned to a cluster of the closest centroid. Each data point can be part of only one cluster. The process of how the K-Means algorithm works is pretty straightforward.

1. Number of clusters K is specified manually.
2. K data points are randomly selected as centroids.
3. All data points are assigned to the closest centroid.
4. Centroids of newly created clusters are recalculated (centroids now might be "virtual" - not actual data points but the real centre of the cluster).
5. Steps 3 and 4 are repeated until one of the following criteria are matched:
    - Centroids are not changed

- Data points remain in the same clusters.
- Algorithm reaches the maximum number of iterations.

5. Neural Networks and Deep Learning:

An artificial Neural Network is a branch of machine learning modelled to mimic the network of neurons in a brain. On other hand, deep learning is an advanced subfield of machine learning that uses algorithms inspired by the structure and function of the Artificial Neural Networks. Deep learning is trained by using large sets of labelled data that learn features directly from the data without the need for manual feature extraction.

Common deep learning algorithms include convolutional neural networks and recurrent neural networks. Deep learning models are often referred to as deep neural networks. The term "deep" usually refers to the number of hidden layers in the neural network. Traditional neural networks only contain two to three hidden layers, while deep networks can have many more.

Neural Network algorithms are constructed with the following connected layers:

- **Input Layer:** this layer accepts all provided inputs.
- **Hidden Layer:** this layer is between the input and the output layers where computations are performed. In the case of deep learning then it means the network joins neurons in more than two to three layers.
- **Output Layer:** output is delivered via this layer.

The most basic unit of a Neural Network is a Perceptron. A Perceptron is a single layer Neural Network that is used to classify linear data. It has 4 important components:

1. **Inputs.**
2. **Weights and Bias.**
3. **Summation Function.**
4. **Activation or transformation Function.**

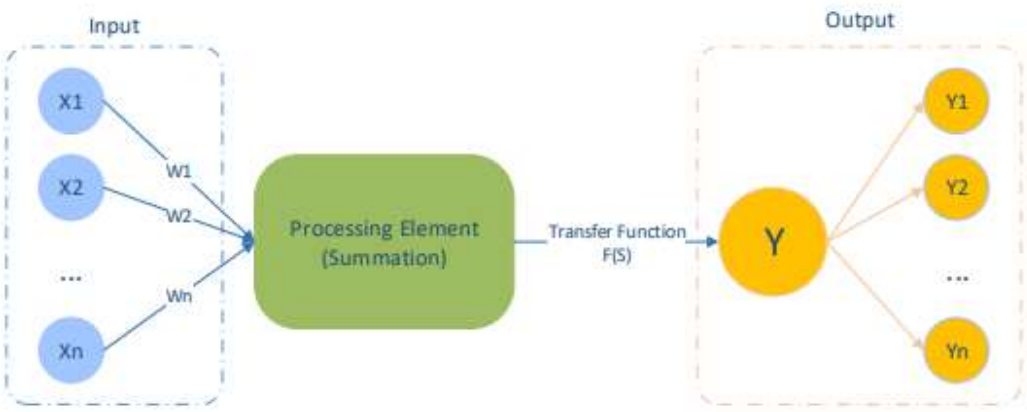


Figure 10: Artificial Neural Network

The inputs X received from the input layer are multiplied with their randomly chosen assigned weights w. While the weights determine the slope of the classifier line, bias allows to shift of the line toward left or right. Normally, bias is treated as another weighted input with the input value $x0$. However, the multiplied values are then added to form the Weighted Sum. The weighted sum of the inputs and their respective weights are then applied to a relevant Activation Function. The activation function maps the input to the respective output. This process is illustrated in figure 10.

The output of a neuron can be expressed mathematically as:

$$y = Activation(SUM(input * weight) + bias).$$

The activation function does a nonlinear transformation of the input data and thus enables the neurons to learn better. Some examples of activation functions are Sigmoid, ReLU and Softmax.

Sigmoid Function is defined as:

$$\frac{1}{1+e^{-x}}.$$

ReLU (Rectified Linear Unit) function is defined as:

$$f(x) = max(0, x).$$

6. Natural Language Processing (NLP):

Syslog messages are a sort of human language. However, human language is a complex set of a huge variety of words that are used to express huge amounts of information. Different combinations of words, different shapes of chosen words, and even different topics can add to or change the meaning of expressed information. Given the nature of computers, it is challenging to interpret such information correctly. However, to override these obstacles the Natural Language Processing (NLP) is used. NLP is a branch of artificial intelligence that helps to interpret the human language for computers. The objective of NLP is to read, decode and understand human language.

There are several NLP techniques:

- Named Entity Recognition.
- Tokenization.
- Stemming and Lemmatization.
- Word Embedding.
- Natural Language Generation.

- Sentiment Analysis.
- Word2Vec.
- Sentence Segmentation.

Named Entity Recognition (NER) is popular NLP, which identifies names, nouns, etc. in a given phrase or paragraph. It is widely used for example for news categorization, in search engines or to help categorize feedback provided by customers' reviews.

The tokenization technique, as is obvious from its name, is splitting the text into tokens. Tokens can be understood as words, sentences, characters, numbers, etc.

Stemming is a process of reducing words into their root shape. It basically removes the suffixes, for example, it adjusts the word "denied" to "den".

Lemmatization is a process that returns the base form of the word. For example, it adjusts the word "denied" to "deny".

Word Embedding is the collective name for a set of language modelling and feature learning techniques in NLP, where words or phrases are mapped to vectors of real numbers. Such techniques are Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). However, BoW is the most used one. It creates "a bag" of all occurrences of each word, without caring about the position of the word in the text. Each word is basically represented by its own variable and the value of the variable means the number of occurrences of the word. To demonstrate this technique, the following two sentences are considered: "Deny IP due to Land Attack." and "Deny protocol connection spoof." At first, it tokenizes 35 the sentences and creates list of all words: "Deny", "IP", "due", "to", "Land", "Attack", "protocol", "connection" and "spoof". Then words are mapped to vectors of real numbers. First sentence is transformed into vector [1, 1, 1, 1, 1, 1, 0, 0, 0] and second sentence is transformed into vector [1, 0, 0, 0, 0, 0, 1, 1, 1].

Natural Language Generation (NLG), as it is clear from the name, it is a technique to convert structured data into the desired language. It is also called data storytelling because it can help to understand patterns or insights in structured data. It can compose structured data into understandable reports, for example, financial reports from the stock market.

Sentiment Analysis is the most common pre-processing technique for analysing "the emotion" of written text. Mostly it is analysing subjective texts, such as reviews, whether they are positive, negative, or neutral. That helps companies to better understand the reputation of their products and services and understand the customer experience.
The process of sentiment analysis is pretty straightforward: at first, it breaks the text down into tokens (words, phrases, sentences, etc.), identifies token(s) carrying sentiment, and assigns a sentiment score to it from within the range of -1 to 1.

Word2Vec is an unsupervised predictive deep learning-based model. It is shallow since it only uses 2 layers in its NN. It generates continuous dense vector representations of words that capture semantic and contextual similarity.**Word2Vec** leverage either the Continuous Bag of Words (CBOW) model or the Skip-gram model to create the WE representations, and they are described in the subsections below. Words that are more similar in context will be closed in the WE space than words from a different context. The original implementation uses hierarchical SoftMax as the output unit and represents the vocabulary as a Huffman binary tree. A Huffman binary tree assigns short binary codes to common words which in this case reduces the number of output units needed in the NN.

Sentence Segmentation is dividing a text into sentences or phrases. It is also called Sentence boundary detection as it identifies sentence boundaries between words. Sentence Segmentation is one of the basic NLP techniques as it is quite easy to understand and use.
There are several libraries that support NLP such as Natural Language Toolkit (NLTK) and Spacy.

Chapter 3: Proposed Solution

I. Introduction:

In this research, two methods are used to answer the research questions: Literature Review and Experiment. The design of this project is to analyse the data and consider suitable methods that can be used to find the anomalies from the log files. So, a literature review is conducted to gain knowledge on the machine learning techniques used previously for this sort of issue, we have also examined other deep learning architectures in other domains that could potentially address the problem at hand. This helped us in selecting the algorithms that can be used to develop the model. The selection of algorithms depends on how accurately they will predict the anomalies, which means selecting algorithms that show the best performance results by using conventional statistical measurement metrics. A literature review (survey) of existing machine learning algorithms in the domain that could potentially work best for this problem. The experiment is chosen over other research methods as it involves the manipulation of variables that are necessary which is important to obtain the results in this study. The type of problem addressed is dealt with by classification as the algorithm needs to classify an anomaly from the log file. The data which is collected is in its raw form and not fully structured or labelled. So, the log file is labelled while parsing the data, later only the data required to feed the algorithms were taken into consideration. This makes it labelled data which is why we used supervised machine learning techniques in the study. Whereas, unsupervised techniques are used when necessary which is discussed in the later sections.

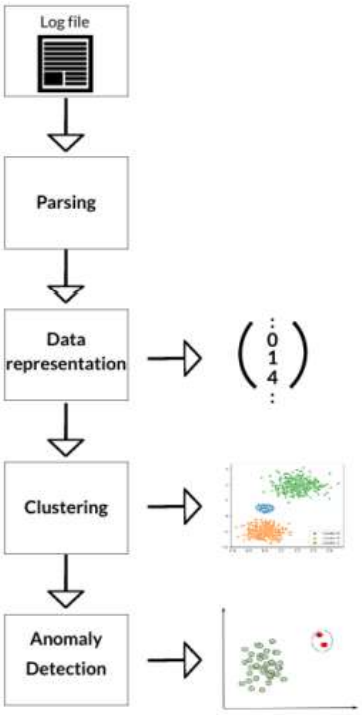


Figure 11: Method for finding anomalies

II. The process of log analysis:

From research conducted in this project, there seems to be a re-occurring series of steps that need to be performed in order to construct a complete log analysis system.

1. Data Log collection:

Logs are routinely generated by systems. The message that a log contains can be used for different purposes and one of them is anomaly detection. Therefore log collection is the first step in detecting anomalies in logs. Collecting and storing logs is a central part of log analysis. It is however not in the general interest of this paper to examine collection and storage of logs.

2. Log parsing:

To enable log analysis based on techniques such as machine learning, logs need to be transformed into structured events with the same type of fields in each log message. Some parts of a log message can be fairly easy to extract. For example, the timestamp is usually the first header in a log message. Other parts, such as free-text, can be more challenging to extract. The free-text message often contains a lot of information in various formats. These messages are often a combination of constant parts with variable values. For example, by parsing log messages, different event templates can be created based on the constant parts and the variable values are passed along with the template. Parsers are often divided into offline or online parsers. Offline means that the logs are parsed as a batch, in other words, the logs are first collected, and then the whole batch is parsed. Online meaning that logs are parsed in real-time as they are collected, in a streaming fashion. There exist different techniques and tools for log parsing. Some of these techniques include rule-based, source code-based, and data-driven parsing. A common approach when it comes to rule-based parsing is to use a regular expression (Regex). Using Regex does not work for general purposes since each type of log message requires a unique regular expression parsing and domain-specific expert knowledge. Source code parsing works by creating event templates from the source code of the program responsible for logging and then matching log messages to these templates to recover its structure. This technique requires access to the source code which can become complex when there are multiple systems to collect logs from. Data-driven parsing works by creating templates from the log data itself. The templates will depend entirely on the structure of the logs. The advantage of data-driven parsing is that it is more general and can be used for all log structures. It also does not require access to source code.

3. Features extraction:

In order to run anomaly detection algorithms on the log data, features that will be fed into the algorithm need to be extracted. This step can be done in a variety of ways and the best approach depends on the anomaly detection algorithm, the log dataset, and what kind of anomalies that need to be detected. Many features are extracted and most of them relate to user behaviour. The combined features aim to answer the four following questions, who, where, when, and what. In this case, the data is divided into user sessions and relevant data is extracted from each session. One example of a feature is the login username. This is categorical data and not all algorithms can accept categorical data as input. In other cases, one may use a time window to extract features. Different types of windows are used. These include fixed windows, sliding windows, and session windows. In the case of fixed windows and sliding windows the occurrences of certain log messages inside each time window can be counted and a matrix can be created where each row contains information about the occurrences of a specific type of log message in that particular time window. This matrix will contain strictly numerical data and can be used as input data to certain clustering algorithms.

2012-02-03 18:35:34 SampleClass6 [INFO] everything-normal-for-id 577725851

Date	2012-02-03
Time	18:35:34
Class Type	SampleClass6
Log Level	[INFO]
Log Type	everything-normal-for-id
Id Number	577725851

Table 1: Features Extraction

Fixed Window

The fixed window uses the timestamps of log messages and classifies logs that occur in the same time window as a sequence of logs. In this case, the time difference is constant through the entire execution of the feature extraction. The amount of log messages in one window is recorded and put into an event count vector. One of the disadvantages of using a fixed window is that some log messages that appear in one window may actually be related to logs in another

window. For example, two logs that together create an anomaly may be put in different time windows and in that way bypass detection. This can reduce the accuracy of anomaly detection.

Session Window

Instead of using the time to determine log sequences, sessions can be used. A session window uses identifiers in the log messages to separate sequences of log messages. There are a variety of identifiers that could be used when creating a session window. One example could be a user session. That means that every action taken by a specific user is put into the same session window. This method would find anomalies related to user behaviour. For example, if a user performs a certain action an unusual amount of times this can be seen as an anomaly. That user session would then be labelled as anomalous and further investigation is advised.

4. Pre-processing:

To make machine learning models perform better and make data easier to work with, the features can be refactored. Three common ways to do this are standardizing, normalizing, and changing the dimensions of the features.

Standardizing features

Machine learning models may perform poorly if the features are not somehow normally distributed, therefore standardization may be important for constructing an accurate model. Normal distribution means that the mean is zero and the standard deviation is one. If a feature is much larger than other features, this feature might dominate and the algorithm may be unable to interpret other features in a correct way. A feature is therefore standardized compared to how many times this feature occurs in other event vectors. For example, if a user performs an action 200 times in a time period and in the same time period the user performs another action one time. The action that is performed 200 times will then dominate the event vector. The action that was only performed once, could however be the abnormal behaviour of that user and not the action that was performed 200 times. Each action in an event vector is therefore standardized compared to how many times this action is performed by other users. The standardization of a feature is calculated as follows:

Where x is feature value with mean

$$z = \frac{x - \rho}{\sigma}$$

And standard deviation

$$\rho = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

Where N is the number feature values. $\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \rho)}$

Normalizing features

The process of normalizing data can be useful in many contexts of data mining and can sometimes have different meanings depending on the context. In this project, normalization involves turning individual samples of data in the form of vectors and scaling them to unit form. This means that the length of the vector is 1 and can usually be achieved by dividing each element in the vector by the Euclidean length of the vector. The normalization of a feature can be performed with the following formula:

$$x' = \frac{x}{||x||}$$

Where x is the original vector and ||x|| is the Euclidean length of the vector. The Euclidean length of the vector is calculated by the square root of the sum of each element squared:

$$||x|| = \sqrt{x_0^2 + x_1^2 + \ldots + x_{n-1}^2 + x_n^2}$$

Normalization gives the attributes in the dataset equal weights which can decrease redundancy and prevent "noisy" samples from taking over. K-means clusters data by measuring Euclidean distance between cluster centres. This technique can be highly affected by irregularities in the dataset which makes normalization effective.

Data Pre-processing

Collected datasets can have various formats and contain various data. For example, the dataset used in this thesis contains server logs with txt extension. Data pre-processing is crucial for data before being applied to machine learning models. It contains many steps like Tokenization, removing stop-words, and Normalization (Stemming and Lemmatization). Moreover, it also contains feature extraction (Word Embedding) where the data are encoded into numerical feature vectors. However, data pre-processing is handled by NLP as described in subsection 4.2. A further step that is needed before applying the data to the machine learning model is data splitting. Data splitting is a process that splits the data into training features, training labels and testing features, and testing labels, as described in subsection 4.2 below.

4.1 Natural Language Processing:

All machine learning algorithms are based on certain mathematical calculations. In order for the algorithm to be able to process the dataset as a text, then this text has to be tokenized. Moreover, these tokens need to be normalized and then transformed into numbers. There are

multiple options for how to transfer words into numbers. The set of all words used in the dataset is then called Bag of Words. If a word occurs in a certain log message, then its variable will have a value of 1, otherwise, it will have a value of 0. If this word is in the log message twice, the value will be 2, etc.

4.2 Data Splitting:

Given the nature of this thesis, there are two types of datasets being used: labelled and unlabelled. To ensure proper data pre-processing and feature extraction, there are two methods created. The difference between labelled and unlabelled datasets is pretty clear: a labelled dataset contains a log message and label and an unlabelled dataset contains only a log message. However, since the log message will be represented by a matrix of independent variables X and the label will be represented by dependent variable y, the difference in importing and splitting of the data will be in returned variables: importing and splitting the labelled dataset will return matrix X and vector y, importing unlabelled dataset will return only matrix X. Since this thesis is dealing with two completely different datasets, there are two different methods to import and split data. Both methods are described below.

Pre-processing and Splitting Log Dataset

As mentioned above, there are two types of datasets: labelled and unlabelled. At first, the dataset is loaded into memory, and then it is split into a matrix of independent variables X and a vector of dependent variable y.

To ensure only relevant values are imported into matrix X, there are a few columns from the dataset excluded. Namely, the first column contains date, the second one contains Time, the third one contains Class Type and the sixth one contains Id Number. All these values are irrelevant, and hence are not imported into matrix X. Importing values into vector y needs to be specified as well. Since the dataset contains different written labels and this project is focusing on anomalies, anomalous labels are transformed into value 1 and benign traffic is transformed into value 0. Some of the values in this dataset are not processable. There are two columns that instead of numeric values contain nothing (Nan) or infinity (inf). To take care of such data, there are two simple for loops. First, one goes through both columns and calculates the average value inside of them and also to find the maximum value. The second one goes through the same columns again and replaces Nan with an average value and inf with maximum value.

4.3 Machine Learning Structure and Models:

After the data is pre-processed, it is ready to be fed to the machine learning models. Almost all models are implemented using the Python library for machine learning Scikit-learn. Only the Artificial Neural Network model is implemented using the Python library Keras. However, since the project deals with classification problems then the implementation of classification machine learning models will be described. In unsupervised learning, clustering algorithms will be used.

4.3.1 Logistic Regression Model:

Parameters were chosen by manual tests and comparison of the precision. Implementation is shown on Chapter 4 Section 2.3.

4.3.2 K-Nearest Neighbor Model:

The model uses the function Hamming and utilizes 15 neighbors. A number of neighbors were chosen randomly, only following a simple rule of not using multiple of a number of classes. Implementation is shown on Chapter 4 Section 2.2.

4.3.3 Naive Bayes Model:

The model uses the Gaussian Naive Bayes algorithm, which is specifically written for classification problems, hence effective for the challenge of this thesis. Implementation is shown on Chapter 4 Section 2.1.

4.3.4 Decision Tree Classification Model:

The model uses the default number of samples for a node to be a leaf of 1, a minimum of a number of samples to split internal node of 2 and strategy to choose split of each node, splitter, is chosen as best. To measure the quality was chosen function entropy and used a random state of 0. These parameters were chosen by manual tests and comparison of the precision. Implementation is shown on Chapter 4 Section 2.4.

4.3.5 AdaBoost Classification Model:

The AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. Implementation is shown on Chapter 4 Section 2.5.

4.4 Log Analysis Tools:

The Human Machine Interface (HMI) is a fundamental element in the use of any IT project and contributes enormously to its success. It presents the fruit of the work carried out. Therefore, an interface must be as close as possible to the different modes of human perception and communication, that is to say effective, easy to use and adapt to its context of use.

In this context, in this section, the existing tools that are examined in this project are outlined and described in terms of how functionality such as parsing and anomaly detection is performed.

4.4.1 Python:

Python facilitates developers to increase their confidence and Productivity in their developing software from development to Deployment and maintenance. The benefits of making Python the Perfect solution for learning and AI-driven projects include simplicity And consistency, flexibility, access to powerful AI and machine learning (ML) libraries and frameworks, platform independence, and large Communities. These things increase the popularity of the language.

Figure 12:Python Logo

4.4.2 Anaconda Navigator:

Anaconda is an open-source distribution of the Python and R programming Languages for data science that aims to simplify package management and Deployment. Package versions in Anaconda are managed by the package Management system, conda, which analyses the current environment before Executing an installation to avoid disrupting other frameworks and packages

Figure 13:Anaconda Logo

4.4.3 Jupyter Notebook:

The Jupyter Notebook is an open-source web application that allows you To create and share documents that contain live code, equations, Visualizations, and narrative text. Its uses include data cleaning and Transformation, numerical simulation, statistical modelling, data Visualization, machine learning, and much more.

Figure 14:Jupyter Notebook

4.4.4 Flask:

Flask is a web framework, it's a Python module that lets you develop Web applications easily. It's has a small and easy-to-extend core: it's a Micro framework that doesn't include an ORM (Object Relational Manager) or such features.

Figure 15:Flask Logo

4.4.5 Sklearn:

Scikit-learn (also known as sklearn) is a free software machine learning Library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, And is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Figure 16:Sklearn Logo

4.4.6 Gensim:

Gensim is an open-source library for unsupervised topic modelling, Document indexing, retrieval by similarity, and other natural language Processing functionalities, using modern statistical machine learning. Gensim is implemented in Python and Cython for performance.

Figure 17:Gensim Logo

4.4.7 Natural Language Processing (NLP) :

Natural language processing is a subfield of linguistics, computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and Analyse large amounts of natural language data.

Figure 18:NLP with Python

4.4.8 Pandas:

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

Figure 19:Pandas Library

4.4.9 Keras:

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML.

Figure 20:Keras Library

4.4.10 Matplotlib:

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Figure 21:Matplotlib Library

4.4.11 TensorFlow:

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Figure 22:TensorFlow Logo

4.4.12 Bootstrap:

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Figure 23:Bootstrap Logo

Chapter 4: Experimental Results and Discussion

I. Methodology:

In this part of the study, each machine learning algorithm was used to analyse the database. The supervised learning algorithms were trained and tested for predicting channel position from the different classification schemes. Once all the algorithms had analysed the data, the evaluation metrics for each algorithm were compared to test the hypotheses previously mentioned and draw conclusions.

II. Experimental Results:

1. Unsupervised Learning Results:

1.1 Word2vec:

Word embeddings are an essential part of solving many problems in NLP, it depicts how humans understand language to a machine. Given a large corpus of text, word2vec produces an embedding vector associated with each word in the corpus. These embeddings are structured such that words with similar characteristics are in close proximity to one another. Given an input word, skip-gram will try to predict the words in context to the input whereas the CBOW model will take a variety of words and try to predict the missing one.

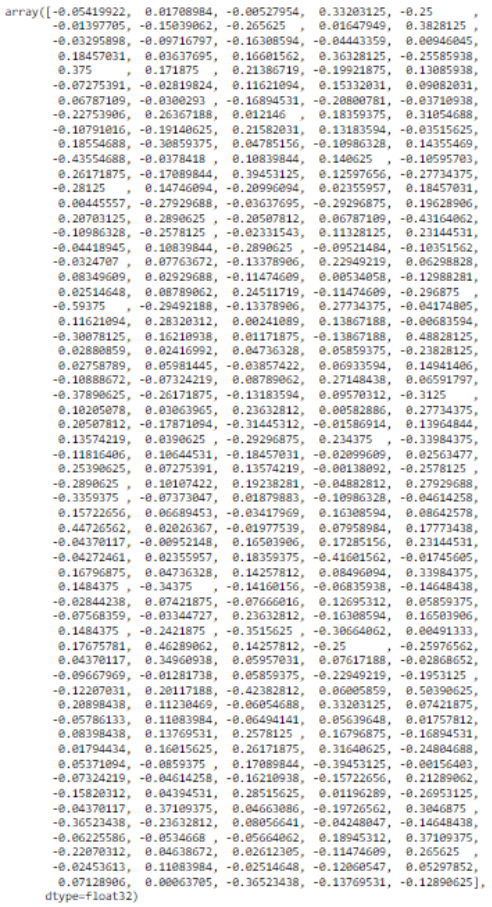


Figure 24:Word2vec Model Result

2. Supervised Learning Results:

2.1 Naives Bayes:

Naive Bayes is a probabilistic classifier that returns the probability of a test point belonging to a class rather than the label of the test point. It's among the most basic Bayesian network models, but when combined with kernel density estimation, it may attain greater levels of accuracy. This algorithm is applicable for Classification tasks only, unlike many other ML algorithms which can typically perform Regression as well as Classification tasks.

```
MultinomialNB()
                        precision    recall  f1-score   support

          detailforid       1.00      0.99      1.00       142
everythingnormalforid       1.00      1.00      1.00        28
          incorrectid       0.33      1.00      0.50         1
            missingid       1.00      1.00      1.00         1
      systemproblemid       0.00      0.00      0.00         1
   verbosedetailforid       1.00      1.00      1.00       232

             accuracy                           1.00       405
            macro avg       0.72      0.83      0.75       405
         weighted avg       1.00      1.00      1.00       405

[[141   0   1   0   0   0]
 [  0  28   0   0   0   0]
 [  0   0   1   0   0   0]
 [  0   0   0   1   0   0]
 [  0   0   1   0   0   0]
 [  0   0   0   0   0 232]]
Accuracy of the model on Testing Sample Data: 1.0

Accuracy values for 5-fold Cross Validation:
 [0.99258176 1.         0.99813744 1.         0.99689136]

Final Average Accuracy of the model: 1.0
```

Figure 25: Naive Bayes Model Result

2.2 KNN:

KNN works by finding the distances between a query and all the examples in the data, selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).

```
KNeighborsClassifier(n_neighbors=15)
                        precision    recall  f1-score   support

          detailforid       1.00      0.99      1.00       142
everythingnormalforid       1.00      1.00      1.00        28
          incorrectid       0.00      0.00      0.00         1
            missingid       0.00      0.00      0.00         1
      systemproblemid       0.00      0.00      0.00         1
   verbosedetailforid       0.98      1.00      0.99       232

             accuracy                           0.99       405
            macro avg       0.50      0.50      0.50       405
         weighted avg       0.98      0.99      0.99       405

[[141   0   0   0   0   1]
 [  0  28   0   0   0   0]
 [  0   0   0   0   0   1]
 [  0   0   0   0   0   1]
 [  0   0   0   0   0   1]
 [  0   0   0   0   0 232]]
Accuracy of the model on Testing Sample Data: 0.99

Text(0, 0.5, 'Testing Accuracy')
```

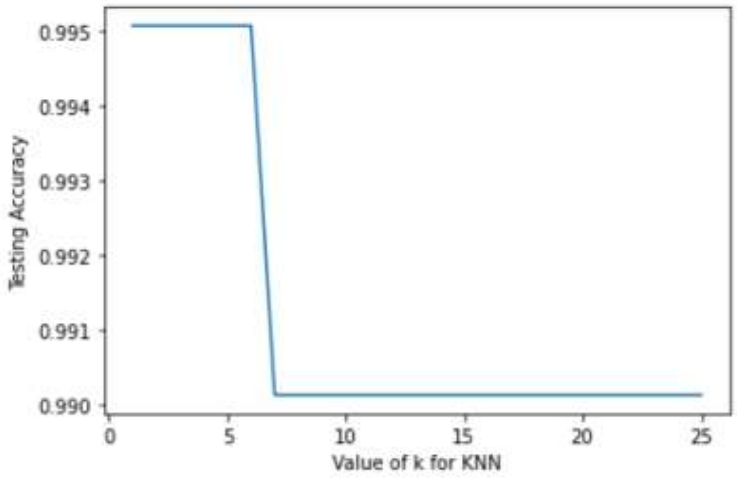


Figure 26: KNN Model Result

2.3 Logistic Regression:

In the Machine Learning world, Logistic Regression is a kind of parametric classification model, despite having the word 'regression' in its name.

This means that logistic regression models are models that have a certain fixed number of parameters that depend on the number of input features, and they output categorical prediction, like for example if a plant belongs to a certain species or not.

```
          0         1         2         3         4         5         6  \
0  1.000000  0.000000  0.644068  0.478723  0.706943  1.000000  0.000000
1  1.000000  0.000000  0.644068  0.478723  0.706943  1.000000  0.000000
2  1.000000  0.000000  0.644068  0.478723  0.706943  1.000000  0.000000
3  0.160417  0.224824  0.000000  0.392287  0.892696  0.721739  0.583156
4  1.000000  0.000000  0.644068  0.478723  0.706943  1.000000  0.000000

          7         8         9  ...       292       293       294       295  \
0  0.186207  0.818418  0.450224  ...  1.000000  0.370797  0.674279  0.000000
1  0.186207  0.818418  0.450224  ...  1.000000  0.370797  0.674279  0.000000
2  0.186207  0.818418  0.450224  ...  1.000000  0.370797  0.674279  0.000000
3  0.520259  0.000000  0.000000  ...  0.526554  0.109510  0.834585  0.199778
4  0.186207  0.818418  0.450224  ...  1.000000  0.370797  0.674279  0.000000

        296  297       298       299              Survived  \
0  0.844258  1.0  0.921053  0.000000     verbosedetailforid
1  0.844258  1.0  0.921053  0.000000     verbosedetailforid
2  0.844258  1.0  0.921053  0.000000     verbosedetailforid
3  0.967440  0.0  0.732730  0.218072  everythingnormalforid
4  0.844258  1.0  0.921053  0.000000     verbosedetailforid

      Predicted_Survived
0     verbosedetailforid
1     verbosedetailforid
2     verbosedetailforid
3  everythingnormalforid
4     verbosedetailforid

[5 rows x 302 columns]
                      precision    recall  f1-score   support

          detailforid       0.99      1.00      1.00       142
everythingnormalforid       1.00      1.00      1.00        28
          incorrectid       0.00      0.00      0.00         1
            missingid       1.00      1.00      1.00         1
      systemproblemid       0.00      0.00      0.00         1
   verbosedetailforid       1.00      1.00      1.00       232

             accuracy                           1.00       405
            macro avg       0.66      0.67      0.67       405
         weighted avg       0.99      1.00      0.99       405

[[142   0   1   0   0   0]
 [  0  28   0   0   0   0]
 [  0   0   0   0   0   0]
 [  0   0   0   1   0   0]
 [  0   0   0   0   0   0]
 [  0   0   0   0   1 232]]
Accuracy of the model on Testing Sample Data: 0.99
```

Figure 27: Logistic Regression Model Result

2.4 Decision Trees:

The decision of making strategic splits heavily affects a tree's accuracy. The decision criteria are different for classification and regression trees.

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision

tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

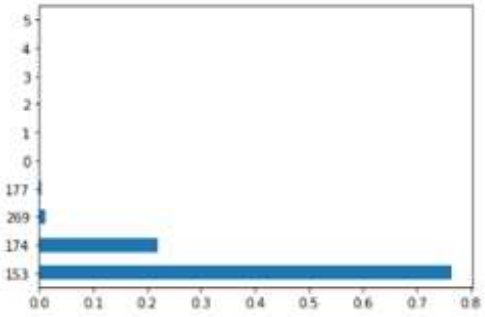
```
DecisionTreeClassifier(max_depth=20)
                        precision    recall  f1-score   support

          detailforid       0.99      1.00      1.00       142
everythingnormalforid       1.00      1.00      1.00        28
          incorrectid       0.00      0.00      0.00         1
            missingid       0.50      1.00      0.67         1
      systemproblemid       0.00      0.00      0.00         1
   verbosedetailforid       1.00      1.00      1.00       232

             accuracy                           1.00       405
            macro avg       0.58      0.67      0.61       405
         weighted avg       0.99      1.00      0.99       405

[[142   0   0   0   0   0]
 [  0  28   0   0   0   0]
 [  1   0   0   0   0   0]
 [  0   0   0   1   0   0]
 [  0   0   0   1   0   0]
 [  0   0   0   0   0 232]]
Accuracy of the model on Testing Sample Data: 0.99

<AxesSubplot:>
```




```
<matplotlib.image.AxesImage at 0x20ce3b375e0>
```

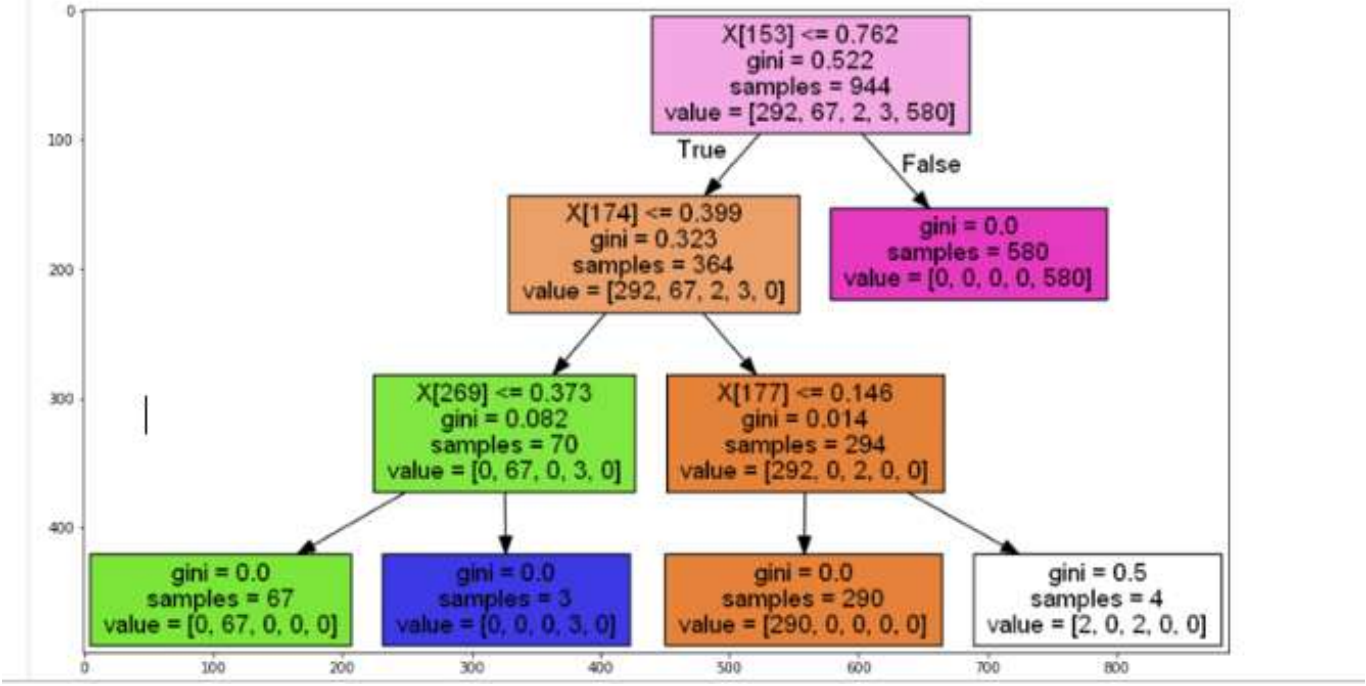


Figure 28: Decision Tree Model Result

2.5 AdaBoost:

Before applying AdaBoost to any dataset, one should split the data into train and test. After splitting the data into train and test, the training data is ready to train the AdaBoost model. This data has both the input as well as output. After training the data, our algorithm will try to predict the result on the test data. Test data consists of only the inputs. The output of test data is not known by the model. Accuracy can be checked by comparing the actual output of the test data and the output predicted by the model. This can help us conclude how our model is performing and how much accuracy can be considered, depending on the problem statement.

```
AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=2),
                   learning_rate=0.01, n_estimators=20)
                     precision    recall  f1-score   support

          detailforid       0.99      1.00      1.00       142
everythingnormalforid       0.97      1.00      0.98        28
          incorrectid       0.00      0.00      0.00         1
            missingid       0.00      0.00      0.00         1
      systemproblemid       0.00      0.00      0.00         1
   verbosedetailforid       1.00      1.00      1.00       232

             accuracy                           0.99       405
            macro avg       0.49      0.50      0.50       405
         weighted avg       0.99      0.99      0.99       405

[[142   0   0   0   0   0]
 [  0  28   0   0   0   0]
 [  1   0   0   0   0   0]
 [  0   1   0   0   0   0]
 [  0   0   0   0   0   1]
 [  0   0   0   0   0 232]]
Accuracy of the model on Testing Sample Data: 0.99
```

Figure 29: AdaBoost Model Result

3. Machine Learning Metrics and Performance:

To compare the performance of the machine learning models, there are different performance comparison methods implemented. First one is confusion matrix, which is a table that shows summary of correct and incorrect predictions of a model. Next ones are metrics such as accuracy, precision, recall and F1-score.

Description of confusion matrix and metrics can be found in Section 4.

The metrics used to evaluate the performance of the approaches used are mentioned below:

3.1 Accuracy:

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. If we have high accuracy for a particular

Model then the model is best.

$$Accuracy = TP + TN/TP + FP + FN + TN$$

3.2 Precision:

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive r ate.

$$Precision = TP/TP + FP$$

3.3 Recall:

Recall is the ratio of correctly predicted positive observations to the all observations in actual class. It has values between 0 and 1. Higher the recall, better performing the model is.

$$Recall = TP/TP + FN$$

3.4 F1 Score:

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. It ranges from 0 and 1, higher the F1 score better the model is performing.

$$F1Score = 2 * (Recall * Precision)/(Recall + Precision)$$

4. Comparison of Performance of Machine Learning Models:

This chapter compares the performance of twelve different machine learning algorithms mentioned above. Each has a different approach. For results comparison, the confusion matrix and metrics calculated from it such as accuracy, precision, recall and F1-score are used. Confusion matrix is a table showing summary of predicted labels compared to actual labels.

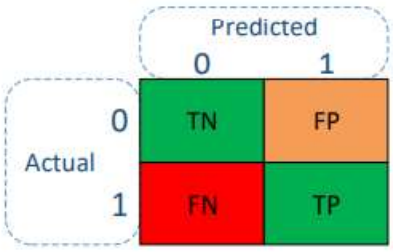


Figure 30: Confusion Matrix

As shown in figure 30, confusion matrix's fields have certain names. As a positive result, the label is considered as an anomaly, and as a negative result as a label not anomaly. Explanation of each field is below:

- True Positive means predicted was positive label (anomaly) and it was correct.
- False Positive means predicted was positive label (anomaly), where actually was supposed to be a negative label (not anomaly). It is also called Type 1 Error or False alarm.
- False Negative means that negative label (not anomaly) was predicted and it was incorrect. It is also called Type 2 Error. This one is critical as it shows anomaly was missed.
- True Negative means predicted negative label (not anomaly) was correct.

Metrics calculated from confusion matrix are Accuracy, Precision, Recall and F1-Score. Accuracy (also called Classification Rate) is a ratio of all correct predictions to all predictions and is calculated by equation shown below:

$$Accuracy = \frac{TP + TN}{P + N}.$$

Precision is a ratio of how many predicted anomalies are correct and is calculated by following equation:

$$Precision = \frac{TP}{TP + FP}.$$

Recall (sometime also called Sensitivity) is a ratio of how many actual anomalies are predicted correctly. This metric will be key for comparison of implemented tools. Recall is calculated by following equation:

$$Recall = \frac{TP}{TP + FN}.$$

F1-Score (also called F-measure) is harmonic mean of Recall and Precision. It will be always closer to smaller value of either Recall or Precision. F1-Score is calculated by following equation:

$$F1\text{-}Score = \frac{2 * TP}{2 * TP + FP + FN} = \frac{2 * Recall * Precision}{Recall + Precision}.$$

4.1 Results of Machine Learning Performance and Discussion :

Results of the machine learning performance can be seen in Table 2 .The most important metric for anomaly detection is Recall. The results show that the most accurate predictions were made

by supervised machine learning models. Almost 100% accurate was model Naives Bayes, followed closely by models AdaBoost and Decision Tree Classifier. And Finally, KNN and Logistic Regression Models. All were able to predict almost all anomalies correctly.

	Accuracy	Precision	Recall	F1-Score
Naïve Bayes	1.0	0.721	0.937037	0.75
AdaBoost	0.99	0.493	0.933333	0.496
Decision Tree Classifier	0.99	0.581	0.929630	0.611
KNN	0.99	0.49	0.914815	0.49
Logistic Regression	0.99	0.665	0.874074	0.666

Table 2: Performance results of Machine Learning Models on dataset

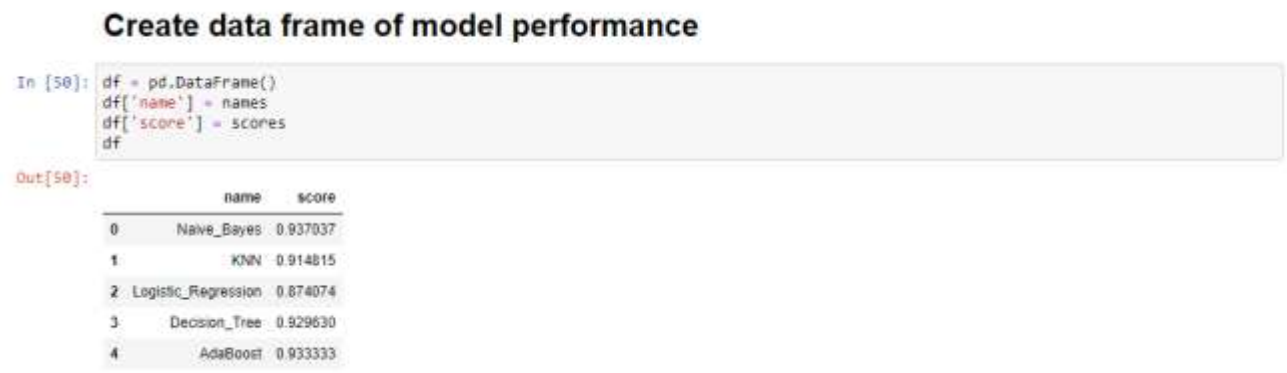


Figure 31: Models Performance

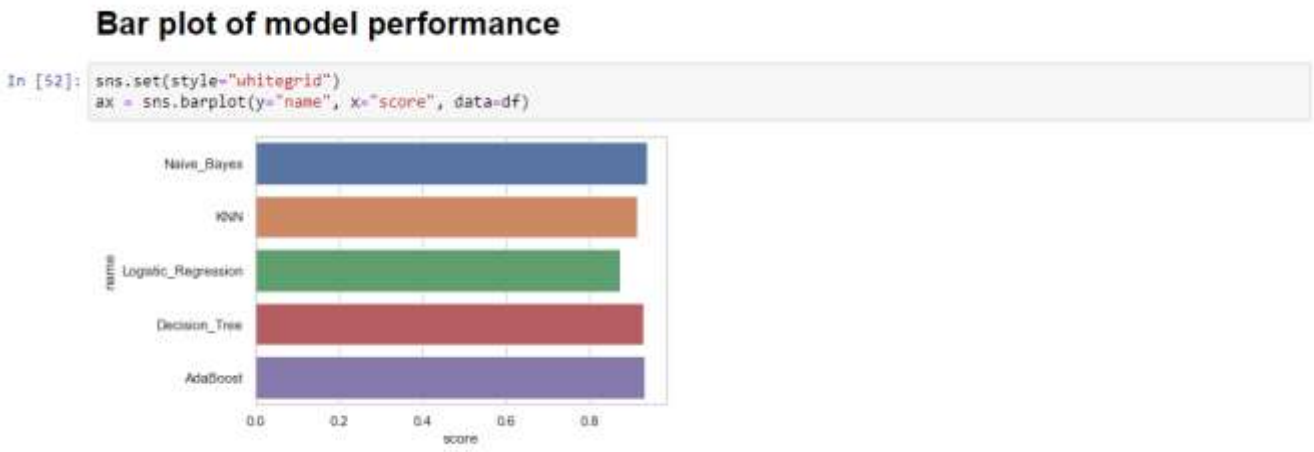


Figure 32: Recall results of Machine Learning Models on dataset

4.2 Conclusion :

As argued, obtaining data directly from the field has very many limitations. Due to these limitations present, artificial intelligence work has been viewed as the best and strongest tool which can aid in supplementing such information and help generate the synthetic well logs. The sets of tools that are used in this research address the need for developing the respective well logs in synthetic form.

Artificial intelligence is a reliable and promising technology that can significantly contribute to solving Anomaly Detection related problems especially when it comes to the importance of fast decision-making processes.

III. Conclusion and Future Work:

This project helps in finding an optimal solution for the anomaly detection problem in various and large log files that has been a problem. Though there have been many researches and studies performed based on this, this project covered studied most of the possible ways that could address this problem.

- First, applying the suitable parsing method so that there won't be any loss of useful information from the log files.
- Second, the test logs are studied to take only the part of the log that is actually useful for this, as the log data contains each and every information sometimes which might not be useful or does not have any information regarding the anomalies.

Taking this information gives only the false predictions of the normal data which is even more disastrous. But, suitable parsing technique and the machine learning models were chosen after a careful literature review.

As future work, developing a method or tool that might actually take the useful part of the log files and then prioritize based on whether the log file which is generated after successful execution of hardware or software tests. Providing a solution on how to avoid the anomalies occurred after finding where they have caused would be a better improvement for this work.

General Conclusion

Machine learning is definitely showing potential in the area of log analysis. As shown in this thesis, it is possible to achieve promising results using unsupervised machine learning for anomaly detection. However, achieving both an accuracy and recall of 100% seems unlikely. There will always be a risk of false positives and false negatives when using machine learning which is why it is best used as a complement to existing analysis methods.

Anomalies in log data are often indicative of suspicious behavior that would benefit from the additional investigation. This is why anomaly detection is useful in log analysis. It could indicate attacking attempts, successful attacks, or simply unwanted behavior. Anomalies do not however guarantee that the behavior is unwanted. Anomalies are usually found by comparing current events with previous events and classifying anomalies as those that differ above a certain threshold. Anomalies that are classified in this way do not have to indicate unwanted behavior, they can simply be unusual behavior.

Anomaly detection is very useful for detecting unknown security issues and unusual behavior. Creating rules to detect and protect against yet unknown threats can be very difficult, if even possible. As attacks become more complex, so must rules. It can also be very difficult to determine what normal behavior is. By using machine learning normal behavior and patterns can be defined. Comparing this with new behavior makes it possible to detect anomalies that could point to security-related incidents. One potential problem is a high false-positive rate that could make it necessary to perform further analysis on the detected anomalies. In these cases, it could still save time since there is a smaller portion of high-risk logs that needs to be analyzed further.

From this project, five different steps have been compiled to go from raw log data to identify anomalies. These include log collection, parsing, feature extraction, preprocessing, and anomaly detection. There are many ways to perform each of these steps all depending on the data and purpose. For parsing, there are three options, data-driven, rule-based, or source code parsing. The more structured and the more common the structure of logs from different devices are, the easier parsing becomes. Feature extraction means selecting information where anomaly detection will be performed. This depends on the purpose and what kind of anomaly

detection that should be used. Features could be, simply choosing certain variable values or a more complex approach such as a session window. Depending on the data, it can be necessary to perform some prepossessing before using it as input in a machine learning algorithm. For example, normalization or dimension reduction. For performing anomaly detection, there are numerous machine learning options. When applying a machine learning model it is important to have a good understanding of possible strengths and weaknesses of the model to be able to interpret the output. Depending on the type of log analysis that is requested, different tools will have different advantages. When it comes to developing your own implementation there is the advantage of being able to customize the model to any extent. It can also be easier to understand since the functionality is not masked in any way. If the only goal is to use a commonly known machine learning algorithm for general anomaly detection, spending time and money on existing tools may be unnecessary. Although existing tools often offer visualization possibilities that can be helpful in many situations. Using an existing tool can also be advantageous if the user is not concerned with understanding the underlying functionality or requires additional customization. Existing tools also offer additional functionality which can be useful depending on the situation. The forensic capabilities tend to be much more prevalent in existing tools which are important when reconstructing events and assessing the damage.

References

- SHARMA, Avneesh: How Different are Conventional Programming and Machine Learning?. Available from URL: . How Different are Conventional Programming and Machine Learning? - KDnuggets
- How Machine Learning Can Enable Anomaly Detection [online].. Available from URL: How Machine Learning Can Enable Anomaly Detection | by Countants | DataDrivenInvestor
- What is Machine Learning? A definition [online]. Available from URL: https://expertsystem.com/machine-learning-definition/
- ROSEBROCK, Adrian: Anomaly detection with Keras, TensorFlow, and Deep Learning [online. Available from URL: Anomaly detection with Keras, TensorFlow, and Deep Learning - PyImageSearch
- BROWNLEE, Jason: Supervised and Unsupervised Machine Learning Algorithms. Available from URL: Supervised and Unsupervised Machine Learning Algorithms (machinelearningmastery.com)
- LEONEL, Jorge: Classification Methods in Machine Learning. Available from URL: Classification Methods in Machine Learning | by Jorge Leonel | Medium
- SRIVASTAVA, Tavish: Introduction to k-Nearest Neighbors: A powerful Machine Learning Algorithm (with implementation in Python & R). Available from URL: K Nearest Neighbor | KNN Algorithm | KNN in Python & R (analyticsvidhya.com)
- BROWNLEE, Jason: Naive Bayes Classifier from Scratch in Python . Available from URL: Naive Bayes Classifier From Scratch in Python (machinelearningmastery.com)

- Decision Tree Classification Algorithm. Available from URL: Machine Learning Decision Tree Classification Algorithm - Javatpoint
- Logistic Regression in Python. Available from URL: https://realpython.com/logistic-regression-python/
- Implementing the AdaBoost Algorithm from Scratch. Available from URL: https://www.kdnuggets.com/2020/12/implementing-adaboost-algorithm-from-scratch.html
- Word2Vec Implementation. Available from URL: https://towardsdatascience.com/a-word2vec-implementation-using-numpy-and-python-d256cf0e5f28
- BILYK, Volodymyr: Guide to Unsupervised Machine Learning: 7 Real Life Examples. Available from URL: Guide to Unsupervised Machine Learning: types of unsupervised learning (theappsolutions.com)
- GARBADE, Dr. Michael J.: A Simple Introduction to Natural Language Processing. Available from URL: https://becominghuman.ai/a-simple-introduction-to-natural-language-processing-ea66a1747b32?gi=ac7200987fec
- Confusion Matrix in Machine Learning. Available from URL: . Confusion Matrix in Machine Learning - GeeksforGeeks