Big-data analytics using rules-based techniques and machine learning to support business decisions

Joshua Uzochukwu Nwankwo

Student ref: 16022703

April 2018

Abstract

Log and data files contain an immense amount of information about the current state of a system. The most common technique used by most system administrators and support executives is to manually define a set of catch phrases and strings such as ERROR, WARN or TRACE in an attempt to retrieve lines containing those words. These methods has been criticized as impractical and inefficient, depending heavily on pattern-matching which requires a lot of domain experience in writing regular expressions. This is in addition to relying on overly simplistic heuristics and making assumptions about the nature of these files. In this thesis, I have presented some techniques for predicting workload, understanding end-user affinity and detecting anomalies in IT systems by exploiting the intrinsic structures within these files. This project formulated workload prediction as a supervised regression problem and then goes on to develop a model for future predictions. Understanding end user's system usage pattern was framed as a time-series problem while anomaly detection was formulated as a supervised classification problem. Gaussian Naive Bayes (GNB) and Support Vector Machines (SVM) were used to address this classification problem and both methods were benchmarked against each other. My results suggest that Gaussian NB performs better and is more accurate than SVMs for classification problems of this sort.

Table of Contents

Chapter 1

Introduction

1.1 Motivation

Big Data refers to a large amount of raw data that has been collected, stored and analyzed. This data contains a huge wealth of information which can be effectively utilized by organizations to make better decisions. Much of big data is unstructured, that is they don't have any form of schema to them. This makes them very difficult for system administrators to analyse them. The traditional approaches used by most administrators is defining a set of rules which would then be used to build regular expressions for skimming through huge log files. However, this is approach is not scalable in the long run will need to be updated frequently to detect new anomalies. Both large and small companies are using big data and related analysis approaches as a way to gain more information to better support their company and serve their customers, benefiting from the advantages of big data.Therefore leveraging machine learning techniques for anomaly detection could potentially save time, resources and cost. The purpose of this project is to explore the applications of machine learning for anomaly detection thus supporting business decisions.

1.2 Project Objectives

The objective of this thesis is to explore the use of machine learning techniques when diagnosing IT system faults, predicting anomalies based on historical data, data files and log files

accumulated over a period of time. This project seeks to implement a solution for analysing huge data sets and harnessing that wealth of information using machine learning techniques for improving the decision making process.

1.3 Thesis Organisation

This thesis is organised into six chapters. The second chapter would present the business case, research questions and literature review on related research. The third chapter would give some theoretical and mathematical aspects that have been used during the course of this project. This chapter provide some theoretical underpinnings. The fourth chapter introduces the methodology, the procedure and steps that were used during the experiments. It also analyses and evaluates the results. Chapter 6 presents the conclusion to the project and future directions.

Chapter 2

Background

2.1 Introduction

The focus of this chapter is to introduce the general background of this project and thesis. Then, the research questions of this thesis will be stated followed by a brief summary of the methodology. The end of the chapter will present the thesis structure and relationship between the thesis chapters.

2.2 Big data and the problem of anomaly detection

The increasing complexity of information technology (IT) systems demands a correspondingly greater effort for systems management. Today, many systems management tasks such as system configuration, performance analysis, performance tuning, error handling, and availability management are often performed manually. This work can be time-consuming and error-prone. Moreover, it requires a growing number of highly skilled personnel, making IT systems costly. IBM for instance, as exploited the use of an autonomic computing initiative, which is a core element of IBM's e-business on demand strategy to addresses this problem by developing and providing powerful concepts for self-management, including new self-healing, self-protecting, self-optimizing, and self-configuring capabilities. The goal is to reduce the burden associated with the management and the operation of IT systems. Autonomic computing systems should simply work, repairing and tuning themselves as needed. This requires

that such systems be able to protect themselves, to identify upcoming problems, and to make required reconfigurations dynamically in order to resolve problems.

Technology companies such as Facebook, Google, eBay and Yahoo have used big data technology at scale for mining complex data sets. As such, other technology companies are harnessing big data related technologies to unlock a wealth of information in their own datasets (Hernández et al., 2017). These data sets includes both structured and semi structured data such as comma-separated values (CSV) file, database logs, emails, application event log files, network traffic logs and user activity logs. Identifying the root cause of application slowness/performance issues, insider or external network intruder activity and consumer purchase pattern are some of the useful information embedded within these data sets. These wealth of information is key to the survival and continuity of the business which those application serves. Product investment decision can be influenced by information hidden within these files. For example, a company's product re-branding strategy or product profitability decision can be influenced by identifying consumer purchase patterns embedded within these log files. In addition, event log files is the right place to look when boosting system performance is key to improving user experience. Log files contains an audit trail of all actions performed by a system and are sometimes the only way to discover errors and exceptions in the underlying system (Zwietasch 2014).

The value of these wealth of information are not without challenges this is due to advancement in technology and newly classified security challenges at the workplace, individuals with malicious intentions continue to improve their craft. Therefore, it becomes increasingly important to develop effective mechanisms and techniques to protect IT systems which serves the business from these challenges. Some of these measures includes improving our ability to spot errors that could potentially lead to incorrect computation, spot exceptions that would cause system downtime, improve our ability to detect anomalies and attacks which could lead to data theft and improving our ability to recommend "Next Best Action" to increase productivity of system administrators rather than all sorts of error-prone manual correlation analysis. One common problem with computing systems is the concept of detecting anomalies. Anomaly detection is extracting a set of patterns with a data set which do not conform to the regular behaviour and this is useful for monitoring the health of computing systems

(Chandola et al. 2009). Traditionally, most log analytic tools use human based inspection and pattern-matching techniques to identify system anomalies. These techniques basically use a set known of predefined-patterns which has to be updated manually to capture all set of known exceptions (Ma 2003). These technique is not scalable and completely fails with new emerging threats because it gets out-paced with the rate at which new exceptions are generated. In a paper titled "Machine Learning in Automated Text Categorization", Sebastiani (2002) emphasised the flaws of a manual approach to Text Categorization (TC) stating that it increases cost in terms of hiring a knowledge expert to write the best set of pattern-matching rules and this approach is less accurate when compared to that achieved by a text classifier. Zwietasch (2014) underscores the methods of identifying events within a log file, he suggested that using more sophisticated techniques based on pattern extraction (unsupervised learning) leads to more accurate results than any other human-based technique. Therefore a human-based approach to log analysis is far from intelligent because it heavily depends on some individual manually updating a set of pattern-matching keywords.

Technical support executives, system administrators of enterprise applications heavily depend on alert and notification monitoring tools to ensure that all applications are up and running, detecting and deploying countermeasures to mitigate against cyber-attacks. In such complex environment time is of essence, every minute counts and the failure or attack of one subsystem could bring down the entire cluster of applications. This could affect end users, if those end users are customers it could have serious financial implications leading to losses in revenue. Supporting such complex systems could be very difficult, therefore a real-time (millisecond delay) alert and notification system which is intelligent enough to spot irregularities, preempt exceptions and recommend "Next Best Action" would not only save the business millions of dollars but would improve consumer experience and increase customer satisfaction.

2.3 Business Case

2.3.1 Inefficiency and productivity of system administrators

System administrators have to debug, searching through millions of lines of log files. The complexity involved is time-consuming which potentially leads to inefficiency and less productivity.

2.3.2 Decline of front-end user(s) perception

Front-end system users constantly complain of errors as a result of missing records or incorrect data which came in as part of a faulty ETL process. ETL cycle should be smart enough to preempt or flag and alert users of potential issues before they actually occur, that is preventive as opposed to reactive.

2.3.3 Product innovation and optimization

End users visit some pages and modules more than other modules, users get very frustrated when too much time is spent trying to reload a page.Using big-data analytics we could analyse event log files to understand end user affinity and browsing behavior.

2.4 Research questions

On the premise of the project background and business case,the following research questions are formulated:

- How can companies harness machine learning models to drive business decisions?
- Can supervised learning techniques be used to predict an application workload?
- Can Gaussian models be applied to solve classification problems when trying to identify anomalous records in a data set?
- Can statistical methods be used to improve user experience by identifying frequently used modules?

- Can time-series metrics be explored to identify system usage drops?

This project will attempt to answer these questions.

2.5 Related Research

An essential aspect of system monitoring includes troubleshooting and detecting computer systems issues within an information technology infrastructure. These infrastructures includes data centers, server farms and networked environments. Sometimes these systems perform poorly, render missing or incorrect information or very often crash to a halt due to variable number of factors. The task of detecting the cause of failure which is often referred to as anomaly detection involves gathering huge amounts of data and making sense of such information. This information is considered the number one source of truth when investigation "what went wrong". Network activity logs, system event logs, CPU/memory utilization logs and data files are some of the places to look when an issue occurs; and are considered a vital source of information when diagnosing information systems related issues (Saneifar et al. 2009).

2.5.1 Feature extraction

Log analysis using machine learning techniques requires input of some sort a s numerical feature vector representations. One of the biggest challenges with log analysis is feature vector extraction which is basically the conversion of log files which are mostly in textual forms to numerical vector representations (Oliner et al. 2012). This is because most machine learning models only accept numbers as opposed to texts. To address this problem, Zheng et al. (2009) presented a method which involves three integrated steps during the log preprocessing phase in an attempt to increase the number of extracted features for failure prediction when analysing log files. While the author claimed this method was effective in identifying failure patterns, the study did not address the problem of dimensionality reduction. Lim et al. (2008) discovered that extracting unique messages in log files could increase the number of extracted features which could potentially increase dimensionality. They proposed a method to address

this problem by replacing features such as URL, dates, timestamps, memory and IP address locations with generic characters. This procedure is known as **de-parameterization**. In their experiment, Lim et al. (2008) also utilized similar techniques by replacing all recurring numeric characters with generic replacements in an attempt to address the dimensionality problem.

Xu et al. (2009) is a major critic of de-parameterization approach to addressing the dimensionality problem. This technique was criticized because it requires prior knowledge of the log file structure, requires skill,a good understanding of regular expressions (REGEX), it is error prone because important features could also be removed during de-parameterization and if certain parameters aren't captured could lead to an exponential increase in feature dimensionality.

In recent deep learning techniques is gaining traction and have become more prevalent in handing Natural Language Processing (NLP) related tasks. Bertero et al. (2017) took this completely different approach to feature extraction transitioning from traditional NLP techniques to recent advances in NLP by processing huge log files and textual documents as regular texts and transforming those words as a dense vector representation of those words. This technique is known as **word embedding**. This technique is synonymous with representing a man as a person and a king and queen as royalty. Ruder (2016) considers this one of the newest applications of unsupervised learning but do have some drawbacks. Some of the biggest challenges with word embeddings is the inability to handle unknown or out-of-vocabulary (OOV) words and an inability to represent words at sub levels. For example, if a word embedding model hasn't seen a word before, it has no idea of how to interpret such words or how to translate those words into a vector space. In word embeddings, every word is an independent vector. Therefore, words ending with "less" which indicates the absence of something, like "dimensionless" or "frictionless" are treated as independent words.Xu et al. (2009) then proposed a composite key technique which is basically a combination of certain keys in the log messages to produce a composite numerical feature vector.

2.5.2 Anomaly detection

The goal of modeling IT systems is so that predicting future problems reoccurrence and detecting anomalies could be less challenging. The most common technique used by most system administrators and support executives is to manually define a set of strings such as ERROR, WARN or TRACE in an attempt to retrieve lines containing those words. Mariani & Pastore (2008) criticized the impracticality of this approach because of how fast these log files grows and the unstructured format of these log files, the study presented a methodology which depended heavily on pattern-matching and rule based techniques using regular expressions. They are known to perform poorly because it scans through all records in a log file both relevant and irrelevant entries aside from the fact that pattern-matching and regular expressions requires an expert who has got a lot of domain knowledge on script programming.

As pointed out by Sebastiani (2002), this technique is not efficient and less elastic for many reasons. Some of which includes, domain experience requirement of how to develop regular expressions; changes to the log file structure would cause existing failure of existing regular expressions which would ultimately require the development of new rules to cater for recent changes; rule-based techniques is not maintainable in the long-run because an expert with domain experience could end up developing hundreds of these regular expressions to handle the different cases that are present in the log file and lastly, it is time-consuming, time spent developing and writing regular expressions could be spent doing something else. In their recent paper titled "Representation Learning for Resource Usage Prediction", Schmidt et al. (2018) criticized this heuristics based approach because it is a one-size-fit-all solution which is generally applied to all IT systems regardless of the specifics of the situation aside from the fact that it overly simplifies the nature of the task and makes assumptions about the statistics and state of the system.

Recent developments in the field of statistics, data analysis has brought us closer to developing efficient algorithms and models that out perform generic methods which are obviously based on overly simplistic heuristics and assumptions. Over the years, the heuristic based approach has gained less traction with the introduction of machine learning (supervised, unsupervised and semi-supervised). One benefit of this transition is that it requires less

human intervention and is less error prone. Anomaly detection using machine learning techniques enables us to build models for differentiating between normal and abnormal situations (Chandola et al. 2009). One major drawback of this technique is that it assumes that the data size of what constitutes a normal situation must be significantly larger than problematic instances, otherwise the margin of error would increase and the model accuracy would be incorrect.

Chapter 3

Theoretical and Mathematical Aspects

3.1 Introduction

This purpose of this chapter is to provide some theoretical and mathematical underpinnings which lays the ground work for this project.

3.2 Probability and Information Theory

Machine learning deals with uncertain or stochastic(non-deterministic) quantities. Stochasticity and uncertainty arise from various sources. Goodfellow et al. (2016) summarised incomplete information about the system being modeled **incomplete modeling**; the inability to observe all the variables and factors that influence the behaviour of a system **incomplete observability**; and gaps or unknowns within the system being modeled **inherent stochasticity** as the three possible sources of uncertainty in machine learning.

The goal of machine learning is to predict the chance, likelihood or possibility that an event will occur given that similar events had already occurred. Murphy (2012) described machine learning as a set of methods used to predict future occurrences or used to drive the decision making process where a level of uncertainty exists, uncovering and exploiting patterns in a previously examined event.A mathematical framework for representing, quantifying and

deriving uncertainty is known as the probability theory (Goodfellow et al. 2016)

3.3 Z-score Model

Z-score is the amount or number of standard deviations below or above the mean of a given population or sample. It describes the score of a random variable X relative to the mean and standard deviation. A basic z-score formula is:

$$z = \frac{(datapoint - mean)}{stdev}$$

$$z = \frac{(x - \mu)}{\sigma}$$

3.3.1 Importance of Z-scores

- It is used to describe how abnormal or normal a situation within a probability distribution are.
- It compares the score across different distributions.
- It is used to derive the exact location of a score in a probability distribution.

The result of Z-score is signed, positive sign indicates that it is to the right of the mean while a negative sign indicates that it is to the left of the mean.

3.3.2 Applications of Z-scores

Z-scores could be used to derive unknown probabilities from a sample which follows a normal distribution. For example, we could use the z-score to calculate the probability that the running time of a database query extracted from the log files took over 1 minute(60 seconds). This information would be useful to quantify the percentage of slow-performing queries.

Mean	**Std. Dev**	**Data Point (Sec)**
35.51	18.21	60

Table 3.1: Sample data for z-score computation

$$P(X > 60) = \frac{(60 - 35.51)}{18.21} = \frac{24.49}{18.21} = 1.35$$

1.35 from the z-score table occupies an area of 0.9115 which is equivalent to 91.15%
The total area under the bell-shaped curve is 1 or 100%, therefore the unknown fragment of the normal distribution is:

$$100 - 95.15 = 8.85\%$$

This implies that the probability of a database query taking over 60 seconds is 0.0885 or 8.85%

3.4 Euclidean Distance

This is a widely used machine learning technique used for calculating the distance between two data points on a plane. Machine learning models such as K-means and K-nearest Neighbor (KNN) relies heavily on euclidean distance to derive the similarities between two sets of data points. Other metrics for calculating distances exits such as Manhattan, Minkowski and Chebyshev distances but most ML models use Euclidean distance.
The distance between two sets of data points (x,y) and (a,b) is mathematically denoted as:

$$dist((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

3.5 Gaussian Naive Bayes

Gaussian Naive Bayes (GNB) is a supervised learning algorithm which is best suited for addressing the classification problem. This model is based on Bayes probability theorem. Bayes theorem provides a way of calculating posterior probability $P(c|x)$ from $P(c), P(x)$ and $P(x|c)$. This model is fast, efficient, easy to implement and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes' Theorem is stated as:

$$P(h|d) = (P(d|h) * P(h))/P(d)$$

Where

$P(h|d)$ is the probability of hypothesis h given the data d. This is called the posterior probability.

$P(d|h)$ is the probability of data d given that the hypothesis h was true.

$P(h)$ is the probability of hypothesis h being true (regardless of the data). This is called the prior probability of h.

$P(d)$ is the probability of the data (regardless of the hypothesis).

You can see that we are interested in calculating the posterior probability of $P(h|d)$ from the prior probability $p(h)$ with P(D) and $P(d|h)$.

3.6 Bernoulli Naive Bayes

One implementation of the Naive Bayes classifier is the Bernoulli Naive Bayes classifier. It is modeled as a Bernoulli distribution. Bernoulli NB classifier assumes that all the features present in the data frame can be represented as boolean or binary values. Since Bernoulli NB classifier assumes that the data is distributed in accordance to multivariate Bernoulli distribution, this implies that each data point is a binary vector (Shimodaira 2014).

In the experiments and results section I have discussed how this model has been applied during the data transformation phase. The process basically involves reconstructing the raw data input into a Bernoulli vector. The possible outcome for each data point is denoted as

$$x = \{0, 1\}$$

It is worth noting that all Bernoulli distributions are Binomial distributions. In fact Bernoulli distributions are a special case of Binomial distributions where the number of tri-

als $n = 1$

If

$$X = \begin{cases} 1 & \text{with probability } p, \\ 0 & \text{with probability } 1 - p, \end{cases}$$

then the probability distribution of the random variable X is a Bernoulli distribution.

If $X = X_1 + \cdots + X_n$ and each of $X_1, \ldots, X_n$ has a Bernoulli distribution with the same value of p and they are independent, then X has a binomial distribution, and the possible values of X are $\{0, 1, 2, 3, \ldots, n\}$. If $n = 1$ then that binomial distribution is a Bernoulli distribution.

What is a Bernoulli Random Variable?

A Bernoulli random variable is one that has one of two possible outcomes.That is, a random variable X is one that satisfies the condition $P(X = 1) = p$ or it's complement $P(X = 0) = 1 - p$

3.7 Data File Vectorization

One major challenge with machine learning algorithms is the inability to handle raw textual input. Source files, system, console and web server logs all come in plain text format. They cannot be used in this raw form for analytics and machine learning. Most learning algorithms take input features as numeric vectors. Therefore,conversion of textual documents into vector forms is a required step. The technique used to transform text elements in data sets into numeric vectors is known as **text vectorization**. Bag-of-words and Boolean Existential are some vectorization models which could be used.

3.7.1 Word Count Frequency Approach

This technique is also known as the **bag-of-words** model. It involves counting and representing each word in the document as a vector space. A document-term matrix (DTM) is first constructed for each word. Each row in the DTM corresponds to the data file or raw document while each column corresponds to the terms. In text vectorization we often use terms as opposed to words.

3.7.2 Boolean Existential Approach

This model is used to represents each word in the document as a logical or boolean matrix. The raw input is validated using certain rules and criteria, if that condition is met, then the value is set to 1 otherwise it is assigned the default value of 0. Each cell in the matrix can only hold one value which must either be 0 or 1, these are the only possible entries this matrix can hold.

Chapter 4

Methodology

4.1 Introduction

This chapter details the procedure used to attain the stated objectives. It describes what has been done to address this problems. It starts off by presenting the general framework for analysing textual documents and the various techniques that were used.

4.2 The General Framework for Mining and Analysing data files

Xu (2010) presented a series of steps used mining and analysing console logs. This procedure would form the general framework for analysing log and data files presented in this project. The framework features four steps for mining console logs. These steps includes converting each lines or messages from an unstructured text to a data structure log parsing; converting the data structure to a numeric vector, this step is known as vectorization or feature extraction; applying machine learning techniques to identify patterns, anomalies and clusters machine learning; and an intuitive presentation of the identified patterns using graphs and other visual metrics visualization.

4.3 Data Representation and Preprocessing Phase

Computing a representation for all the data points within a collection is a very essential step in finding anomalous structures in a dataset. The process of collecting information from a text file, for example, the description of a log entry involves a series of steps which are required in order to represent a piece of information in such a way that conveys the exact meaning of the original text. Vectorization also known as numerical representation of textual data is a technique used to establish the similarities between data points. Therefore a good vectorization approach will minimize or reduce the euclidean distance between homogeneous data points and maximize or increase the distance between heterogeneous data points.

In this project three different numerical representations has been adopted. Each of these numerical representations seeks to group log file events and then represents each of those events and data points. Firstly, the log file is sequentially read line by line and stored in a collection. A collection is basically an in-memory data structure. This collection is called a sliding window.

4.3.1 TF-IDF Numerical Representation

Term Frequency Inverse Document Frequency (TF-IDF) is a very famous technique for text to number transformations. TF-IDF represents textual data as a multidimensional multinomial matrix. It is divided into two phases each consisting of several stages. The first phase is called the term frequency phase while the other is known as the inverse document frequency phase. During the first phase, the entire document is loaded and then it constructs a mapping of each term to the occurrence of that term in the entire document this is done during the first preprocessing step. The second step, involves computing the tf-idf values for each event in the log file. The result of this step is an extracted feature which is one dimensional in size and holding a value for each term. Behind the scenes, TF-IDF uses a nearness distance measure known as euclidean distance to calculate which events are similar and which are anomalous. This transformation technique has been widely used in natural language processing (NLP) for information retrieval from extremely large corpus of textual data. TF-IDF main idea is to determine the word/term frequencies. Terms that repeatedly occur in a collection of

documents have lower scores while those that occur rather infrequently are assigned higher scores. By doing this, terms with higher scores can easily be differentiated from those with lower scores. This makes it possible to isolate anomalous entries and fit for classification.

The second phase is the inverse document frequency phase. The main idea behind this procedure is to compute the rareness of a term relative to other terms within an entire collection.

4.3.2 Hashing Vectorizer

The benefits counts and term frequency vectorization techniques cannot be overemphasized but comes with a number of limitations. One draw back of this approach is that the list of vocabulary may become very large for extremely large document corpus. This situation will require lots of large vectors for term and document encoding which will ultimately lead to an increase in memory requirements and could potentially impede the performance of the algorithm or in some cases may crash the application.

A work around for this is to hash the terms. Hashing is a technique used to convert strings and texts to numeric representations. The main idea behind hashing is that a term is converted into an integer which is then fed into the machine learning algorithm. One disadvantage of this approach is is that it is a one-way hash, this means that when a text is converted into an hash representation it cannot be reversed back to it's original text.

Chapter 5

Results

5.1 Introduction

This chapter presents results from the performed experiments. A successful log analyzer identify known and unknown anomalies in a data and log file by observing patterns using machine learning techniques. The result from improving site design and user experience by analysing the hit rate is presented first, K-means data clustering techniques used to predict system performance based on the running time and number of records processed is presented second, followed by detecting anomalies in data files using rule and decision based techniques is presented after.

5.2 Analysing server log files for hit rate and usage patterns

Analysing server log files is an important business metric which could be leveraged to improve site design and ultimately user experience. Two sets of experiments were conducted in this section. The first set of experiment is counting the hit rate, that is, the number of times an end user visited a page. The second set of experiment is understanding the hit rate pattern per day to determine what constitutes a normal pattern and what is outside the scope of normal. This experiment was conducted using K-means clustering and time series algorithm. Understanding this behaviour is very crucial to the business because abnormal drops beyond

a normal pattern is a red flag which indicates a deeper problem.

5.3 Experiment 1: Counting web module hit rate per day

5.3.1 Procedure

The following procedures were adopted.

1. Data cleansing: This procedure is crucial to the experiment. It is part of the preprocessing phase which was discussed in the methodology section. It basically involves discarding unwanted entries in the log file while preserving entries which are important to the given analysis.

2. Classification: At this phase similar entries in the log file are grouped and aggregated by functionality which are also known as modules. For example, a log entry such as **http://192.168.0.20:8080?module=getManualSmsFileFormat** and a similar like such as **http://192.168.0.20:8080?module=insertSmsStaging** in principle refers to the same thing. It conveys the fact that the end user is trying to send an SMS to a customer.

3. Load to database: This phase basically aims at improving performance. Reading millions of log entries line by line is a time consuming task which requires memory. Persisting to a database will ultimately improve performance by way of indexes when those entries are required for subsequent processing.

5.3.2 Result

Module	**Hit rate**
Phone calls	6000
Promise To Pay	4230
SMS	3010

Table 5.1: Daily hit rate across the top modules

5.3.3 Analysis

The result for experiment 1, Counting web module hit rate per day are presented in table 5.1. The results shows that phone calls have the highest hit rate of 6000 while promise to pay (PTP) module has the second highest hit rate of 4230 followed by SMS with a hit rate of 3010. This results suggests that more end users are visiting the call module and making phone calls to customers more than any other functionality. The result of this experiment is used for clustering during the second experiment to understand application or module usage pattern per day. This is an important business metric which could be used to identify module usage drops beyond a certain threshold.

5.4 Experiment 2: Understanding usage pattern by using time-series

5.4.1 Procedure

We have adopted the same procedure as those used in experiment 1 with additional steps.

1. Summarizing and aggregating daily hit rate: This procedure basically involves compiling summaries of the daily hit rate and then persisting that to a database. Five database table objects were used for the 5 different modules which were examined. This makes it easy to study each module independently.

2. Clustering: This was conducted using a machine learning algorithm known as K-means. This algorithm will naturally create clusters which are groups of data points with some sort of similarity. Domain experience is used to identify which group of points are outside the scope of normal. Those are referred to as outliers.

5.4.2 Result

The data file used for this analysis consists of two columns, the first column is the day in which the call was made and the second column is the total number of calls made per day while the last column is the probability density function which is basically the dependent

variable p(x). The raw data set is uploaded to my github repository, you could retrieve it using this "https://github.com/joshluisaac/phd-thesis/tree/master/datafiles". The results for detecting drops in the daily number of calls made is presented in table 5.2 and illustrated in figure 5.1 and 5.2. The table is a summary of the mean and standard deviation derived from the data set.

Mean	**Standard Deviation**
4911.72	767.76

Table 5.2: Summarized mean and standard deviation for call made over 1000 days

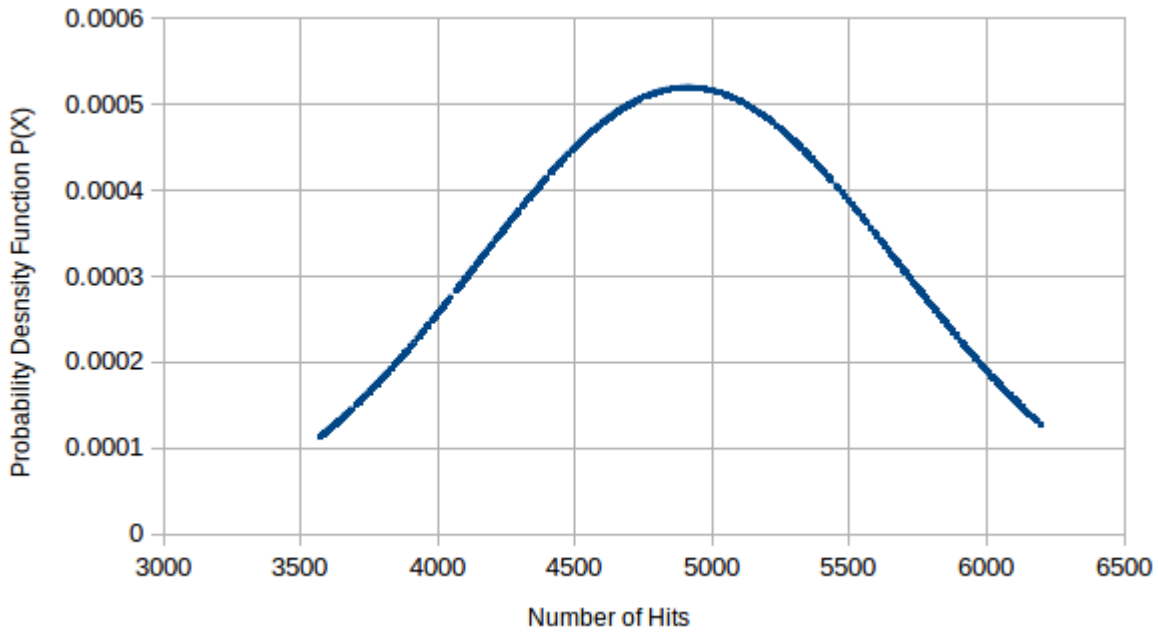

Figure 5.1: Normal distribution plot of the number of hits for calls made

5.4.3 Analysis

Plotted on the x-axis of figure 5.1 is the number of calls made per day while on the y-axis is the probability density function (PDF) which is a dependent variable which responds to changes in X. It is evident from figure 5.1 that it follows a normal distribution with a mean of 4911.72 and a standard deviation of 767.76. Total number of calls 1 standard deviation to the left of the mean is considered below the mean that situation is considered bad anything below 2 standard deviations to the left of the mean is considered extreme. Figure 5.2 shows

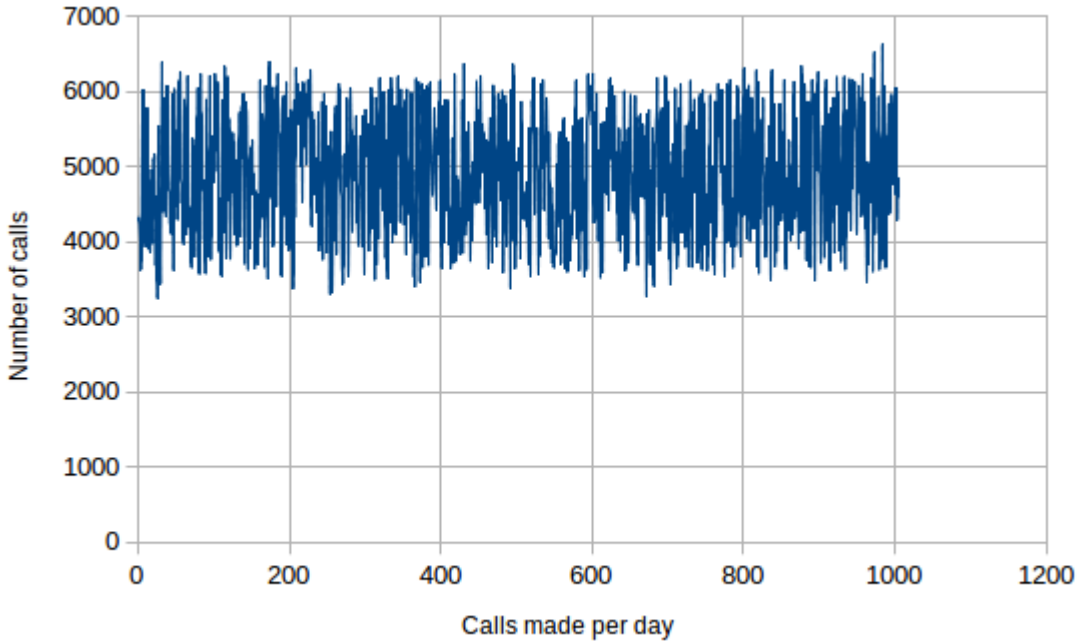

Figure 5.2: Number of hits to call module as a time series plot

the number of calls made per day over the past 1000 days as a time-series plot. A time-series plot is used for predicting and forecasting future events based on a previously observed pattern. It is evident that there were sharp drops in the total number of calls in between day 0 and day 50. These are considered outliers or extreme situation which is obviously 2 standard deviations to the left of the mean.

5.5 Experiment 3: Workload predication

The workload prediction problem comprises of ETL application workload and disk space availability predictions. The results from each of these experiments are presented in the subsequent sections. These predictions were framed as a supervised learning regression problem for the following reasons.

- All the variables are continuous numeric and not category.
- There is a linear relationship between predictors and predictants or input and response.
- All the predictors are independent of each other.
- Data is free of missing values and outliers.

5.5.1 ETL workload prediction

5.5.1.1 Aims

The aim of this experiment is to test whether supervised regression models could be used to predict the future performance of an application based on continuous input variables.

5.5.1.2 Result

The results for predicting the future performance of the ETL application is presented in Tables 5.3, 5.4 & 5.5 and illustrated in Figures 5.3 and 5.4. Plotted on the x-axis is the running time and on the y-axis is the through put per second. The y-axis is the output/response we want to predict.

Metric	**Payload**	**RunningTime**	**Throughput/sec**
count	287.000000	287.000000	287.000000
mean	89255.627178	52.613240	1827.247387
std	3692.565405	12.841975	563.078152
min	83076.000000	30.000000	1099.000000
25%	85222.000000	40.000000	1427.000000
50%	90167.000000	56.000000	1547.000000
75%	92334.000000	63.000000	2296.000000
max	93655.000000	82.000000	3100.000000

Table 5.3: Workload prediction statistical metrics

Metric	**Value**
Mean absolute error (MAE):	3.21106048906
Mean squared error (MAE):	15.9477899498
Root mean squared error (RMSE):	3.99346841102
Constant/Intercept	92.54785537643912
Slope/coefficient	-0.02192117

Table 5.4: Model evaluation and accuracy

$$Y = \beta o + \beta 1X + \epsilon$$

$$Y = -0.022x + 92.55 + 3.99$$

Univariate linear regression model for future workload prediction

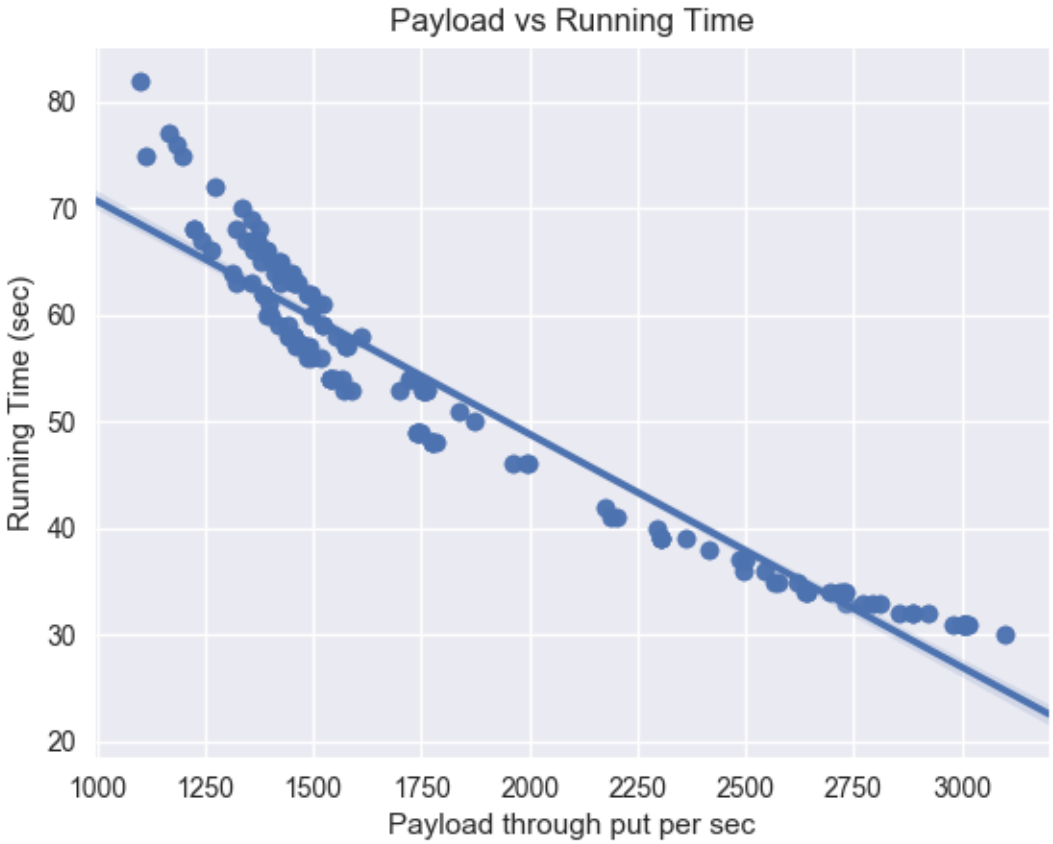

Figure 5.3: Workload prediction on actual values

No	Actual	Predicted	Through put per sec
0	54	58.263138	1564
1	67	62.537767	1369
2	62	62.230871	1383
3	31	27.266597	2978
4	38	39.564376	2417
5	60	61.792447	1403
6	39	42.041469	2304

Table 5.5: Actual and predicted workload

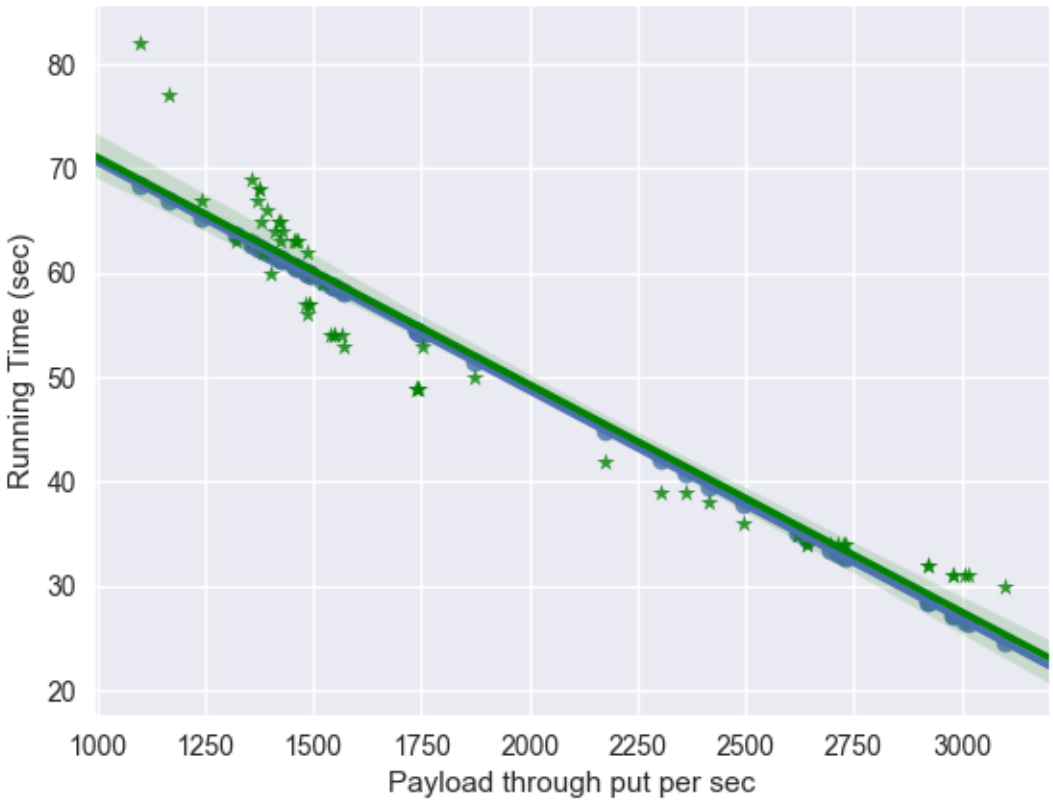

Figure 5.4: Graph of actual and predicted workload

5.5.1.3 Analysis

We can see from the results that the mean values for the payload, total running time and through put/sec is calculated as 89255.63, 52.61 and 1827.25 respectively. This values represent the an average across the observed sample size of 287 records which was used for the test and prediction sample. A standard deviation of 3692.57, 12.84 and 563.08 was also obtained for each of these features respectively.

5.5.1.4 Model Evaluation

The accuracy of the model was evaluated using Root Mean Squared Error (RMSE), Mean Absolute Error (MAE) and Mean Squared Error (MSE). The RMSE value of $3.993 \approx 4$. This implies that our error is less than 7.6 8% of the observed mean value of 52.61. A less than 10% error is evidence of a good regression model.

5.6 Experiment 4: Anomaly detection

The anomaly detection problem was formulated as a classification problem. It falls into a supervised classification problem because the task is to predict which class a new data point belongs. Gaussian Naive Bayes (GNB) classifier and support vector machines (SVM) were used to predict the class/label of a new data point. The data set labels are grouped into the following 5 categories;

- Valid: If an entry is valid, this requires no further investigation.
- Missing-Data: If an entry is missing.
- Missing-Primary-Key: Primary key is missing.
- Invalid: If an entry is invalid and will be rejected.
- Invalid-Data-Load-Default: Invalid but defaults would be loaded.

5.6.1 Classification problem

5.6.1.1 Aims

This experiment aims to test if GNB and SVM can be applied to solve classification problems when trying to detect anomalous entries in a data set.

5.6.1.2 Result

The results for predicting which class an entry in a data set falls into is presented in Tables 5.6, 5.7 and 5.8

5.6.1.3 Analysis

The training set presents a couple of features which includes customerId, customerName, InvoiceDate, InvoiceStatus and label. The label colmun is what we are trying to predict. This column can hold one of the following values; Valid, Missing-Data, Invalid-Data, Missing-Data-PK, Invalid-Data-Load-Default. The other columns can only hold one of [0,1,2]. “0”

Metric	CustomerIdPKEY	CustomerNameOPT	InvoiceDateNN	InvoiceStatusOPT
count	78.000000	78.000000	78.000000	78.000000
mean	1.076923	0.923077	0.923077	0.923077
std	0.619368	0.833849	0.734480	0.833849
min	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	0.000000	0.000000
50%	1.000000	1.000000	1.000000	1.000000
75%	1.000000	2.000000	1.000000	2.000000
max	2.000000	2.000000	2.000000	2.000000

Table 5.6: Classification prediction statistical metrics

Classification Model	Accuracy	Accuracy
Gaussian Naive Bayes	0.92	92.30%
Support Vector Machine	0.73	73%

Table 5.7: Accuracy for the different classifiers

Data point	Classification Model	Label
[2,0,0,1]	Gaussian Naive Bayes	Invalid-Data
[1,1,1,1]	Gaussian Naive Bayes	Valid
[2,0,0,1]	Support Vector Machine	Invalid-Data
[1,1,1,1]	Support Vector Machine	Invalid-Data

Table 5.8: Classification of unseen data point

implies that the entry is missing, “1” implies that the record is present while “2” denotes that the entry is invalid. There are certain rules in place depending on the column specification which is basically the data type and NULL constraints. The individual values of each column is combined with the column specification to deduce the correct class or label.

Primary key field rules

A primary key field by definition cannot hold empty, NULL and invalid entries. Therefore a 0 or 2 is considered a rejected entry. Primary key fields have the suffix “PKEY” to their feature name.

Optional field rules

An optional column has got the suffix “OPT” to it’s feature name. These fields could hold empty or NULL values by definition since they are optional. Invalid and bad entries will also be accepted but would be defaulted to the system default value depending on the data

type specification. For example, an optional numeric field containing the value $24X7$ is by default an anomalous entry but given that this is an optional field it would be replaced with the system numeric default value. The default value for numeric field types is 0 while the default value for variable character types is NULL.

Null constrained fields rules

As the name suggests null constrained fields cannot hold null or invalid entries. This is a very strict type and as such the entry is required and must be valid. A value of 0 or 2 is considered invalid and would be rejected.

The data set was randomly sliced into test and train sets using a *test_size* parameter which was set to 0.33. This means that 33% of the original data is allocated to the test set while the remaining 67% is assigned to the training set. Table 5.8 is the unknown data point we are trying to predict. After training our model we can use it to make meaningful predictions.

5.6.1.4 Model Evaluation

Table 5.7 presents the accuracy score using both classifiers. The accuracy of both models was estimated using sklearn accuracy score. The Gaussian classification model has a better accuracy score compared to support vector machine. The Gaussian model had a 90% chance of getting the prediction correct as opposed to SVM which is only correct 70% of the time.

Chapter 6

Discussion and Conclusion

6.1 Discussion

The presented results in the previous chapters seem to suggest that anomaly detection and workload prediction is viable using machine learning techniques. I presented results from a series of experiments conducted in an attempt to test a formulated hypothesis. A total of five experiments were conducted all exploiting machine learning models. End user affinity and understanding the user's behaviour was formulated as a time-series problem, workload prediction was framed as a supervised univariate regression problem while anomaly detection in data files was formulated as a supervised classification problem in which I explored two classification models, Gaussian classifier and support vector machines. 33% of the total data set was allocated to the test set, the results seem to suggest that Gaussian classifier was more accurate than SVMs when dealing with a multinomial feature vector.

6.2 Conclusion

For a fact log files contain an immense amount of information for system monitoring and troubleshooting but aren't sufficiently utilized by system administrators. In this project I have presented of automatically processing log and data files in an attempt to identify patterns, anomalies and weird behaviours. The two biggest challenges working with log and data files is the size and unstructured nature of these files. These two challenges were addressed in

the methodology and experimentation sections during the data cleansing and preprocessing phases. This phase basically involves parsing and extracting useful content. The result of this is a significant reduction in the amount of data that is required for subsequent phases. Just as Xu (2010) iterated, analysing data and log files is a very practical business problem that shows up everyday in organizations that rely heavily on IT and computing systems. The data set used for this project all came from real-world production systems with sizes ranging from megabytes to gigabytes. My experiments reveal that system administrators and IT support executives can gain deeper and meaningful insights by using machine learning techniques to analyse log and data files which seem impossible for humans. By exploiting the intrinsic structures in log and data, this work suggests that we could turn data and log files into reliant and accurate monitoring systems for making future predictions and anomaly detection, thereby supporting business decisions.

Bibliography

Bertero, C., Roy, M., Sauvanaud, C. & Trédan, G. (2017), Experience report: Log mining using natural language processing and application to anomaly detection, *in* '28th International Symposium on Software Reliability Engineering (ISSRE 2017)', p. 10p.

Chandola, V., Banerjee, A. & Kumar, V. (2009), 'Anomaly detection: A survey', *ACM computing surveys (CSUR)* **41**(3), 15.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. (2016), *Deep learning*, Vol. 1, MIT press Cambridge.

Lim, C., Singh, N. & Yajnik, S. (2008), A log mining approach to failure analysis of enterprise telephony systems, *in* 'Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on', IEEE, pp. 398–403.

Ma, P. (2003), 'Log analysis-based intrusion detection via unsupervised learning', *School of Informatics University of Edinburgh* pp. 1–70.

Mariani, L. & Pastore, F. (2008), Automated identification of failure causes in system logs, *in* 'Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on', IEEE, pp. 117–126.

Murphy, K. (2012), *Machine learning, a probabilistic perspective*, The MIT Press.

Oliner, A., Ganapathi, A. & Xu, W. (2012), 'Advances and challenges in log analysis', *Communications of the ACM* **55**(2), 55–61.

Ruder, S. (2016), 'An overview of gradient descent optimization algorithms', *arXiv preprint arXiv:1609.04747* .

Saneifar, H., Bonniol, S., Laurent, A., Poncelet, P. & Roche, M. (2009), Mining for relevant terms from log files, *in* 'KDIR'09: International Conference on Knowledge Discovery and Information Retrieval', pp. 77–84.

Schmidt, F., Niepert, M. & Huici, F. (2018), 'Representation learning for resource usage prediction', *arXiv preprint arXiv:1802.00673* .

Sebastiani, F. (2002), 'Machine learning in automated text categorization', *ACM computing surveys (CSUR)* **34**(1), 1–47.

Shimodaira, H. (2014), 'Text classification using naive bayes', *Learning and Data Note* **7**, 1–9.

Xu, W. (2010), *System problem detection by mining console logs*, University of California, Berkeley.

Xu, W., Huang, L., Fox, A., Patterson, D. & Jordan, M. I. (2009), Detecting large-scale system problems by mining console logs, *in* 'Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles', ACM, pp. 117–132.

Zheng, Z., Lan, Z., Park, B. H. & Geist, A. (2009), System log pre-processing to improve failure prediction, *in* 'Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference on', IEEE, pp. 572–577.

Zwietasch, T. (2014), Detecting anomalies in system log files using machine learning techniques, B.S. thesis.