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

Joshua Uzochukwu Nwankwo

April 2018

Abstract

The following examples show how to produce Harvard style referencing using biblatex.

Table of Contents

1 Introduction **3**
- 1.1 Overview . . . 3
- 1.2 Motivation . . . 3
- 1.3 Project Objectives . . . 3
- 1.4 Thesis organisation . . . 3

2 Background **4**

3 Theoretical and Mathematical Aspects **5**
- 3.1 Introduction . . . 5
- 3.2 Probability and Information Theory . . . 5
- 3.3 Z-score Model . . . 5
  - 3.3.1 Importance of Z-scores . . . 5
  - 3.3.2 Applications of Z-scores . . . 6
- 3.4 Machine Learning Algorithms . . . 6
- 3.5 K-means Clustering . . . 6
  - 3.5.1 Euclidean Distance . . . 6
- 3.6 Naive Bayes Classifier . . . 6
  - 3.6.1 Gaussian Naive Bayes . . . 6
  - 3.6.2 Bernoulli Naive Bayes . . . 6
  - 3.6.3 Multinomial Naive Bayes . . . 7
- 3.7 Decision Trees . . . 7
- 3.8 Selection of Features . . . 7
- 3.9 Data File Vectorization . . . 7
  - 3.9.1 Word Count Frequency Approach . . . 7
  - 3.9.2 Boolean Existential Approach . . . 8
  - 3.9.3 Fuzzy Relevance Matrix . . . 8
- 3.10 Summary . . . 8

4 Methodology **9**
- 4.1 Introduction . . . 9
- 4.2 The General Framework for Mining and Analysing data files . . . 9
- 4.3 Data Representation and Preprocessing Phase . . . 9
  - 4.3.1 TF-IDF Numerical Representation . . . 10
  - 4.3.2 Hashing Vectorizer . . . 10

5 System Design and Implementation **11**
- 5.1 Log File Preprocessing . . . 11

6 Results **12**
6.1 Introduction . . . 12
6.2 Analysing server log files for hit rate and usage patterns . . . 12
6.3 Experiment 1: Counting web module hit rate per day . . . 12
6.3.1 Procedure . . . 12
6.3.2 Result . . . 13
6.3.3 Analysis . . . 13
6.4 Experiment 2: Understanding usage pattern by applying K-means clustering algorithm . . . 13
6.4.1 Procedure . . . 13
6.4.2 Result . . . 14
6.4.3 Analysis . . . 14
6.5 Experiment 3: Workload predication . . . 15
6.5.1 ETL workload prediction . . . 15
6.5.1.1 Aims . . . 15
6.5.1.2 Procedure . . . 15
6.5.1.3 Result . . . 15
6.5.1.4 Analysis . . . 18
6.5.1.5 Model Evaluation . . . 18
6.6 Experiment 4: Anomaly detection . . . 18
6.6.1 Classification problem . . . 19
6.6.1.1 Aims . . . 19
6.6.1.2 Procedure . . . 19
6.6.1.3 Result . . . 19
6.6.1.4 Analysis . . . 20
6.6.1.5 Model Evaluation . . . 20

7 Conclusion and Discussion **21**

Chapter 1

Introduction

1.1 Overview

1.2 Motivation

1.3 Project Objectives

1.4 Thesis organisation

Chapter 2

Background

Chapter 3

Theoretical and Mathematical Aspects

3.1 Introduction

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 Machine Learning Algorithms

3.5 K-means Clustering

3.5.1 Euclidean Distance

3.6 Naive Bayes Classifier

3.6.1 Gaussian Naive Bayes

3.6.2 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 implementation and methodology 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 trials $n = 1$

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 $P(X = 0) = 1 - p$

3.6.3 Multinomial Naive Bayes

3.7 Decision Trees

3.8 Selection of Features

3.9 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, Boolean Existential and Fuzzy Relevance Matrix are some vectorization models which could be used.

3.9.1 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.

3.9.2 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.

3.9.3 Fuzzy Relevance Matrix

This project adopted xxxx and xxx vectorization approach

3.10 Summary

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 various machine learning algorithms that were used during the course of this project, it also justifies the rationale behind some of the adopted design decisions. This chapter elucidates the evaluation techniques used to validate the accuracy and correctness of the applied machine learning algorithms.

4.2 The General Framework for Mining and Analysing data files

Ref 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 4 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

System Design and Implementation

5.1 Log File Preprocessing

Chapter 6

Results

6.1 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.

6.2 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.

6.3 Experiment 1: Counting web module hit rate per day

6.3.1 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.

6.3.2 Result

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

Table 6.1: Daily hit rate across the top modules

6.3.3 Analysis

The result for experiment 1, Counting web module hit rate per day are presented in table 6.1 and illustrated in figure XXX. 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.

6.4 Experiment 2: Understanding usage pattern by applying K-means clustering algorithm

6.4.1 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.

6.4.2 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 link. The results for detecting drops in the daily number of calls made is presented in table 6.2 and illustrated in figure 6.1 and 6.2. The table is a summary of the mean and standard deviation derived from the data set.

Mean	Standard Deviation
4911.72	767.76

Table 6.2: Summarized mean and standard deviation for call made over 1000 days

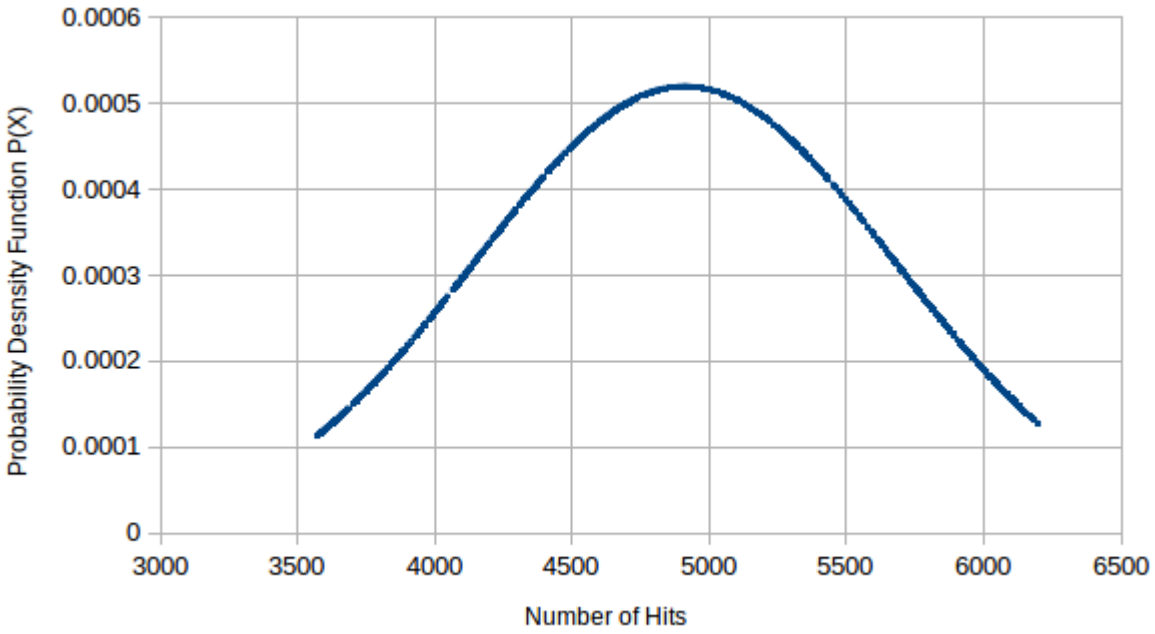

Figure 6.1: Normal distribution plot of the number of hits for calls made

6.4.3 Analysis

Plotted on the x-axis of figure 6.1 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 6.1 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 6.2 shows 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.

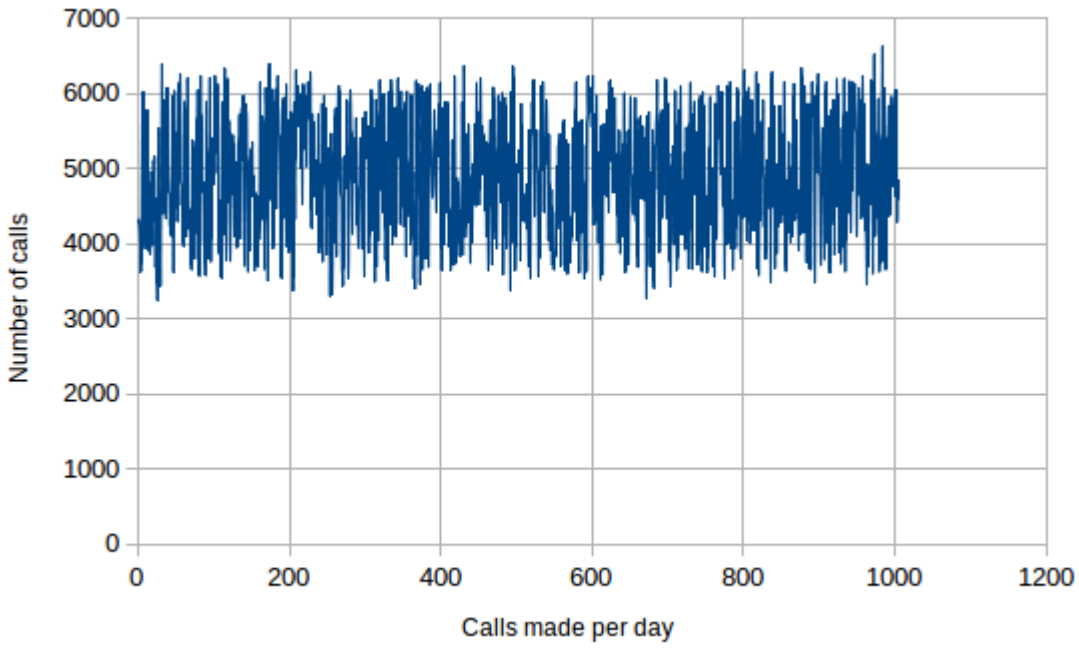

Figure 6.2: Number of hits to call module as a time series plot

6.5 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.

6.5.1 ETL workload prediction

6.5.1.1 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.

6.5.1.2 Procedure

-

6.5.1.3 Result

The results for predicting the future performance of the ETL application is presented in Tables 6.3, 6.4 & 6.5 and illustrated in Figures 6.3 and 6.4. 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 6.3: 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 6.4: 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

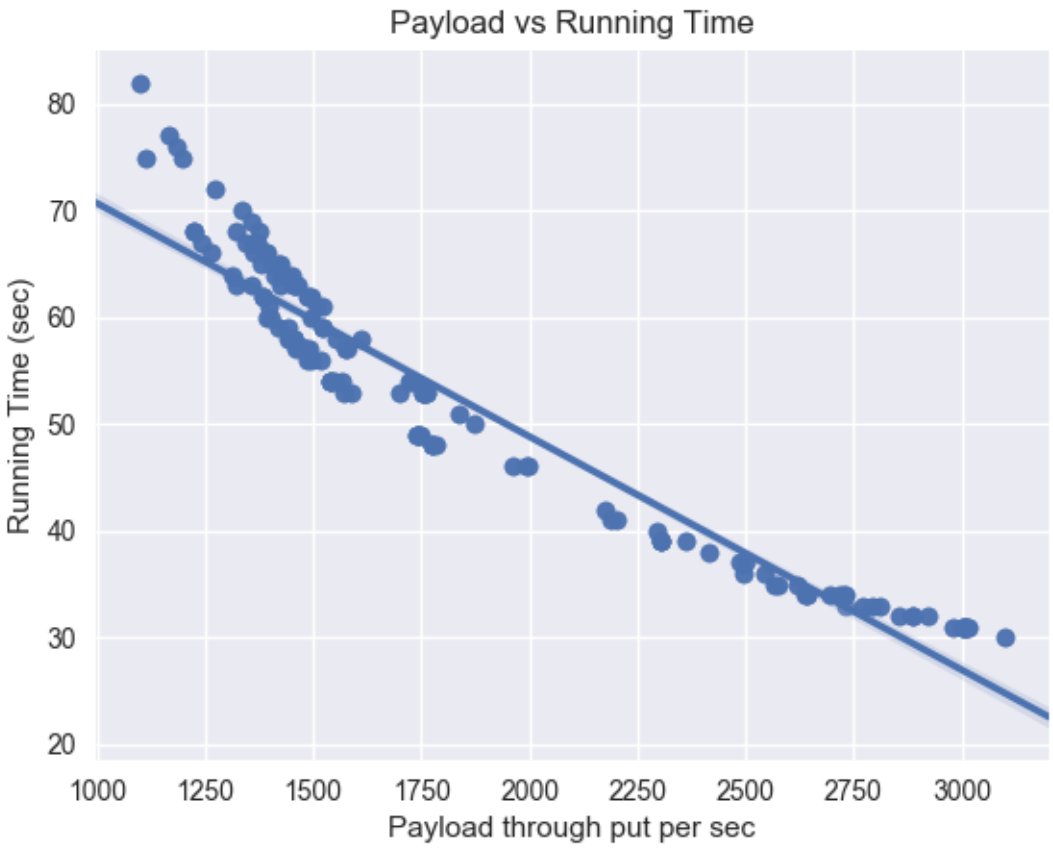

Figure 6.3: Workload prediction on actual values

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 6.5: Actual and predicted workload

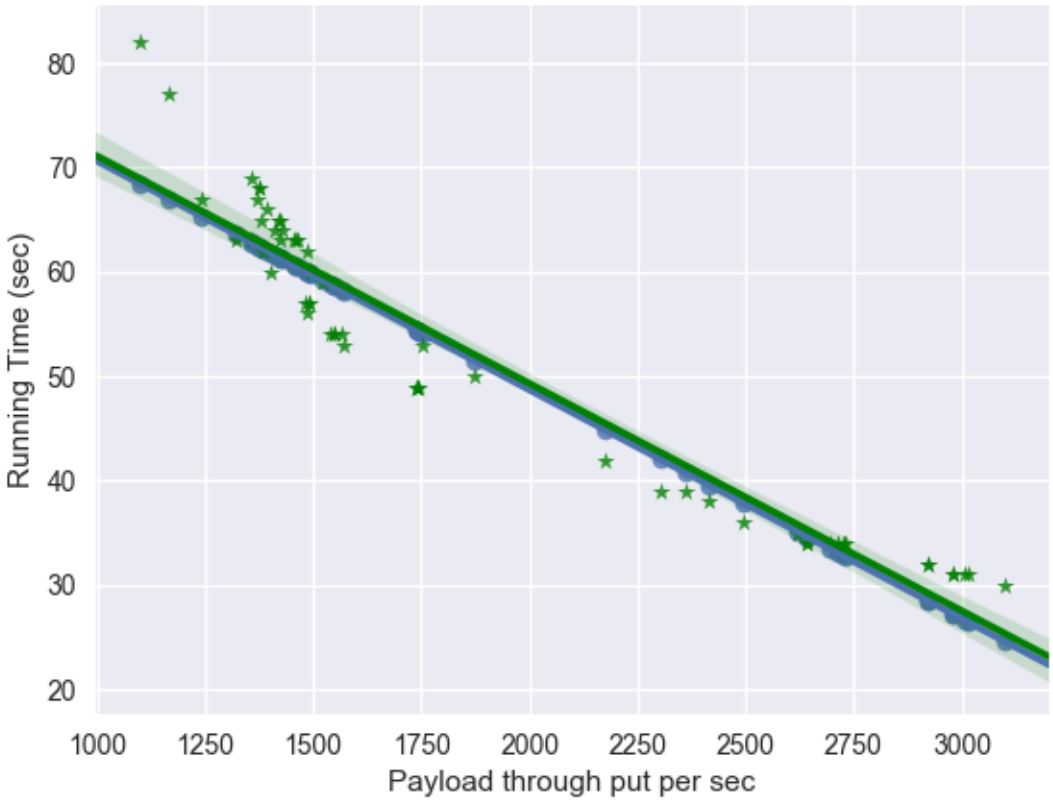

Figure 6.4: Graph of actual and predicted workload

6.5.1.4 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.

6.5.1.5 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.

6.6 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.

6.6.1 Classification problem

6.6.1.1 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.

6.6.1.2 Procedure

6.6.1.3 Result

The results for predicting which class an entry in a data set falls into is presented in Tables 6.6, 6.7 and 6.8

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 6.6: Classification prediction statistical metrics

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

Table 6.7: 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 6.8: Classification of unseen data point

6.6.1.4 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" 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 6.8 is the unknown data point we are trying to predict. After training our model we can use it to make meaningful predictions.

6.6.1.5 Model Evaluation

Table 6.7 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 7

Conclusion and Discussion

Bibliography

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

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

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