A Pattern-Driven Corpus to Predictive Analytics in Mitigating SQL Injection Attack

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS OF EDINBURGH NAPIER UNIVERSITY,

FOR THE AWARD OF DOCTOR OF PHILOSOPHY

IN THE FACULTY OF ENGINEERING, COMPUTING & CREATIVE INDUSTRIES

Solomon Ogbomon Uwagbole

September 2018

Declaration

This thesis is the result of my work and includes nothing, which is the outcome of work done in collaboration except where explicitly indicated in the text. It has not been previously submitted, in part or whole, to any university or institution for any degree, diploma, or other qualification.

Signed: Solomon Ogbomon Uwagbole

Date:

Copyright

Acknowledgements

Balancing PhD research with work commitments is a challenging combination which I could not have done without the support of my families, supervisors and Napier University. My motto to any undertaking is small incremental ideas are building blocks to a big idea; this remains my inner driving force throughout this PhD journey.

I thank my mother and father, Mrs Agnes Etusi Ogbomon and late Mr Paul Ogbomon Uwagbole who have worked hard to implant the value of education in me and my siblings. The entire family of Paul Ogbomon Uwagbole, grandparents late Mrs Fatima Imongirie and Pa Agene Akhuemonkhan. Beloved Uncles late Friday Usiholo Imongirie and Clement Agene Akhuemonkhan, RIP. My profound thanks to Ambassador Amerib-Okun Okoedion and family, including Dr. Frederick Okoedion and family for their role model and assistance.

My daily motivation to continue the PhD journey is drawn from the extended family members, including those who are no longer with us. I remembered here my immediate late brother and his son Ojie-amien Innocent Ogbomon Uwagbole and Israel Ogbomon Uwagbole, RIP. My thanks to Mr & Mrs J Ikhine, Josephine, Alenbesunu, Ekihalo, Omontese and Obehi Ogbomon Uwagbole. My profound thanks to Suraya Bte Abdul Hameed for her immeasurable assistance, including those I forgot to mention here.

I am most grateful to my wife Mrs Sarah Uwagbole and our son Isaac Ogbomon Uwagbole who is born during the research journey and have started school for their support. I also use this opportunity to thank Martin & Sue Muncer, not the least for helping with Isaac’s school runs.

My gratitude to the School of Computing, Napier University for the opportunity to allow me to go through this PhD journey. The PhD has been very challenging for me as it was undertaken at my midlife with juggling between existing commitments at work, family and academics.

I could not have come to this day without the support of my supervisors and panel chair. My profound thanks to Professor William Bill Buchanan OBE for his patience including Professor Jessie Kennedy and Dr Lu Fan.

Contents

Tables

Figures

Equations

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AUC	Area Under Curve
AWS	Amazon Machine Learning Web Services
CIA	Confidentiality, Integrity and Availability
CLI	Call-Level Interface
CSIC	Council of Scientific Investigations (Spanish Research Council)
CM	Confusion Matrix
CMA	Computer Misuse Act
CPU	Central Processing Unit
DFA	Deterministic Finite Automata
ECCWS	European Conference on Cyber Warfare and Security
FN	False Negative
FP	False Positive
FPR	False Positive Rate
FSA	Finite State Automaton
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IoT	Internet of Things
LR	Logistic Regression
MAML	Microsoft Azure Machine Learning
ML	Machine Learning
MLR	Multiclass Logistic Regression
MNN	Multiclass Neural Network
NLP	Natural Language Processing
NE	Negative events
NETSQLIA	Numerical Encoding to Tame SQLIA
NFA	Nondeterministic Finite Automata
NN	Neural Network
NO	Negative Observations

OWASP	Open Web Application Security Project
PE	Positive events
PO	Positive observations
RDMS	Relational Database Management System
RegEx	Regular Expression
RFID	Radio-frequency identification
ROC	Receiver Operating Characteristic
SDLC	Software Development Life Cycle
SDN	Software-Defined Network
SFA	Symbolic Finite Automaton
SMOTE	Synthetic Minority Over-Sampling Technique
SMT	Satisfiability Modulo Theories
SQL	Structured Query Language
SQLI	Structured Query Language Injection
SQLIA	Structured Query Language Injection Attack
SQLIV	SQLI Vulnerabilities
SVM	Support Vector Machine
TCLR	Two Class Logistic Regression
TC SVM	Two Class Support Vector Machine
TMH	Tune Model Hyperparameters
TN	True Negative
TP	True Positive
TPR	True Positive Rate
UCI	University of California, Irvine
UML	Unified Modelling Language
URL	Universal Resource Locator
WAF	Web Application Firewalls
WS	Web Service
XSS	Cross Scripting

Abstract

The back-end database provides accessible and structured storage for each web application's big data internet web traffic exchanges stemming from cloud-hosted web applications to the Internet of Things (IoT) smart devices in emerging computing. Structured Query Language Injection Attack (SQLIA) remains an intruder's exploit of choice to steal confidential information from the database of vulnerable front-end web applications with potentially damaging security ramifications.

Existing solutions to SQLIA still follows the on-premise web applications server hosting concept which were primarily developed before the recent challenges of the big data mining and as such lack the functionality and ability to cope with new attack signatures concealed in a large volume of web requests. Also, most organisations' databases and services infrastructure no longer reside on-premise as internet cloud-hosted applications and services are increasingly used which limit existing Structured Query Language Injection (SQLI) detection and prevention approaches that rely on source code scanning. A bio-inspired approach such as Machine Learning (ML) predictive analytics provides functional and scalable mining for big data in the detection and prevention of SQLI in intercepting large volumes of web requests. Unfortunately, lack of availability of robust ready-made data set with patterns and historical data items to train a classifier are issues well known in SQLIA research applying ML in the field of Artificial Intelligence (AI). The purpose-built competition-driven test case data sets are antiquated and not pattern-driven to train a classifier for real-world application. Also, the web application types are so diverse to have an all-purpose generic data set for ML SQLIA mitigation.

This thesis addresses the lack of pattern-driven data set by deriving one to predict SQLIA of any size and proposing a technique to obtain a data set on the fly and break the circle of relying on few outdated competitions-driven data sets which exist are not meant to benchmark real-world SQLIA mitigation. The thesis in its contributions derived pattern-driven data set of related member strings that are used in training a supervised learning model with validation through Receiver Operating Characteristic (ROC) curve and Confusion Matrix (CM) with results of low false positives and negatives. We further the evaluations with cross-validation to have obtained a low variance in accuracy that indicates of a successful trained model using the derived pattern-driven data set capable of generalisation of unknown data in the real-world with reduced biases. Also, we demonstrated a proof of concept with a test application by implementing an ML Predictive Analytics to SQLIA detection and prevention using this pattern-driven data set in a test web application. We observed in the experiments carried out in the course of this thesis, a data set of related member strings can be generated from a web expected input data and SQL tokens, including known SQLI signatures. The data set extraction ontology proposed in this thesis for applied ML in SQLIA mitigation in the context of emerging computing of big data internet, and cloud-hosted services set our proposal apart from existing approaches that were mostly on-premise source code scanning and queries structure comparisons of some sort.

1 Introduction

Figure 1-1 presents the thesis organisation layout with the sections in this chapter.

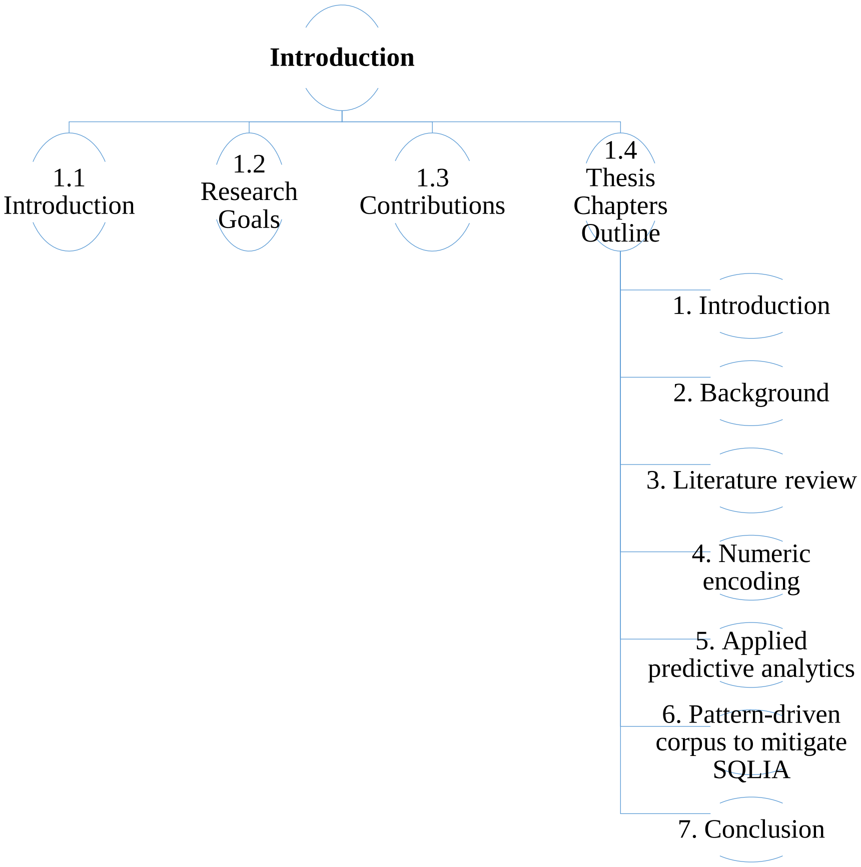


Figure 1-1 Thesis organisation chart: The figure displays the chapter visual representation of the different sections to visualise the layout of the entire thesis.

1.1 Introduction

SQLIA is an intruder exploit of choice for vulnerable web applications to pilfer confidential data from the database with potentially damaging consequences. There are challenges in securing web applications with continuous innovations in internet applications that have seen an astronomical growth of big data emanating from web requests to cloud-hosted web applications and the Internet of Things (IoT). This large volume of web requests in internet traffic needs analyses to intercept spurious web requests destined to a vulnerable web application and associated back-end database. Applying AI techniques provide a solution to detect and prevent SQLIA in big data context, but there are issues of non-availability of pre-existing pattern-driven data set required to train a supervised learning model.

An AI approach can provide scalable data mining for SQLIA detection and prevention. The application of an AI approaches to analysing large volumes of web requests could effectively predict attack signatures by classifying web requests based on the labelled data set. The labelled data set is used to train the classification algorithms to predict SQLIA thereby dropping suspicious web requests before reaching the back-end database to pilfer protected data. However, AI techniques have met with the issue of lacking an existing data set [1], [2]. Throughout this thesis, we refer to corpus as a data set following the University of California, Irvine (UCI) ML repository naming convention [3]. The existing solutions [4], [5], [14], [6]–[13] to mitigate SQLIA were all before emerging computing of cloud-hosted services and the upward trend in big data internet traffic and as such lack the functionality and ability to cope with new attack signatures concealed in web requests.

An intruder is described as an individual or a group that breaches the security of a protected computer system by employing an array of techniques to disrupt and pilfer confidential data. A typical method used to steal sensitive data is by SQLIA and which often results in new signatures stealthily bypassing the protection of Web Application Firewalls (WAF). Back-end database security exploits have thus evolved with the availability of the off-the-shelf tools used by an intruder to automate SQLIA. These tools are often combined by an intruder to provide both penetration testing and exploits in SQLIA with a few notable examples: Burb Suite [15], SQLMap [16], BSQL Hacker [17], SQLninja [18], BSQL Hacker [17] and Metasploit [19].

Automated SQL Injection Vulnerabilities (SQLIV) penetration testing and SQLIA exploit tools like Burb Suite, SQLMap and SQLninja as a few examples, offer the intruder a new method in creating many more derivations of new attack signatures over the existing SQLIA string lookup approach mitigation proposed by researchers over the years [4], [5], [14], [20], [6]–[12]. These existing approaches to mitigate SQLIA existed in an era when web application servers were hosted in an organisation’s own intranet and extranet. These existing solutions to address SQLIA are no longer applicable in emerging computing with a cloud-hosted services’ edge Software-Defined Network (SDN) application endpoints which have broad web request hits across the internet.

SQL Injection (SQLI) is not only a vulnerability arising from developers’ lack of security awareness in web application development to require the input sanitisation but an exploit of the free text processing capability of the SQL engine. This SQL engine design flexibility in free-text processing has ramifications in both legacies, and new web applications which lack sanitisation and so can become SQLI vulnerable.

SQLIV have thus not gone away and continue to feature in the Open Web Application Security Project (OWASP) top list of vulnerabilities [20]. A Google search of ‘SQLi hall of shame’ [21] throws light on how topical SQLIA issues are, even within reputable organisations, with a few examples listed in Table 1-1 and further described below.

SQLIA is still used in many data vulnerabilities and pilfering security attacks, including the Florida County election website which when hacked resulted in usernames and passwords being stolen [22]. The TalkTalk Mobile UK hacks involved a teenager pilfering more than four million customer details that include sensitive information like passwords and credit card numbers [23], [24]. Also, the VTech hacking resulted in the leak of 4.8 million customer transaction details [25]. SQLIV were also found in New Joomla [26] and McAfee ePolicy Orchestrator (ePO) [27]. Equifax, which provides credit report services around the world, was hacked with 145 million Americans personal data compromised [28]. Once hacked, the confidential data pilfered by an intruder is then often traded on the dark web for ransom, blackmailing and financial gains.

Table 1-1 SQLIA and SQLIV examples

Company	Type	Description	Year
TalkTalk Mobile UK	SQLIA	4 million customer details including personal details, passwords and credit card numbers pilfered [23], [24]	2015
VTech	SQLIA	4.8 million customer transaction details pilfered [25]	2015
Symantec	SQLIV	Console vulnerability [29]	2016
Facebook	SQLIV	Employee details vulnerability [30]	2016
McAfee ePolicy Orchestrator)	SQLIV	McAfee ePolicy Orchestrator (ePO) [27]	2017
Joomla CMS web	SQLIV	SQLIV were found in New Joomla [26]	2017
Equifax	SQLIA	145 million Americans personal data compromised [28]	2017
Wordpress	SQLIV	Vulnerability in WordPress plugins that needed to be patched [31]	2017
GoDaddy	SQLIV	Vulnerability detected by researchers [32]	2017
Joomla	SQLIV	Vulnerabilities identified and fixed by Joomla developers [33]	2018
BSNL	SQLIA	Bharat Sanchar Nigam Limited (BSNL) hacks of potentially 47000 plus employees record compromised [34]	2018
Indian Railway	SQLIV	Vulnerabilities of non-sensitive data exposed [35]	2018

The proposed model delivers a self-contained SQLIA detection and prevention solution which is the subject of the empirical evaluation by statistical measures, ROC curve and cross-validations including a comparison with existing SQLIA research over the years. A prototype of the proposed model in Figure 1-2 illustrates the proposed model with the details of the numbered stages provided below:

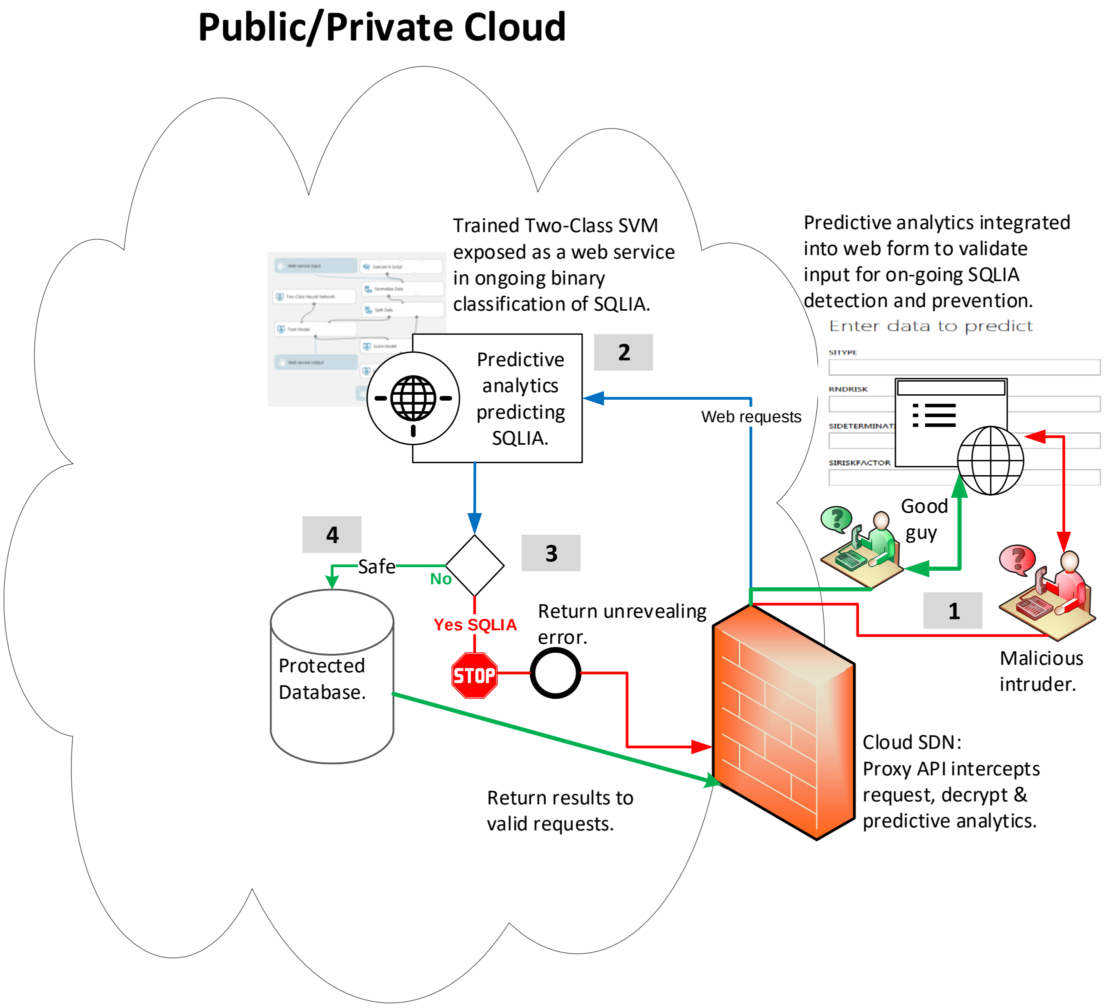

Public/Private Cloud
Trained Two-Class SVM exposed as a web service in ongoing binary classification of SQLIA.
Predictive analytics integrated into web form to validate input for on-going SQLIA detection and prevention.
Enter data to predict
SITYPE
RNDRISK
SIDETERMINAT
SIRISKFACTOR
Predictive analytics predicting SQLIA.
2
Web requests
Good guy
1
Malicious intruder.
4
Safe
No
3
Yes SQLIA
STOP
Return unrevealing error.
Protected Database.
Return results to valid requests.
Cloud SDN: Proxy API intercepts request, decrypt & predictive analytics.

Figure 1-2 A prototype of the proposed model: The figure is illustrating the prototype of the proposed model illustrating the design overview including consuming a trained ML model at the web proxy API and client forms in an ongoing SQLIA detection and prevention.

1. The result of a highly trained ML model is deployed as a web service which provides predictive analytics to validate the data input web form at the front-end. When a protected web form receives an input, it verifies the data to predict SQLI. A legitimate expected data pass the first test to proceed to the server for further validation while the spurious request is dropped as it has failed a simple web form input validation.
2. The second stage is to intercept the query string of the web request by Fiddler Proxy API integrated with a trained ML model web service at the cloud SDN endpoint. These captured web requests are analysed to predict SQLIA as most SQLI by an intruder bypass the front-end web form input to inject the query in

transition to the back-end web application servers with SQLIA type to circumvent the security protection of the application. This stage is followed by the system decision phase using the results of the SQLIA prediction analysis to allow the web request to proceed or not.

3. The third stage inferred from the outcome of the intercepted web request for predictive analytics to fulfil the web request or referred it to the fraud team for a second opinion. Analysed web request results deemed true negative to SQLIA are performed while true positive ones are passed to the human expert for further scrutiny before a cleverly crafted response to avoid an intruder using the error messages to further a more SQLIA.
4. The final stage involves legitimate or expected web requests that are truly negative to SQLIA being fulfilled. Also, this mitigation proposed does not rely on query comparison as seen in existing work as the proposal tracks the input data from the web request form driven by a web service to when it is intercepted in-transition to the back-end database at the proxy API for predictive analytics.

In the scheme presented in this thesis, we explored generating sizeable historical learning data that are pattern-driven as input learning data to train a supervised learning model. We implemented Regular Expression (RegEx) [36], [37] to create patterns and String Package [38] to generate all possible derivations of strings that share the same pattern (members strings) to obtain the learning data. These derived member strings are encoded to numerical vector values required to train AI models. The transformation of derived member strings to numeric vectors sets a precedence for the suitability of applying the feature hashing to the data set generated from member strings to get input matrix discussed in Chapters 5 and 6 in SQLIA mitigation in the context of big data.

This thesis applies predictive analytics to demonstrate a proof of concept as it uses the pattern-driven data set containing vector extraction from expected web requests, known attack patterns including SQL tokens present at the injection points. SQL query comprises of lexical or syntactic units named tokens. The SQL tokens are the query keywords, constants and delimiter identifiers [39]. The data set is preprocessed and labelled for supervised learning. The trained classifier is deployed as a web service that is consumed in a custom .NET application implementing a web proxy API [40] to intercept and accurately predict SQLIA in web requests, thereby preventing malicious web requests from reaching the protected back-end database. The models are empirically evaluated in a MAML studio to validate the performance metrics of the approach presented in this thesis.

Recent years have seen the maturing of applied research in AI applications with the availability of commercial platforms like MAML studio [41], [93]. The Microsoft Azure ML AI platform provides cloud-hosted services and infrastructures to support big data analytics in virtually all application domains. Also, it offers empirical statistical measures validation in the form of the ROC curve, CM and k-fold cross-validation. Thus, the experimental results presented in this thesis are implemented and empirically validated on MAML studio.

The proposed model in this thesis is built on MAML studio [42], and the methodology includes extraction of data set attributes items and labelling, classification of SQLIA features, and validation of the supervised learning model. The model is then exposed as a web service for real-time SQLIA detection and prevention.

MAML studio, which is a cloud-hosted ML Platform as a Service (PaaS) allows data scientist to expand the horizons of AI models from the confines of statistical measures and evaluations to building a predictive analytic model that is highly trained and evaluated that is ready to use in real-life applications. It has the programming models of R and Python scripting which allows data scientist to leverage an existing body of knowledge on AI algorithms in these programming languages.

MAML has been seen as a game-changer by data scientist, practitioners applying to business applications in its features offering, including supporting the traditional languages such as R and Python scripting [43]. Also, the simplicity and the quick turnaround from start to finish of building a ready to deploy trained model has made it the top AI platform among the fifteen widely used AI commercial platforms [44].

The MAML studio comes equipped with inbuilt ML standard validation modules of statistical measures [45]–[47], including the ROC curve, CM and cross-validation widely accepted and used in data science to measure the performance of a trained ML model [48]–[51]. The cross-validation module provides a statistical measure of which a low standard deviation or variance in accuracy indicates of a trained model with reduced biases capable of generalisation of unknown data to the classifier in a real-world application [47]. The MAML studio follows the favoured *k*-fold cross-validation *k =10* [48], [49], [52].

The MAML studio provides an established validation model of statistical measures, including the ROC curve and CM. The ROC curve is a graph plot of recall or True Positive Rate (TPR) against False Positive Rate (FPR). It demonstrates the trade-off between sensitivity (TPR) and specificity (FPR) shifts at various thresholds as an increase in one will result in the decrease of the other in the x and y-axis of the graph. A curve towards hugging the top of the y-axis indicates a good performance while towards the x-axis indicates a poor performance [53]. A CM is a favourite in multiclass classification that presents a table containing the classifier performance metrics results of the testing data observations at various thresholds [54]. The ROC curve and CM are widely used in clinical studies for validating medical diagnostic results and have recently found their way into data sciences in the empirical evaluation of classifiers or trained AI models [46], [51], [55]. The MAML studio provides a one-stop solution to these AI principles from procuring data set for training and validating of a trained model including cross-validation.

1.2 Research Goals

In applying ML to SQLIA problems, there is a need for a data set. This thesis proposes the following research questions:

- Can patterns in both web expected requests and SQL tokens including existing SQLIA signatures extraction be used to create a large volume of data set of encoded numeric vector variables required to train a supervised learning model?
- Can the pattern-driven data set be validated?
- Can a web application type be protected produce the artefact for the extraction of a pattern-driven data set?

These research questions are answered in the following contributions:

- The ability to derive a pattern-driven data set to predict SQLIA of any size.
- The training of a supervised learning model using the pattern-driven data set with validation through a ROC curve, CM and cross-validation.
- Providing applied ML predictive analytics to SQLIA detection and prevention using a pattern-driven data set from a web application type domain’s data context.

The results of experiments conducted in this research demonstrate patterns exist in web input data in both legacy and new web applications that can be leveraged to generate as many derivations of member strings (strings that share the same pattern). In applying ML techniques that require data sets with sufficient learning data arising from these patterns, we thus explore Finite State Automaton (FSA) states walk [56]–[58] to generate learning data from these patterns that exist in the input data to any web application type context.

The success of the pattern-driven numeric encoding of features to obtain extensive learning data that contain vector matrices opens-up further work in string features vectorisation. Vectorisation or hashing [59], [60] is a technique of converting strings to input vector matrices to train a supervised learning model in predictive analytics techniques to predict SQLIA at runtime.

1.3 Contributions

The listed contributions below in a broader scope has significance in resolving the issues researchers currently faced in the field of Intrusion Detection System (IDS) which include SQLI in procuring data set to apply AI techniques in the ever-growing big data. In this research work, we present the following novel contributions:

- We presented an ontology in Chapter 4 and related publications [61], [62] for crafting a pattern-driven data set using R string API from the web application type with a technique to encode the data set into vectors required to train a supervised learning model in MAML studio. We trained a supervised learning classification algorithm with this pattern-driven data set and validated the trained model under various classification algorithms with high performance metrics statistical measures including cross-validation as presented. The success of this conceptual approach in Chapter 4 has led to further work in Chapters 5 and 6 by employing string hashing vectorisation in-place of manual encoding on implementing a proof of concept of how the proposal will be applied in a real-world web application.
- We further in Chapter 5 and related publication [63] the numeric encoding of features presented in Chapter 4 with hashing vectorisation to obtain vector matrices to train the classification algorithms. In answering the research question if the intended web application type can produce the artefact for a pattern-driven data set, we implemented a web application that expects dictionary words as a valid input while elements of

SQL tokens and SQLIA type signatures substitution at the SQLI hotspots is predicted as SQLIA positive. The pattern-driven data set is used to train a supervised learning model employing a TC LR and TC SVM classification algorithms with the better-evaluated and cross-validated classifier selected to predict SQLIA. The selected trained TC SVM model is exposed as a web service which is then deployed to the web form and the cloud SDN proxy for intercepting web requests in-transition to a back-end database for analysis.

- We demonstrated in Chapter 6 and related publication [64] a more robust method of pattern-driven data set procurement based on the web application type; we derived a pattern-driven data set using Finite State Automata (FSA) and Symbolic Finite Automata (SFA) techniques as against R string API technique presented in Chapter 5 to derive related member strings. The referred web application expects dictionary words as a valid input while elements of SQL tokens and SQLIA type substitution predicted at the SQLI hotspots are predicted as SQLIA positive. The pattern-driven data set is used to train a supervised learning model employing a TC LR and TC SVM classification algorithms with the better-evaluated and cross-validated classifier selected to predict SQLIA. The trained TC SVM model is exposed as a web service which is then consumed in a web form for input validation and a proxy API at the cloud SDN for intercepting web request for analysis. We observed using the SFA technique to derive related member strings that we could generate a pattern-driven data set with features of related member strings of any size to train a classifier for SQLIA mitigation for real-world application.

On account of the few existing data sets related to this domain of study were either generated for competition as in ECML/PKDD 2007 [65] and HTTP dataset CSIC 2010 [66]. The HTTP dataset CSIC 2010 was motivated for a need for a replacement to out of date DARPA KDD Cup 1999 [67]. These data sets are repeatedly found in research work as test data for textbook evaluations and have come under scrutiny over the years of being antiquated and irrelevant to building the real-world application as it contains some inconsistencies [67], [68]. Also, though these data sets have provided textbook evaluations of various proposals over the years, it does not offer the features to build real-life ML-based SQLIA mitigation. We propose in this thesis for the artefact to construct a pattern-driven data set for an ML model to be extracted from the web application type being protected in ongoing SQLIA mitigation.

Web-driven applications including business, government, private and purpose-built internet-driven applications are implemented for various needs as to have a standardised data set. Thus, this diversity in web application types is too broad to have one standard all-purpose (universal) generic data set to train a supervised learning model to mitigate SQLIA. The few purpose-built data sets [65], [66] would typically contain unprocessed repeating query strings of features that exist in SQLIA types. Applying data set of repeating strings devoid of patterns in ML supervised learning model results in poor performance metrics [70].

The existing approaches applying AI typically obtained a data set either by having a phase in the implementation where tools are used to simulate attack signatures or with the query string logged into a file as a data set to be used in the training of a classifier to predict SQLIA [71]–[74]. Another method is to extract query strings or URL from the web as a data set used in applying ML technique to predict SQLIA [75]. Alternatively, do a textbook evaluation using purpose-built data set of HTTP dataset CSIC 2010 as implemented by Nguyen et al. [76]. These procurement techniques described above results in repeating strings devoid of pattern. The drawback of these methods is the limitation of only predicting SQLIA based on the scope of the saved logged file used as a data set to train a classifier. Also, there are results of poor performance metrics of the hashed matrix vectors required to train a classifier extracted from these query string transformations that would often contain spaces and comments of SQLIA signatures which are often exploited by an intruder. Thus, when such query strings are hashed for the matrix vectors to train a classifier, a shift of spaces between strings would result in high prediction errors of a false negative.

The existing research work [72], [77]–[79] applying the AI approach were used for evaluations using ROC curve and CM, but not how you would deploy the trained model in the implementation of these proposals. We propose that string pattern exists in every input data in both legacy and new web applications and that these can be leveraged to generate as many derivations of related member strings when combined with existing known attack signatures to create a labelled pattern-driven data set fit for training a classifier to build a real-world SQLIA mitigation.

Overall, the approach presented in this thesis obtains pattern-driven data set from the pattern of expected valid data and established SQLIA signatures. The proposal applies string replication, transpositions including using a tool named Regular expression explorer (Rex) [80] to generate similar patterns of strings (related member strings) as to obtain massive learning data to train a classifier. The trained supervised learning model is then evaluated in MAML studio with a highly trained model deployed as a web service in the ongoing SQLIA mitigation as aforementioned and illustrated in Figure 1-2.

1.4 Thesis Chapters Outline

1.4.1 Overview

This thesis outline summarises the chapters starting with the introduction, followed by a background, literature review chapter that form the building blocks to the contributions detailed in Chapters 4, 5 and 6 and ends with a conclusion in Chapter 7. The thesis is laid out in seven chapters including the Chapter one which provides the thesis synopsis. The Chapter one covers the introduction, research goals, contributions and thesis outline which gives a high-level overview of each of the subsequent chapters described below:

1.4.2 Chapter 2: Background

Chapter two has the background and theories around SQLIA and the approach presented in this thesis. In this chapter, we discuss attack intent, injection mechanism; SQLIA types, web proxies [40]; FSA [56], [58], [81], and ML principles that are used throughout the thesis. The background theory on SQLIA which include attack intent; injection mechanism and SQLIA types [82] forms the fundamental principles on which SQLIA mitigation proposed by researchers over the years are based that is discussed here.

Packet interception for analyses to detect intrusion exploiting SQLIA have been explored in Ariu & Giacinto [83]. An application-level proxy applied in this thesis performs better in intercepting and decrypting obfuscated web requests than low-level network packet interception tools which suffer from message fragmentation in large volumes of gigabits per seconds of packets on the wire. The proxy interception of web requests for analysis to mitigate SQLIA has been successfully explored in previous research work by various researchers over the years [84]–[88].

SQLIA mitigation exploring FSA provides a functional approach to pattern analysis for various SQLIA signatures. Analysing web requests for patterns of SQLIA signatures by implementing RegEx can be found in research work by academics proposed over the years [5], [88]. ML techniques have been explored over the years by academics as a bio-inspired technique to mitigate SQLIA [72], [73], [89]–[92]. However, solely relying on pattern analysis with a regular expression in emerging computing with big data emanating from various web requests to analyse for SQLIA is computationally intensive. In this thesis, we apply a regular expression to generate pattern-driven data set used to train a classifier in ML techniques to mine the large web requests in SQLIA mitigation [61]–[64]. This background knowledge of SQLIA and FSA fundamentals forms the basis to mitigate SQLIA in the techniques proposed in this thesis.

1.4.3 Chapter 3: Literature Review

Chapter three examines the literature review on SQLIA to establish research up to date and identify gaps in the existing proposals by various researchers. We broadly classify these proposals over the years into three categories. Namely, SQLIV testing and detection; defensive coding in web application code sanitisation for SQLI prevention; and dynamic runtime analysis, including taint-based and approaches applying Artificial Intelligence (AI) employed in this thesis. Figure 1-3 presents the three categories of proposals over the years.

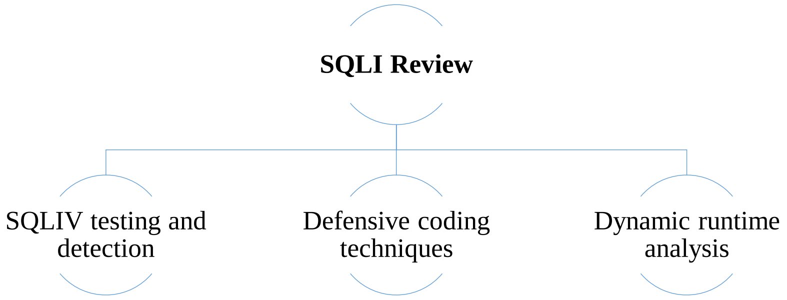


Figure 1-3 SQLI review category chart

SQLIV testing and detection approach is a vulnerabilities penetration testings that include static testing [8], [82], [94]–[100] and dynamic runtime testing [101]–[109]. Penetration testing approaches are mostly applied in SQLIV detection, and there is a requirement for source code access and with only a handful that extended the proposal to include prevention [110].

Defensive coding is a developer-centric web application code sanitisation for SQLI prevention [111]–[118]. The defensive coding is an approach usually prescribed by OWASP and needs the developer’s security awareness before web application development. Also, the functioning of this approach strongly relies on string lookup and regular expressions which are computationally intensive in a large volume of web requests to and fro cloud-hosted applications as seen recently in emerging computing.

Dynamic runtime analysis [4], [5], [124], [125], [10], [84], [119]–[123], including taint-based [7],[126], [127] and approaches applying AI involves dynamic runtime comparison of queries with static generated queries to detect SQLIA signatures. The taint-based technique involves labelling of data which is then tracked to the database entry to assert if it is from trusted or untrusted sources. In AI [71], [72], [134]–[141], [74], [100], [128]–[133] the static generated queries exist as a labelled data set which is then used to train a classifier to predict SQLIA in web requests.

In this thesis, we reviewed SQLIA mitigations proposals over the years and observed the following:

- Proposals over the years were aimed at web applications hosted within organisations' intranet or extranet.
- The proposals were string lookup that required source code access.
- The proposals would not scale in emerging computing of big data emanating from the enormous volume of distributed web requests to cloud-hosted applications.
- The non-availability of pattern-driven data set to apply AI, but with the few data set relating to SQLIA being out of date in relevance to real-world application.

On account of the literature review, we propose in this thesis a pattern-driven data set to train a supervised learning model in SQLIA mitigation to have answered the research questions.

1.4.4 Chapter 4: Numerical Encoding to Tame SQLIA

Chapter four titled Numerical Encoding to Tame SQLIA is the subject of an IEEE conference paper titled Numerical Encoding to Tame SQLIA (NETSQLIA) [62] and a European Conference on Cyber Warfare and Security Conference (ECCWS) paper titled Applied web traffic analysis for numerical encoding of SQL Injection Attack Detection and Prevention [61]. In these papers, we discussed our first approach to generate a pattern-driven data set where none exists, including using ML supervised learning algorithms to train this derived pattern-driven data set in towards SQLIA mitigation with evaluation through ROC curve and CM. We inferred in these papers experimental results of accurate prediction of True Positives (TP) and True Negatives (TN), but with an improvement in prediction errors of low False Positives (FP) and False Negatives (FN) with cross-validation of low standard deviation (variance) of accuracy demonstrating a trained model of reduced biases to unknown data.

The web application types are so diverse and dynamic to rely on out of date competition-driven data sets [65], [66] that were only good for the detecting of anomalies in the competition and project for which they were generated. These academic competitions-driven data sets do not benchmark performance metrics for an AI trained model in the real-world diverse web application types. On account of the lack of all-purpose data set of the various web applications that exist; there is a need for a paradigm shift from relying on out of date competition-driven data sets to a pattern-driven data set. The pattern-driven data set which varies in web application types is generated from the pattern that exists in the expected legitimate web request input, including illegitimate web requests containing SQL tokens and injection attack signatures to a web application which is one of the novel contributions of this thesis.

This chapter examines the viability of obtaining pattern-driven learning data to train a supervised learning model towards applying ML techniques in SQLIA detection and prevention. It discusses the following research question of: Can the patterns that exist in both expected web requests and SQL tokens including existing SQLIA signature extraction be used to create a pattern-driven data set with vectors required to train a supervised learning model and evaluated?

We derived a data set from the patterns that exist in an expected input data to web applications including SQL tokens and SQLIA signatures. These SQLIA mitigation artefact are further encoded to numerical values to obtain vectors required to train ML supervised learning models. We identified in the experiments conducted; there is a need for the proposed model implementation to be able to process actual string input as against numeric values, this sets precedence for further work in subsequent chapters. We further the numerical encoding to a feature hashing in following Chapters 5 and 6 and related conference papers. Feature hashing transform strings into a set of features represented by vector matrices [60], [142] require to train an ML model.

1.4.5 Chapter 5: Applied predictive analytics to mitigate SQLIA

Chapter five titled Applied predictive analytics to mitigate SQLIA forms the subject of an IEEE conference workshop paper publication titled Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention [63]. In this paper, we presented the hashing of features to obtain vector matrices needed to train a supervised learning model to adopt the proposal in a proof of concept implementation in an application that processes strings input on intercepted web requests to predict SQLIA.

This chapter examines the research question: if a web application type being protected has the artefact for the extraction of a pattern-driven data set? The chapter discusses employing the string patterns of R string package to derive massive learning data of expected web request input. The data set is transformed to vector matrices obtained by hashing to train a supervised learning model. The chapter demonstrates a proof of concept of a web application where a pattern-driven data set containing English dictionary words labelled as SQLIA negative and SQL tokens including SQLIA signatures labelled as SQLIA positive to train a classifier to predict SQLIA. An encounter of SQL tokens including known SQLIA signatures in an intercepted web request for analysis is predicted as SQLIA positive while dictionary words excluding SQL keywords are predicted as SQLIA negative. The trained model is evaluated using statistical measures, and a ROC curve with prediction results of high TP and TN, but low FP and FN of SQLIA with cross-validation of low variance in accuracy demonstrating a trained model of reduced biases to unknown data.

1.4.6 Chapter 6: Pattern-driven data set to mitigate SQLIA

Chapter six titled Pattern-driven data set to mitigate SQLIA is the subject of an IEEE conference paper publication titled An Applied Pattern-Driven Data set to Predictive Analytics in Mitigating SQL Injection Attack [64].

The chapter further the research question answered in Chapter 5 with a different method to generate pattern-driven data set. This chapter employs FSA and SFA principles to produce all possible related member strings to derive a pattern-driven data set from a web application type expecting dictionary word as a valid input. The data set from the web application type is used to train an ML model to demonstrate a web application type could produce a relevant artefact for SQLIA mitigation. The approach presented uses hashed pattern-driven data set from a web application type context as against relying on access to the web application’s source code or queries comparison, which has been employed in existing static and dynamic signature research approaches over the years. We obtained high performance metrics results validated through the statistical measures and the ROC curve, including cross-validation of low variance in accuracy, demonstrating the trained model of reduced biases to unknown data in a real-world web application.

1.4.7 Chapter 7: Conclusion

Chapter seven has the conclusion. Based on the experimental results of the pattern-driven data set, we inferred patterns exist in every input data in both legacies (age-old web applications) and new web applications. These patterns can be leveraged to generate as many derivations of related member strings as to obtain massive learning data to train a supervised learning model that is validated with statistical measures including ROC curve, CM and cross-validation. We conclude that pattern-driven learning data extracted from any web application using the approach presented in this thesis can be used to train a supervised learning model that is deployed as a web service in ongoing SQLIA detection and prevention.

2 Background

Figure 2-1 presents the organisation chart layout of this chapter with the sections.

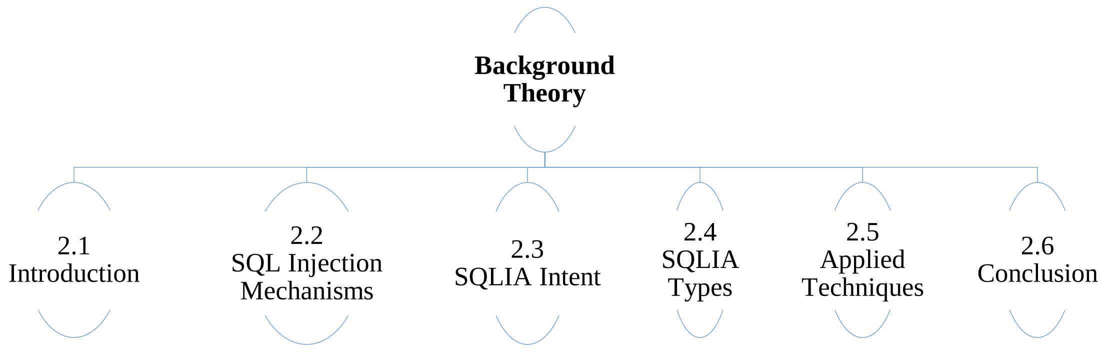


Figure 2-1 Background theory chapter organisation chart: The chapter organisation chart shows the arrangement of the chapter's sections.

2.1 Introduction

SQLIA is any manipulation of web requests from a front-end web application by an intruder with intent to compromise the confidentiality, integrity and authorisation of the back-end database. Over the years, researchers have proposed various approaches to detect and prevent SQLIA [4], [5], [14], [20], [110], [143]–[145], [6]–[12]. The approach presented in this thesis intercept web requests by proxy from any SQL injection mechanism and analysed for SQL injection intent and types in providing a solution to mitigate against SQLIA. To better address the issues of SQLIA, researchers have discussed this background theory under the following headings of SQLIA mechanism, intent and types [82]. The rest of the chapter is organised as follows as illustrated in Figure 2-1. In Section 2.2 we discuss various conduits for SQLIA under SQL injection mechanisms. In Section 2.3 we discuss the intruder intention to carry out SQLIA under SQLIA intent while in Section 2.4 we discuss various SQLIA types employed by an intruder for an effective SQLIA as to provide a broad understanding of the SQLIA in delivering a solution. Also, we discuss in Section 2.5 the fundamental of the applied techniques in mitigating SQLIA presented in this thesis.

2.2 SQL Injection Mechanisms

Figure 2-2 presents this section organisation chart with the associated subsections.

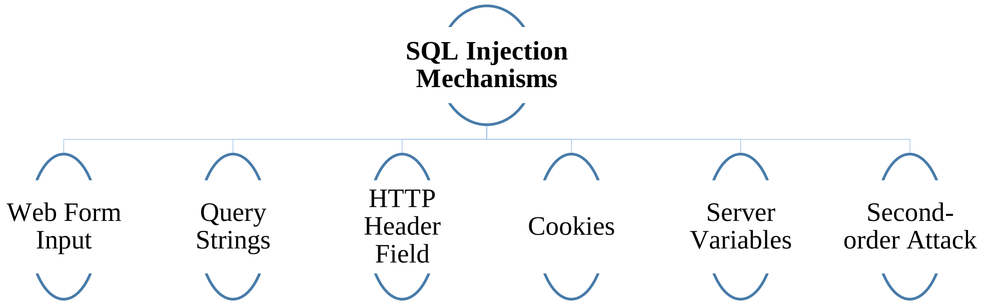
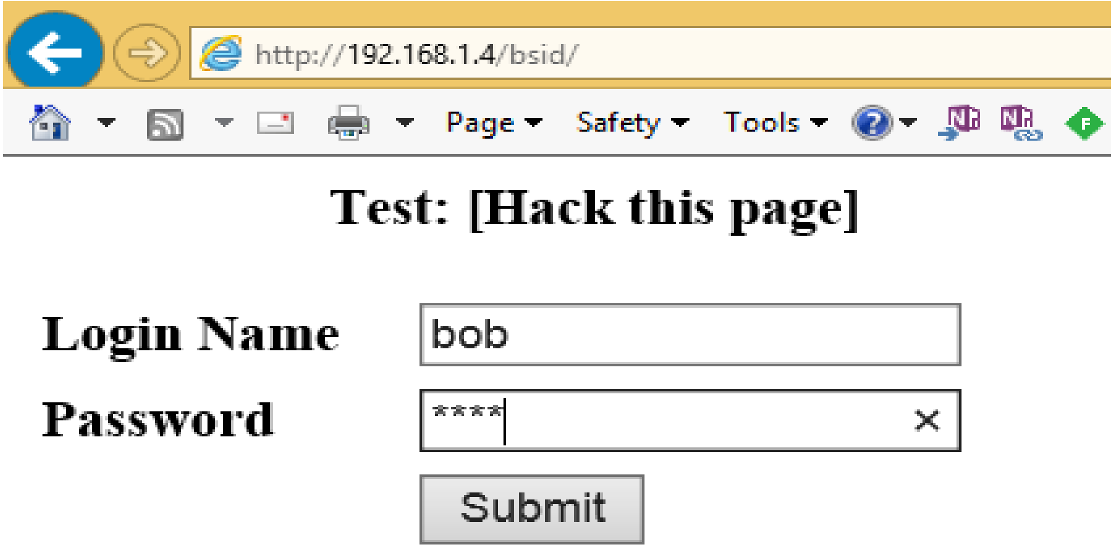


Figure 2-2 The section organisation chart: The section organisation chart with the related subsections.

The SQL injection mechanism is the access technique explored by an intruder to compromise the security of a protected database. It is the commonly used conduit to introduce SQLIA to a back-end database-driven web application.

2.2.1 SQL Injection through Web Form Input

The web browser displays web forms that provide interactive access to web applications driven by HTTP GET or POST operations [146]. The input form of a loaded web page can be one of the conduits for SQLIA. An intruder can compromise a well-intention validation web form requiring a login name and password as shown in Figure 2-3 to an injected example presented in Figure 2-4 below.



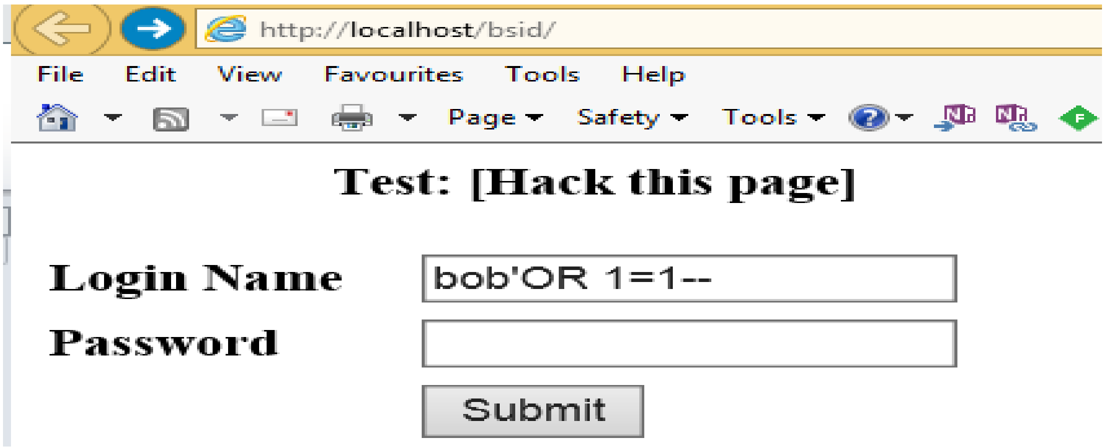
Test: [Hack this page]

Login Name bob

Password ****

Submit

Figure 2-3 Web form input: A figure of a web form illustrating input data or web request data that is exploited as one of the conduits for SQLIA.



http://localhost/bsid/

Test: [Hack this page]

Login Name: bob'OR 1=1--

Password:

Figure 2-4 An Injected web form: A figure of a web form illustrating an SQLIA with just login name receiving an input bob appended with a tautology SQLIA type signature of OR 1=1 and -- to omit the password field.

In the tautological SQLIA example presented in Figure 2-4, the requirement for the password is bypassed with the login name field receiving an input of *bob* but with the injection signature of *' OR* (SQL symbol and keyword), *1=1* (a tautology which is always true to return all records) and -- (comment to ignore everything after the login name field e.g. the password field in this case).

2.2.2 SQLIA through query strings

During HTTP POST or GET operations of regular web form interactions, the Universal Resource Locator (URL) and associated input parameters termed query string can be injected by an intruder to circumvent any validation being provided to a secure web application. A valid web request that would appear in the query string as http://bsid/bsid/Data Page.aspx?Login Name=bob &Password=@bob can be injected by an intruder by recrafting the query string to http://localhost/bsid/DataPage.aspx? LoginName='OR%201=1--&Password= thereby omitting the requirement for a valid login name and password to access the application.

2.2.3 SQLIA through HTTP header fields

Web request messages contain HTTP header, which the following header parameters of X-Forwarded-For, user-agent and referrer are susceptible to SQLIA. The HTTP header parameters are typically overlooked and as such 75 % of the web application penetration scanners could not detect HTTP header SQLIA vulnerabilities [147]. The header parameters of X-Forwarded-For, User-agent and Referer fields can be injected with strings 'OR 1=1.

2.2.4 SQLIA through cookies

Cookies have stored state information which can be exploited in SQLIA to gain unauthorised access to the back-end database. Cookies contain stored state information of the actual interaction with the web application which can then be intercepted by an intruder to circumvent validation requirements thereby having full unauthorised access to the database.

2.2.5 SQLIA through the second-order attack

The second-order is an SQLI through a web application that allows an update without a proper input validation. An intruder will cleverly craft a trojan horse into the input supplied during registration, for example, a login name of "Alice" is registered as "Alice` --``. The symbols " `--" provide a ticking time bomb to be exploited at a later day by the intruder to update any ``Alice`` login with a new password to hijack the account.

2.3 SQLIA intent

Figure 2-5 presents this section organisation chart with the associated subsections.

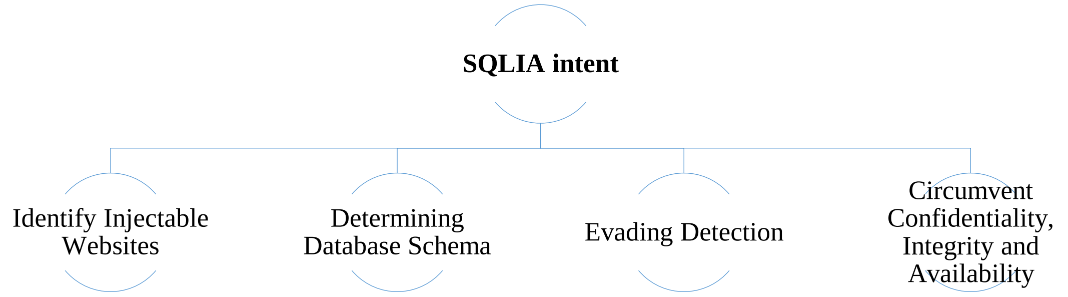

SQLIA intent
Identify Injectable Websites
Determining Database Schema
Evading Detection
Circumvent Confidentiality, Integrity and Availability

Figure 2-5 The section organisation chart: The figure shows this section organisation chart with the related subsections.

An intruder will carry out SQLIA for a purpose or an intention which is aimed to circumvent confidentiality, integrity and availability of the fidelity of a Relational Database Management System (RDMS). In this section, we discuss both the preliminary steps an intruder will employ to establish an SQLIA and the intent for such attack.

Though recent years have seen some tools being made available that automate these probing of application's web pages for vulnerabilities as discussed in 1.1 above, the manual process is still being used by a determined intruder to carry out these probes in advance of SQLIA.

2.3.1 Identifying injectable websites

A web form page can be tested for SQLIA vulnerabilities by looking at the telltale signs of parameter passing indicated by a question mark (?), e.g. http://bsid/Data Page.aspx? Login Name=bob in the query string [148]. Alternatively, in a scenario where a web form exists without the parameter passing in the query string, e.g. http://bsid/DataPage.aspx, a resulting error by a single quote (‘) input on the web form can expose vulnerabilities to SQLIA. Further to this manual process, automated vulnerability scanners referred to in 1.1 above can also be used.

2.3.2 Determining database schema

The schema holds the metadata of a database which contains detailed information about the names of tables, columns, functions and data types which are invaluable information to an intruder to launch SQLIA. A simple single quote (‘) input to a web form can spill out this sensitive information as an error. A more sophisticated approach can be employed by using penetration testing SQLIV scanners like Burb Suite [15], SQLMap [16] and BSQL Hacker [17].

2.3.3 Evading detection

An intruder would want to remain undetected so as not to set off an alarm in an audit log or any protection mechanisms to the web application as to be able to carry out its planned attack from start to completion. Apart from an intruder achieving a successful attack by employing tools that stealthily evade detection, there are also severe legal consequences in various jurisdictions that result from the hacker being caught. For example, the perpetrator of hacking can be dealt with under the Computer Misuse Act (CMA) 1990 in the United Kingdom [149].

2.3.4 Circumvent confidentiality, integrity and availability

Preserving the principle of information security of a protected asset is to maintain the attributes of Confidentiality, Integrity and Availability (CIA). SQLIA intent is to circumvent any of these attributes in the following ways: bypassing confidentiality by unauthorised access to confidential or encrypted data stored in the database; compromising database integrity by intercepting and modification of protected data; and carrying out malicious operations to drop tables and database resulting in non-availability of the database. These operations are performed by any individual or combination of the following methods: determined drawn-out manual processes (which include executing remote commands) and automated SQLIA exploit tools.

2.4 SQLIA types

Figure 2-6 presents this section organisation chart with the associated subsections.

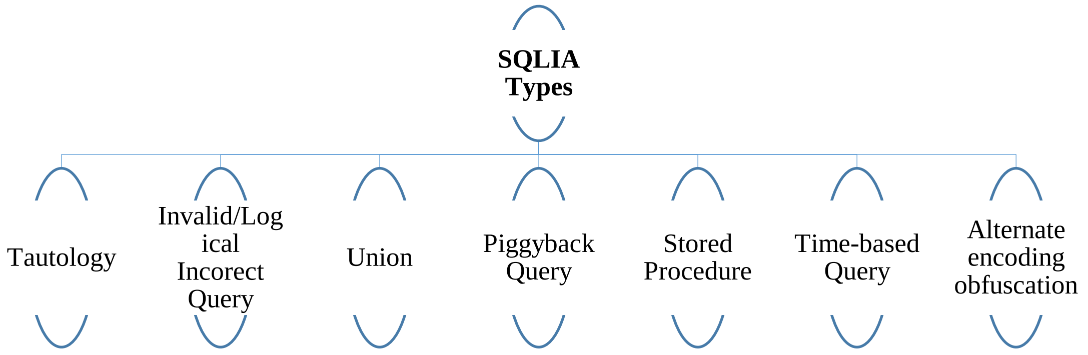


Figure 2-6 The section organisation chart: The figure shows this section organisation chart with the related subsections.

SQLIA types are the techniques employed by an intruder to carry out SQLIA. The intruder would typically apply various SQLIA types in combination to open a vulnerable web application to be more susceptible as to ease the intended SQLIA from a start to a successful finish.

Throughout this section, simple query examples (not exhaustive) are presented in Table 2-1 below to illustrate the different SQLIA types known signatures that are labelled SQLIA positive in the encoding of features into vectors. The section below provides a high-level overview of the SQLIA types which form the basis for the feature extraction of SQLIA payload that is encoded into numerical data (vectors). The layout of the tables below has SQLIA type (attack type), web request query strings, parsed Transact-Structured Query Language (TSQL), attack intent, and SQLIA signatures pattern.

2.4.1 Tautology

Tautology based SQLIA type is carried out by injecting vulnerable sites with query strings that are altered in transit to retrieve data from the database. The classic example is by assigning modified strings to the WHERE clause which the query is incorrectly terminated with a single quote (‘) or with a tautological statement like OR 1=1 as shown in Table 2-1. The outcome is always parsed by the SQL parser to be true, meaning that the security validation of login credentials will be bypassed to retrieve all the records from the affected back-end database table.

Table 2-1 SQLIA types and attack signatures: A table of SQLIA types illustrating attack signatures.

SQL Types	Query string	Parsed TSQL	Attack Intent	Pattern
Tautology	http://localhost/bsid/DataPage.aspx?LoginName=bob'OR%201=1--&Password=	SELECT loginName, password FROM tblUser WHERE loginName= 'bob' OR 1=1--	Injectable hotspots, circumventing authentication, pilfering data	`--,1=1, ‘a’=’a’ etc.

Logical incorrect queries	http://localhost/bsid/DataPage.aspx?LoginName=';IF((SELECT%20user)%20=%20'sa'%20OR%20(SELECT%20user)%20=%20'dbo')%20SELECT%201%20ELSE%20SELECT%201/0;--2%80%99&Password=	SELECT loginName, password FROM tblUser WHERE loginName=’; IF ((SELECT user) = 'sa' OR (SELECT user) = 'dbo') SELECT 1 ELSE SELECT 1/0; --	Detecting vulnerabilities, pilfering data, and database fingerprinting	SQL Keywords e.g.
				SELECT
				IF, ELSE
				sa, dbo,1/0

Union queries	http://localhost/bsid/DataPage.aspx?LoginName=&Password=UNION%20SELECT%20CreditNo,%20CustAddress%20from%20tblCreditinfo	SELECT loginName, password FROM tblUser WHERE loginName=' ' UNION SELECT CreditNo, CustAddress from tblCreditinfo	Bypassing authentication and pilfering data.	‘ UNION SELECT
Piggyback queries	http://localhost/bsid/DataPage.aspx?LoginName=;%20SELECT%20*%20from%20tblCreditInfo%20--	SELECT loginName, password FROM tblUser WHERE LoginName="; SELECT * from CreditCardInfo --	Extracting data beyond the scope of the web application and executing commands	‘ ;
Store procedure queries	http//bsid/bsid/login.aspx?LoginName=’ ’;exec xp_cmdshell 'attrib "c:\test\spuriousfile.vbs" +r'	SELECT * FROM tblUser where loginname ='‘; exec xp_cmdshell 'attrib "c:\test\spuriousfile.vbs " +r'	Privilege escalation, remote command, performing denial of service	;,Exec, xp_cmdshell Attrib c:\test\spuriousfile.vbs " +r
Time-based queries	http://localhost/bsid/DataPage.aspx?LoginName=';%20waitfor%20delay%20'00:00:10'--&Password=	SELECT loginName, password FROM tblUser WHERE loginName=' '; waitfor delay '00:00:10'--	Probing of injectable hotspots and database schema	‘ waitfor delay '00:00:10'--
Alternate encoding obfuscation	http%3A%2F%2Flocalhost%2Fbsid%2FDataPage.aspx%3F%0ALoginName%3Dbob%27OR%25201%3D1--26Password%3D%0A	SELECT loginName, password FROM tblUser WHERE loginName=' '; waitfor delay '00:00:10'--	Obfuscating data to evade detection	% Hex values & Numeric values --

2.4.2 Invalid/logical incorrect query

The incorrect query is often used to probe the vulnerabilities of the target before the actual SQLIA. The information gained during this initial phase is used by the intruder to determine what form of further attack to carry out on the target web application. The example query string in Table 2-1 will return a divide by zero error if the login account is not sa or using dbo schema.

2.4.3 Union

This attack exploits the union command ability to query multiple database tables to retrieve confidential data far beyond the tables used in the web application as shown in Table 2-1. There are intermediate steps of using ORDER BY to get column names [150].

2.4.4 Piggyback queries

The intruder exploits the semicolon (;) which is the end of an expression of a valid SQL query. In this example, the intruder appends a semicolon to a valid SQL statement to carry out SQLIA. An intruder would typically combine other SQLIA type [150] to obtain database table names. In Table 2-1, the SQL query is piggyback to run SELECT * from tblCreditinfo to gain unauthorised credit card information.

2.4.5 Stored procedure

The intruder elicits database information by exploiting xp_cmdshell if enabled, to trigger a trojan horse file for a malicious attack at a later day. Also, stored procedures are vulnerable to privilege escalation, buffer overflows and even manipulated to gain elevated permission access to perform operating system-wide operations [82]. In the example query presented in Table 2-1, an attacker runs xp-cmdshell against spurious text files loaded in the target to circumvent the security of the database.

2.4.6 Time-based query

The time-based attack is a time-delayed type of SQLIA where an intruder probes a site to elicit a response after a period of time. The intruder starts off by issuing a SQL query and waits for results. The query shown in Table 2-1 will display a response after ten seconds, which provides an intruder with the information required to further additional attacks.

2.4.7 Alternate encoding obfuscation

Alternate encoding obfuscated exploits a combination of escape and Unicode characters which the computer systems interpret as valid characters without distinction from deliberate obfuscation. An intruder can circumvent solutions being provided in pattern matching of SQLIA as it becomes unreadable as shown in Table 2-1.

Not limited to the above injection mechanisms and SQLIA types which are the subject of the pattern-driven data set labelling of SQLIA positive in this thesis; there are other forms of pattern evading techniques like the use of comments, whitespace, character casing and encryption that are exploited by an intruder in SQLIA.

2.5 Principles of Applied Techniques

Figure 2-7 presents this section organisation chart with the associated subsections.

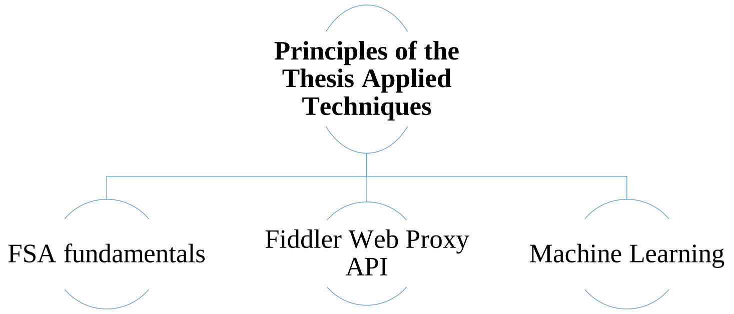


Figure 2-7 The section organisation chart: The figure presents this section organisation chart with the associated subsections.

We propose in this thesis the interception of web requests at the proxy API to predict SQLIA. The scheme presented here demonstrates deriving data set using the FSA technique to create pattern-driven learning data needed to train a supervised learning model. A Web Proxy API is required to intercept web requests to application endpoints while the MAML platform provides a cloud-hosted environment to build and deploy an ML model through a web service.

2.5.1 FSA fundamentals

FSA is an abstract state transition machine in response to inputs as to change from one state to the other [56], [58], [151]. In the scheme presented in this thesis, we explore the FSA state transition walk for the derivations of related member strings to obtain massive volumes of learning data. ML techniques require a large data set that is meaningful to train a classifier without which a trained model becomes worthless [70]. The approach explores traditional Nondeterministic Finite Automata (NFA) implemented in .NET RegEx to define patterns and constraints that exist in the expected input, including known SQLIA signatures and SQL tokens to be predicted at the SQLIA hotspot.

To apply our approach in generating learning data; there is a need to produce patterns of expected input data. Figure 2-8 below illustrates the FSA states transition walks that forms the fundamental building blocks of our learning data extraction technique from the patterns that exist in the expected input data including SQLIA signature and SQL tokens.

In Figure 2-8, let the transitions alphabets which are the substrate for member strings generation to be represented by a set of {a, b}. The states transition walks can be condensed into a regular expression of *a(ab)*|bb(ab)** which can accept these possible inputs sets of member strings *{a,aab….b$_n$, bb,bba,bbab…b$_n$}*.

The state walks from the initial state to the accepting state via transition states shown in Figure 2-8 can be interpreted as follows:

- Between initial state (1) and accepting state (2), we have {a}.
- Between initial state (1) and state (3) via accepting state (2), we have {aa}.
- Between initial state (1) and state (3) via accepting state (2) and back to accepting state (2), we have {aab}.
- Between initial state (1) and state (3), we have {b}.
- Between initial state (1) and accepting state (2) via state (3), we have {bb}.
- Between initial state (1) and accepting state (2) via state (3), and back again to state (3), we have {bba}.

The states walk can go on with {bbab}, {bbabb…..bn} and so on [152]. This forms a classical NFA transition states with nondeterministic states walk.

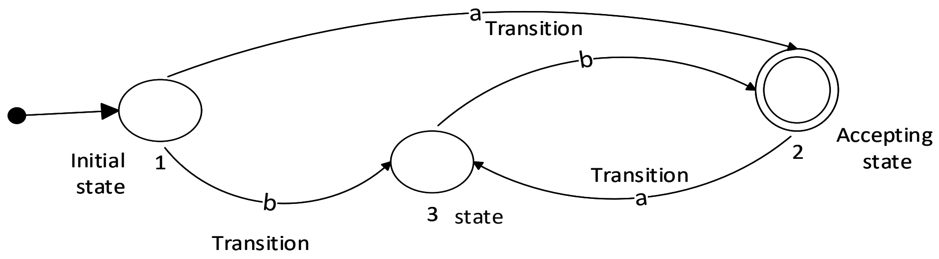


Figure 2-8 FSA illustrating transition state walks

These above manual collations of transition states walk to generate all possible member strings' regular expression is automated by employing the research work by Veanes et al. [153], [154] on Symbolic Finite Automaton (SFA). The SFA technique is used to generate the member strings computationally from the tool they developed named Regular expression explorer (Rex) [80]. We use Rex to generate massive volumes of learning data.

2.5.2 Fiddler Web Proxy Application Programming Interface (API)

A Fiddler proxy provides the functionality to intercept HTTP/ HTTPS web request for debugging, session manipulation, security testing and extending the API through .NET language [40], [155]. We intercept the web requests and predict SQLIA at the hotspot (predicate and expression of the SQL query structure). We extract the query string or the supplied input in a custom .NET application that implements a Fiddler proxy [40] API calling a web service of a trained supervised learning model to predict SQLIA.

A web proxy is the most suitable to intercept web requests including those originating from any injection mechanisms to SDN cloud applications' end-points. An application-level proxy performs better in intercepting and decrypting obfuscated web requests rather than low-level network packet interception tools which suffer from message fragmentation in large volumes of gigabits per seconds of packets in the wire. Also, the injection mechanisms to a vulnerable application can originate from web page forms, second-order injection, exploiting web-enabled server variables, query strings, and through cookies which proxy can intercept.

2.5.3 Machine Learning (Thesis perspective)

In the applied research presented here aimed at towards providing SQLIA mitigation for web application in emerging computing of big data, we identify the upward trend in web requests traffic to cloud SDN because of web application increasingly hosted in the internet cloud. We recognised the universal solution to mine big data which ML provides, and the issue of historical learning data relevant to a real-world web application type in applying ML. We then ask a research question of can patterns in both web application type valid input and SQL tokens including existing SQLIA signature extraction be used to create a large volume of learning data with vectors required to train a supervised learning model to the maximum? This research goal and related questions are answered in Chapters 4, 5 and 6 in employing the ML approach to SQLIA mitigation.

2.5.3.1 ML fundamentals

ML is a technique of data science that assist computers in learning from historical data to predict future trends and outcomes or events. Thus, in applying ML to SQLIA mitigation, there is a need for relevant pattern-driven data set with enough artefact to predict SQLIA which is lacking in the existing proposals as the data set procurement was query extraction that is geared towards query comparison at runtime.

2.5.3.2 Traditional programming vs Machine learning

In traditional programming, it is driven on the paradigm of computational program requirement to process data for an output such as traditional string lookup or query matching approach in existing proposals [4], [5], [96]–[98], [101]–[107], [7], [111], [114], [116]–[123], [8], [124], [126], [127], [10], [82], [84], [94], [95]. Traditional programming lack cognitive automation (applying artificial intelligence to intensive information processing) as any reasoning or operation to be carried out on data need to be programmatically coded by the developer as needed, so the output cannot evolve on its own or from experience.

Conversely, in ML the historical learning data are used to predict an output which then provides cognitive automation in a computational program that continually innovates and improves with experience such as the research goal of this thesis of using a pattern-driven data set to predict SQLIA in the big data context. Figure 2-9 is a figure illustrating the distinction between traditional programming and machine learning [156]. In a terse statement, ML is a process of transforming historical data into a predictive analytics software. Predictive analytics is a process of employing mathematical formulas termed algorithms or classifiers to analyse current data to identify patterns or trends to predict future events.

The ML learning approach is more suited for a computational task that mimics the human learning curve to carry out complex tasks with speed effectively. Employing ML technique offers cognitive automation to the research goals of this thesis as it provides a method to train a classification algorithm to the maximum with the pattern-driven data set to predict SQLIA in the context of emerging computing effectively. This cognitive automation requirement cannot be said of a traditional SQLIA mitigation lookup approach which a programming construct only produce an output or detect an SQLIA signature type which does not account for associated polymorphic signatures derivation by an intruder to bypass protection, e.g. SQLIA type of tautological attack signature of 1=1 can also be written as 1 > 1, ‘a’=’a’, etc. [157].

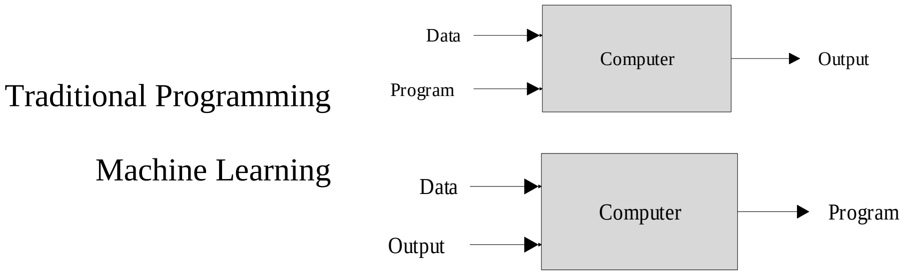


Figure 2-9 Traditional programming vs Machine learning: A figure illustrating the distinction between traditional programming and machine learning.

2.5.3.3 Data set

Availability of sizeable historical learning data or data set relevant to real-world application type has advanced many fields to apply ML to mine these data set with examples in better speech recognition, image recognition, meteorological weather forecasting, and breast cancer diagnosis as a few examples. The University of California, Irvine (UCI) ML repository currently maintains about 438 data sets [3], [158] that are widely used by the ML community. The UCI available data sets are continually increasing as more data sets are added. The procurement of such historical learning data set vectors is straightforward in the natural sciences where observations and measurements over a period are recorded to produce a large volume of data set with feature vectors available to predict future events.

Unfortunately, the issues of ageing data set and lack of standardised ones are known issues in Intrusion Detection System (IDS) research [1], [2], [67], [159], [160] with the antiquated KDD Cup 1999 [161]. Also, the same plight is faced in SQLIA detection and prevention research when applying ML which requires data set input of vectors, but with few old data sets which were provisioned for a challenge competition or a purpose-built as in ECML/PKDD 2007 [65] and HTTP CSIC 2010 [66]. These obsolete data sets had served the purposes in the respective challenge competitions and as such not a test case for validation benchmark or training a classifier for real-world application. Also, it must be pointed out; these data sets do not hold the same weight as the natural science data sets made available by the UCI that are based on actual recordings of measurements and observations over a period. The UCI data sets have features that are relevant in solving real-world classification tasks in better speech recognition, image recognition, meteorological weather forecasting, and breast cancer diagnosis as a few examples.

The obsolete nature of the few existing data sets [65], [66] that relate to the SQLI domain of study can be seen as none of the data sets to have been used in existing work applying ML [71], [72], [134]–[141], [74], [100], [128]–[133]. These existing works employing ML techniques resort to generating their test case data sets. These data sets revolve around URL query string extraction by web crawling or a log file of activities of automated SQL server database interaction which is one of the reasons in most existing work to have gravitated towards query comparison or strings matching of some sort. Also, in most of these data sets that include few competitions-driven data set available in the public domain [65], [66], and privately generated data set for a test case in existing work (not available in the public domain) quickly becomes obsolete. These existing data set short lifespan is primarily due to web application types are so diverse to have a universal generic data set for all ML-based SQLIA mitigation as such cannot serve as an empirical benchmark for the validation of the classification algorithms used.

Thus, that begs the question if the pattern that exists in web requests that are deemed valid, and historical studies of different SQLIA types including SQL tokens can provide artefact to be used to generate a data set to train an ML algorithm to predict SQLIA. We presented in this thesis that based on the web application type; a pattern-driven data set relevant for that web application type to be protected can be inferred. The ML platforms have validation and cross-validation to benchmark results as to ensure only classification algorithms trained to the maximum are deployed as a web service in an ongoing SQLIA mitigation. The pattern-driven data set is a paradigm shift from existing work over the years. The existing test case data set procurements foster on query matching, and comparison of some sort in both traditional signatures lookup and existing AI approaches; this excludes defensive coding which is input validation and sanitisation.

A pattern-driven data set has a set of subsequence and substructure of items that frequently occur together [162]. The pattern-driven data set proposal in this thesis is a data engineered or text pre-processed learning data for ML which does not have a parallel with web crawling for URL query strings or static log files of queries as seen in the procurement of existing test case data sets. The pattern-driven data set paradigm empowers the web developer or system expert to derive a data set based on the web application type they aimed to protect. Below is a list of some of the benefits of towards a pattern-driven data set over the existing data set that is not patterned-driven.

- It is a pattern engineered data set devoid of repeating strings or queries.
- A pattern-driven data set is applied as it has relevance to the classification problem.
- It is derived in the scope of the web application type that the ML-based SQLIA mitigation is being provided.
- It can be scaled to any magnitude as to have a large data set to train a classifier to the maximum as to effectively predict an outcome.

- A pattern-driven data set empowers the web developer or system expert to derive a data set based on the web application type.
- Most importantly, it does not age fast as it is tailored to a web application type.

Conversely, non-pattern driven data set as seen in some existing work and the few competitions-driven data set are logs of repeating strings which gravitate the proposals to query comparison of some sort in SQLIA mitigation employing Natural Language Processing (NLP) of using n-grams (processing of the full length of strings or n > 1) [72]. A log of repeating strings of literally the same strings lacks the intrinsic and essential properties for uncovering patterns in massive data set as in emerging computing big data.

We presented techniques in this thesis proposal and related published papers [61]–[64] to derive a pattern-driven data set on the fly in a real-world application with empirical evaluation and cross-validation in towards breaking the cycle of reliance on a static data set. Any static data set will be out of date and irrelevant in no time, but a technique to create one based on application type will empower system expert to derive a relevant data set of intended application to be protected.

2.5.3.4 Learning algorithms

Supervised learning or classification algorithms use crafted data set by a human with labelled class in the learning data on what to predict while unsupervised learning models are not labelled instead employ clustering of data using distance and weight functions to group related data [163], [164]. Semi-supervised learning uses a labelled data to identify specific groups which further use unlabelled data to discover new groups.

In applying ML to SQLIA mitigation, we have embarked on procuring a human-mediated known pattern-driven data set that has a known labelled output to predict an outcome of SQLIA. This thesis proposal has the trappings of a classic case for a supervised learning predictive analytics from a known input data with known outcomes [156].

Figure 2-10 presents an illustration of the supervised learning logic from human-crafted known input data combined with known outcomes to produce a new predictive model program (supervised learning). Because of the pattern-driven data set approach to an ML classification and regression algorithms, we deemed supervised learning as the functional approach to address the thesis proposal of applied pattern-driven data set for predictive analytics in SQLIA mitigation. Employing the alternatives of semi-supervised classification algorithm and unsupervised clustering algorithm will be the subject of future work in fully automated pattern-driven data set procurement.

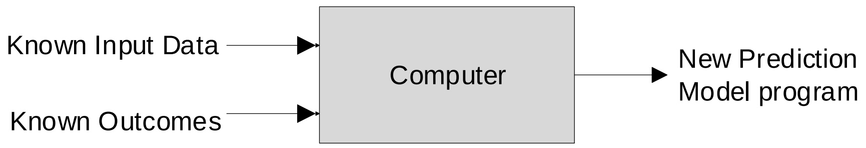


Figure 2-10 Supervised learning logic: A figure illustrating the supervised learning logic from human crafted known input data combined with known outcomes.

2.5.3.5 MAML Platform

The MAML studio or platform provides an out-of-the-box platform for building and deploying a prediction model as web services. It provides a cloud-hosted all-in-one platform to build and deploy a big data analytics solution, supervised, semi-supervised and unsupervised learning models [42].

The MAML studio provides versatile, extensible and ubiquitous cloud-based AI platform. Predictive analytics employ mathematical formulas termed algorithms or classifiers to analyse current data to identify patterns or trends to predict future events. Below lists the justification for using MAML studio in implementing the proposal presented in this research are as follows:

- MAML offers a wide range of well-known algorithms in ML and neural networks that are easily extendable within the MAML studio.
- In bringing this applied research of mitigating SQLIA with AI approach to the mainstream business application, MAML studio provides in addition to the code API of R and Python languages an easy drag and drop components. The drag and drop components have configurable properties to build, test, fine-tune, validate and generate advanced analytics to the pattern-driven data set presented in this thesis.
- The overall research goal is to examine a SQLIA problem in the context of the lack of a pattern-driven data set and provide mitigation that applies AI using this research generated pattern-driven data set. MAML studio met the research goal in

its functionality to provide all in one spot platform for API to extract data set, train a model, validate, cross-validate and deploy the trained model as a web service to be consumed by any application.

- The issue that stands out in some existing approaches on SQLIA mitigation [74], [125], [129], [133], [193] is being computational expensive. The availability of MAML cloud-based software and platform as services resolves the issue by harnessing these large-scale supercomputers employing AI techniques to deliver ready to use solutions that would have been limited relying on a single or few servers in the intranet or extranet.

2.6 Conclusion

We examined in this chapter SQLIA mechanism, intent and types to better address SQLIA. We also reviewed the fundamentals of the tools applied in this research to mitigate SQLIA that are discussed in the subsequent chapters. SQLIA research over the years [6], [8], [167]–[171], [10], [11], [20], [92], [100], [111], [165], [166] have reviewed the SQLIA mechanism, intent and types that are discussed above to have proposed various on-premise mitigations to SQLIA. These various proposals were functional in web servers hosted within the organisation’s intranet and extranet but now restricted with emerging computing.

The emerging computing of IoT and cloud-hosted services have increased the volume of web requests that need analysing for SQLI. These are web requests to back-end database emanating from diverse client web applications and emerging computing smart devices. The enormity of data requires a technique more scalable than string lookup which ML provides that is proposed in this thesis.

3 Literature Review

Figure 3-1 presents the organisation chart layout of this chapter with the sections.

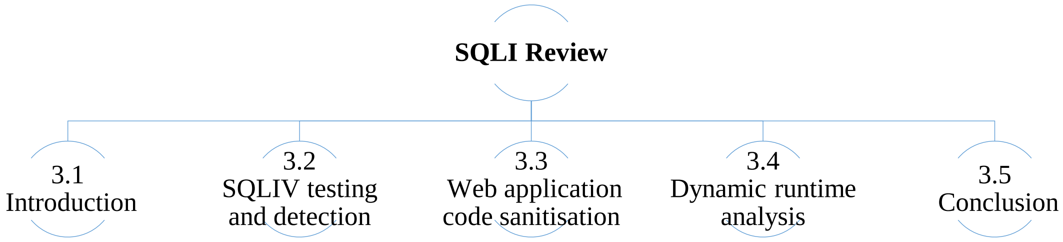

SQLI Review
3.1 Introduction
3.2 SQLIV testing and detection
3.3 Web application code sanitisation
3.4 Dynamic runtime analysis
3.5 Conclusion

Figure 3-1 The chapter organisation chart: The chapter organisation chart presents the layout of the chapter's sections.

3.1 Introduction

This chapter reviews the existing research literature on SQLI detection and prevention techniques that enhance the security of web applications. The research area of SQLI detection and prevention has seen diverse methodologies proposed over the years by various researchers and this chapter broadly classify these approaches into three sections. The rest of this chapter is organised as follows. Section 3.2 discusses SQLIV testing and detection. 3.3 discusses defensive coding in web application code sanitisation for SQLI prevention. 3.4 discusses dynamic runtime analysis, including taint-based and approaches applying AI (a similar approach to that implemented in this thesis) in the detection and prevention of SQLIA. The chapter ends with a conclusion in section 3.5. Each section is followed by a discussion on existing literature review to establish the gap in the context of emerging computing, and the contribution this thesis makes to fill in the gap.

3.2 SQLIV testing and detection

Figure 3-2 presents this section organisation chart with the associated subsections.

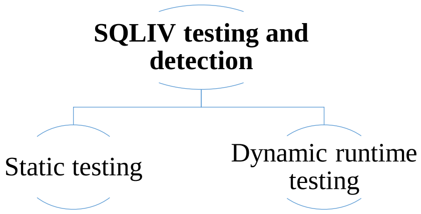


Figure 3-2 The section organisation chart: The figure shows this section organisation chart with the related subsections.

Researchers have applied penetration testing techniques to SQLI detection. Penetration testing techniques are aimed to detect application vulnerabilities during investigative auditing of web application and thus are not a traditional SQLIA prevention approach.

3.2.1 Static testing

Static code analysis applies penetration testing to examine the internal structure of the code used in web applications to detect errors and correctness at compile time. Static analysis or white-box testing is applied in detecting SQLIV to check the static syntactic structure of the code as to detect SQLIV during compile time. The approach is mostly developer-centric to create a secure web application during the Software Development Life Cycle (SDLC) and source code auditing for vulnerabilities.

Though static penetration testing approaches looked plausible in the past to detect vulnerabilities in web applications; its drawback is its reliance on the source code for SQLIV detection. The emergence of cloud computing has changed the accessibility of application source code for SQLIV detection. Also, the application domain boundary has shifted from the organisation's servers being hosted within their intranet to external cloud hosting on the internet. Hosting services, applications and infrastructure on the web such as the cloud have restricted access to source code due to the vulnerability risk of unrestricted source code.

Gould et al. [94], [95] proposed a static analysis tool named JDBC Checker to check the correctness of dynamically generated query strings. The approach explored a finite state automaton to flag the presence or absence of errors in the query strings. The JDBC checker is a Java code-centric tool aimed at only detecting SQLIV as against prevention [82].

Wassermann and Su [96] extended static code analysis to detect tautology SQLIA type by employing regular expressions for automatic dynamic code generation as to detect only tautological SQLIA type. Although effective, Wassermann and Su's approach did not go far enough in detecting other SQLIA types as a tautology is only one form of attack technique, as discussed in 2.4 above.

Fu et al. [97] designed a static analysis tool named SAFELI for detecting SQLIV at compile time by inspecting the bytecode (intermediate language) of a web application using symbolic execution to detect SQLIV at the injection hotspot. Web applications, in recent years, have become restrictive regarding access to source code which the SAFELI tool relies on to function; this hinders its ability to retrofit to vulnerable web applications.

Al-Khashab et al. [98] explore a static approach in substituting SQLI hotspots with an optimised query to detect SQLIA. In doing so, they hope that an intruder would not be able to deduce the optimised version of the SQL query in the parse tree structure to circumvent the protection provided. Unfortunately, the approach is labour intensive for developers requiring that they unwind code and start generating the optimised versions and replacing the SQLI hotspots with these optimised query versions. Also, the outcome of the optimised query does not solely rely on the query processor engine but is also dependent on other hardware resource factors such as the Central Processing Unit (CPU) on the server that is not considered in their approach.

Shar and Tan [99] build a PhpMiner II static analysis tool to predict web application vulnerabilities from static attributes during input validation and sanitisation. The tool provides an inexpensive detector of vulnerabilities, but suffers from relying on source code which is not always available for cloud-hosted legacy web applications.

Medeiros et al. [100] implemented a static analysis tool to search for vulnerabilities with further use of Machine Learning (ML) classifier for detection and classification of vulnerabilities in a web application. Their approach is lacking in real-life web application testing as the implementation relies on synthetic data which does not evolve with new signatures.

3.2.2 Dynamic runtime testing

Dynamic runtime testing is a penetration testing technique used to detect error and correctness at runtime without peering through the code, contrary to the white-box testing, which statically peered through the internal workings of the code to detect vulnerabilities. Black-box testing is applied in the SQLIV detection at runtime by dynamic analysis of a simulated mimicry of an actual web application interaction from the user to detect vulnerabilities. These approaches [101],[102] employ an automated unit test and web crawling tools that are fed with attack features at runtime to simulate or mimic an actual injection of input from an intruder to detect SQLIV.

Huang et al. [101] in their black-box testing approach developed a tool named Web Application Vulnerability and Error Scanner (WAVES) which applies web crawling to identify a data entry point (web form), and then uses an SQLI log to detect SQLIV. The method lacks completeness as it only detects cross-scripting and SQLIV in a web application but does not prevent them. Also, the approach will struggle to detect SQLIV in real-time big data scenarios due to their complex validation procedure to establish SQLIV.

Shin et al. [103] developed an SQLIV detection tool named SQLUnitGen that combines static analysis of the source code and runtime automated testing for vulnerability detection. The functioning of the SQLUnitGen consists of three processes. The first process employed symbolic execution techniques that include compiling a program to intermediate language code (bytecode) for an evaluation to recognise input that reaches the SQL query statement hotspot. The second process follows these input transformations with actual attack features. Finally, the third process compiles the program to finish as to generate a test result in a summary graph showing either a test case input success (no SQLIV) or failure (presence of SQLIV).

Approaches employing symbolic execution suffer from computational complexities to scale in large programs due to multiple loop iterations used in the evaluations as to increase multiple possible paths in the program exponentially [104]. Also, the symbolic evaluation dependency on loop iterations in evaluations is a limiting factor in emerging computing such as big data mining.

Lin et al. [105] employ a black-box testing and validation function with a secure gateway to SQLIV, but its drawback lay in need to tweak the approach for the detection of new vulnerabilities constantly.

Shahriar et al. [106] developed a tool called Mutation-based SQL Injection Vulnerability Checking (MUSIC) which is a mutation-based SQLIV testing tool. The approach employs a mutation algorithm to create nine mutation operators that introduce SQLIV into an application’s source code to generate mutants. These mutants can only be killed off by a test data containing SQLIV. The tool employs this successful elimination of mutants to establish the presence of SQLIV while the survival of mutants indicates the absence of SQLIV. Although successful, the approach is only effective against a simple WHERE clause condition in a SQL query and does not address a store-procedures SQLIA type.

Ciampa et al. [107] developed a penetration testing tool named VIp3R (“viper”) which is based on pattern matching error messages and output results produced by the application to infer SQLIV. The notion of detecting SQLIV with error messages is flawed as a successful intent to pilfer data can be carried out without SQL Server emitting any error messages as not all SQLIA techniques would return errors, e.g. the Blind SQLIA type does not include output error messages.

Appelt et al. [108] proposed a penetration testing approach for detecting SQLIV that employs mutation operators of SQLI patterns to manipulate legitimate input to trigger the detection of vulnerabilities. In further work, Appelt [110] extended the black-box testing approach using ML to train the Web Application Firewall (WAF) to detect vulnerabilities continuously. Appelt's penetration testing approach aimed to generate diverse and mutated attack string patterns for WAF to identify and prevent the likes of SQLIA. Appelt et al. approach look plausible, but an intruder is always one step ahead of human-automation to circumvent any mitigation against SQLIV using predefined synthetic mutated features, and their technique does not adapt to future novel attack.

Ceccato et al. [109] developed a tool named SOFIA for Oracle database web-driven applications. Although this is a useful black-box vulnerability testing tool, it does not offer a prevention method.

Liu et al. [102] developed a prototype tool named SQLEXP based on a black-box penetration testing approach that explores a generic static SQL penetration testing. This tool is used to generate a matrix that is assigned a test case pattern and further compared to the dynamic matrix generated at runtime to detect SQLIV. As with other penetration tests, the tool offers application vulnerability testing but does not prevent SQLIA.

3.2.3 Discussion

White-box and black-box penetration testing approaches are mostly applied in SQLIV detection with only a handful that extends the approach to include prevention [110]. Approaches that rely on both static and dynamic testing to detect SQLIV at runtime require full access to source code for detection. However, as increasing numbers of applications and services are being hosted in the cloud, access to source code has become increasingly restricted. Most modern web applications are loosely coupled (tiered application API for reusability) as against on-premise, where the domain application boundary is demarcated between the intranet and the internet. Existing approaches to SQLIV detection were based mainly on an old paradigm of the on-premise database that needed to be protected from SQLIV on the web application server within the intranet receiving web requests from the internet. This old paradigm of the on-premise hosted web application is not the case with cloud-hosted applications.

In emerging computing, there is a need for a shift in paradigm as the application domain boundary now includes the internet cloud services which require the use of the proxy API to intercept web requests at the application endpoints to detect SQLIV as to prevent SQLIA. Also, the astronomical growth of data requires machine learning to mine big data for SQLIV as against traditional data string lookup employed in most signature string searches [172]. The section below discusses the defensive coding approach to code sanitisation during SDLC.

3.3 Web application code sanitisation for SQLIV prevention

Figure 3-3 presents this section organisation chart with the associated subsections.

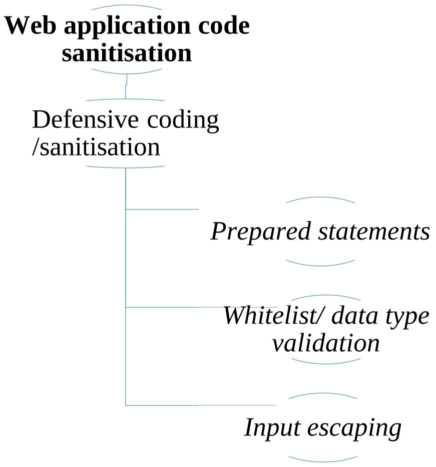

Web application code sanitisation
Defensive coding /sanitisation
Prepared statements
Whitelist/ data type validation
Input escaping

Figure 3-3 The section organisation chart: The figure shows this section organisation chart with the related subsections.

Defensive coding is an approach aimed at the developers' awareness of security vulnerabilities at design time of web applications to address SQLIV. It is an approach that has handed down guidelines by OWASP for securing web applications. Defensive coding involves prepared statement and validation of the source code for escape characters. The developers need to be trained to identify the SQLIV during SDLC as to integrate the security principles into web application development.

Defensive coding is functional during the development of new web applications, including testing and debugging. It is a manual approach that is inherent in mainstream web development programming languages like PHP, JavaScript and ASP.NET which is seen as one of the plausible defences against SQLIV. However, relying on the developer for manual defensive coding is a daunting and labour-intensive process to maintain. Also, not all developers have the knowledge and time to follow the OWASP and other software vendor's security guidelines against SQLIV in a race against time to turnaround web applications.

3.3.1 Defensive coding/sanitisation approaches

OWASP [112] has provided white papers and guidance for building a secure web application to address SQLIV. The recent OWASP proactive controls white paper [113] gave ten recommendations to be included in every software development. The manual defensive coding usually referred to as code sanitisation involves parameterised queries or prepared statements, a whitelist of inputs, including data type validation and escaping of user input.

3.3.1.1 Prepared statements

Prepared statements include the use of parameterised queries in the variable binding to ensure the intruder does not change the intention of the intended query [112]. A .NET [114] implementation of a prepared statement or parameterised queries uses command parameters for all SQL queries call in the source code to accept the parameters in both queries and stored procedures.

3.3.1.2 Whitelist of inputs including data type validation

Whitelist input validation provides a mechanism to reject unexpected data input that could be malicious. The approach requires defined patterns of both malicious and valid web requests to be established as to be able to accept or reject legitimate web requests respectively. The OWASP white paper [115] provides prescriptive guidance on the whitelist of data input.

3.3.1.3 Input escaping

OWASP prescribed input escaping of all user-supplied input [112] to ensure that SQL characters in queries do not itself introduce SQLIV. To give an example, in tautological SQLIA type an intruder would exploit a single quote (‘) as used in a name O'Brien. Accepting a single quote by escaping of user input has different implementations depending on the database. On SQL server-based application will escape a single quote by adding a second single quote, e.g., O'Brien is escaped in a query (SELECT * FROM tblUser WHERE LoginName = ‘O’’Brien’). In a web application, the same is escaped with double quotes as in WHERE LoginName = “O'Brien”. In the paragraphs below, are some notable existing approaches on input sanitisations.

McClure and Kruger [111] developed a tool named SQLDOM based on the Call-Level Interface (CLI) [173] which provides automated input escaping and data type validation in addressing SQLIV. The solution consists of two parts with the first being an abstract object model that offers developers functionality to generate their schema, and the second being the generation of dynamic SQL statements at compile time against the generated database schema. The SQLDOM tool detects SQLIV if there is a mismatch between the dynamic SQL statements against the generated schema.

Thomas et al. [116] extended SQLDOM to automate prepared statements. Both above tools can be applied to new web application development, but the drawback is that they are lacking in retrofitting to legacy applications. Natarajan and Subramani [117] proposed SQL-IF which is a dynamic runtime tool checking web form input, but limited to special characters, keywords and Boolean characters.

Karakoidas et al. [118] applied a static analysis to generate prepared statements to mitigate SQLIV dynamically. The limitation that Karakoidas et al. have in detection is that prepared statements mitigation is only one of many methods in defence coding techniques to protect against SQLIA and does not offer a total solution in its entirety.

3.3.2 Discussion

There is a need to harness emerging computing in areas of ML and big data mining regarding security vulnerabilities lookups to address the limitations of traditional string lookups. Although defensive coding is plausible mitigation against SQLIV, the weakness is the technique employs traditional string lookups in input data validation to meet the ever-growing demands of internet applications big data. Also, the approach suffers a drawback of lacking retrofitting to legacy applications as a developer at design time needs access to the source code for implementing prepared statement and validation of escape characters.

Modern mission-critical business cloud-hosted applications and services do not allow the free flow of source code. However, this does not imply that traditional web applications protected within the organisation’s intranet (and now being deployed as cloud-hosted services) are free from SQLIV. In fact, most of these applications being ported to the cloud-hosted environment would contain a mixture of new web applications with code sanitisation and legacy applications that may still harbour SQLIV. Though defensive coding accounts for web input sanitisation, it cannot adequately deal with an interception of the web request for replay (malicious modification of the intercepted web request) by an intruder who employs SQL injection mechanism (conduit) other than web form input injection.

There is a need for organisations that host applications and services in the cloud to have a proxy layer to intercept web requests for SQLIV analysis. We propose in this thesis an approach that does not rely on the source code to detect and prevent SQLIA in input validation. We explore AI to predict SQLIA at the injection points by using a proxy to intercept web requests.

The section below discusses dynamic runtime approaches that involve generating static valid SQL queries which are dynamically matched at runtime against web requests to detect SQLIA. Also explored in dynamic runtime are profiles [121] and query sizes [122].

3.4 Dynamic runtime analysis for detection and prevention of SQLIA

Figure 3-4 presents this section organisation chart with the associated subsections.

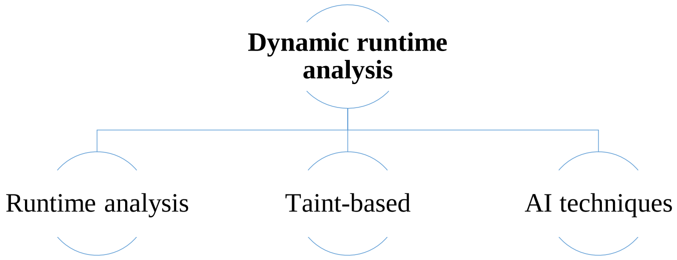


Figure 3-4 Dynamic runtime analysis techniques category chart: The figure shows this section organisation chart with the related subsections.

There are several existing techniques proposed by researchers under dynamic runtime analysis, which are to be discussed below. The dynamic runtime analysis is usually preceded in most approaches by statically generating all possible valid queries to a web application and matching the web requests at runtime against these valid queries to detect an anomaly or SQLIA. Also, included is the tainting techniques which involve the labelling of data which is then tracked to the sink (database entry). In the context of emerging computing and big data mining for SQLIA, we also discuss existing ML techniques including the technique proposed in this thesis to predict and prevent SQLIA.

3.4.1 Runtime analysis

Boyd & Keromytis [84] proposed SQLrand that employs proxy and parser whereby a query is randomised and de-randomised at the proxy on the assumption that an attacker will create invalid queries to cause runtime errors which will identify the SQLIA. The solutions were designed to fit into both new systems and can also be retrofitted into an existing web application for deployment. The authors claimed the methodology to have a negligible impact on performance; but, it is unlikely that it would be scalable in today's large data-driven web applications. Also, the process of randomisation was simplistic as the string appended to the SQL keywords could easily be replicated by an attacker.

Valeur et al. [121] proposed anomaly based system approach that learns and score profiles of the standard database access performed by web-based applications. The web requests or dynamic queries are intercepted where input vectors are inserted into tokens marked as a constant and compared by using statistical scoring methods. The learning phases are sectioned into two halves. The scores of the second half are then calculated with values exceeding the threshold set in the first half deemed anomalous. The authors evaluated the approach to have a high degree of success in the detection of SQLIA. However, the limitation is that it requires extensive training for excellent detection capabilities [82].

Buehrer et al. [4] proposed a SQL parsing tree which uses a combination of proxy and SQL parse tree for a genetic algorithm SQL syntax sequence alignment for detection of SQLIV. Though this approach applies gene sequencing alignment techniques, it has a similarity with valid queries matching techniques to establish SQLIA detection.

Halfond & Orso [5] proposed AMNESIA, a hybrid of static and dynamic approach that employed pattern matching between valid requests against dynamic web requests at runtime to detect and prevent SQLIA. Although the approach scaled well in traditional string matching at the time, it is unlikely that the method could meet the demand of emerging computing in big data scenarios that would require predictive analytics techniques for big data mining.

Wei et al. [123] proposed matching a static valid web application's SQL query against a dynamic analysis of queries at runtime to detect and prevent stored procedures SQLIA type. The drawback to this approach is that it is limited to only stored procedure detection SQLIA type and does not cover the other SQLIA types as discussed in Chapter 2. To address this limitation, Karuparthi & Zhou [91] presented an enhanced version of a dynamic query matching method [121], [122] to detect other SQLIA types. However, the enhanced model is a traditional string lookup technique which will not be functional in internet big data analysis.

Bisht et al. [11] proposed an approach named CANdidate evaluation for Discovery Intent Dynamically (CANDID). The approach was centred on the Java-based web application code analysis by mining programmer intended query structure of any input and comparing it against the structure of the actual query as to detect and prevent SQLIA. The benefit claimed by the authors of the approach is that it can be retrofitted, but the bytecode transformation could be computationally expensive to scale well in big internet data-driven traffic. Also, there are restrictions to source code access in the cloud-hosted applications to generate bytecode.

Fu and Li [125] proposed a string constant solver algorithm named SUSHI employing an automata-based approach to solve a simple linear string equation for attack signatures and error traces thereby exposing SQLIA and Cross Scripting (XSS) vulnerabilities. The approach reduces false positives in string analysis techniques, but it is computationally expensive to scale well in current big data-driven web applications.

Fernando and Abawajy [119] applied both static and dynamic approaches for detecting and preventing SQLIA in Radio-frequency identification (RFID). The drawback to this method is that it is not web-agnostic, which most modern RFIDs support.

Shahriar and Zulkernine [14] proposed an information theory based detection framework for SQLIV prediction. The approach applies static and dynamic script analysis of the query to calculate the entropy to detect malicious queries. The approach is not robust to dynamically detect all SQL types like stored procedure attack.

Jang & Choi [122] presented an approach for detecting SQL injection attacks using query result size. The approach employs a technique that dynamically analyses the developer intended query result size of the given input and then detects attacks by comparing this against the product of the actual query. The implementation includes running input against two layers: substitution variable verification and Query Cost Estimator (QCE). The replacement variable verification checks the attack patterns and vulnerabilities in the SQL queries. The QCE computes the predicates in SQL queries and estimates the table and column cardinalities in addition to estimating and comparing query result sizes. The drawback of this approach is a limitation in complex relationship queries that use an inner join which compares the result size, meaning that it will yield high false positive rates.

Kar et al. [120] proposed SQLiDDS; an approach that consists of offline and runtime. During the offline, the tail-end (after WHERE clause) of suspected queries are extracted and transformed to compute MD5 hash value stored in a reference hash table which uses hierarchical agglomerative clustering employing the averaging link method to aggregate related queries into a single document of highly related clusters. The transformation scheme where the comparison between static and dynamic query is computed by counting words could yield a false result as there are possibilities to arrive at the same number of words in the computation schemes.

3.4.2 Taint-based

Wassermann and SU [126] proposed a static algorithm for string analysis to find SQLIV. The algorithm statically generates a set of possible database queries that a web application may generate by employing context-free grammars which are then tracked by information flow from untrusted sources to these queries to detect vulnerabilities. It is a taint technique of tracking data flow from untrusted sources, but its drawback is the high false positive rates [82].

Halfond et al. developed a tool named Web application SQL Injection Preventer (WASP) [7] that is an attack runtime prevention method employing positive tainting of tracking trusted data as against traditionally negative tainting of tracking untrusted data. Though taint approaches with trusted data are fraught with the risk of increased false positive, the author deemed the proposal more plausible than related approaches which saw high false negative rates of tracking a taint data with untrusted data source employed in these cited papers [165], [174]. The WASP improvement in syntax-aware evaluation used a database parser to interpret the query before it is executed and provided a flexible mechanism that allows different trust policies to be associated with various input sources.

Kie et al. [127] developed a tool named ARDILLA for SQLI and XSS vulnerabilities detection through a technique of input generation, dynamic taint propagation and input mutation to detect vulnerabilities. ARDILLA automates the creation of test inputs to reveal SQLI and XSS by employing tainting techniques; the test input is symbolically tracked as it executes to mutate the inputs to produce vulnerabilities. The tool had few false positive rates, and addressed second-order vulnerability as claimed by the authors, but a taint technique that employs tracking of data from untrusted sources to detect vulnerabilities are known for high false positive rates. A further drawback to the use of this approach in modern web applications is that the flow of data to applications is so diverse that there can be no guarantee of receiving data from only trusted sources.

The section below discusses ML approaches to SQLIA detection and prevention, which is also the subject of the thesis presented in this thesis. After a careful study of the literature reviewed in the sections above, we deemed ML is a plausible approach to mitigate SQLIA in emerging computing.

3.4.3 Machine Learning and Neural Network runtime analysis

This method applies artificial intelligence which requires a robust data set with different patterns in data items to train a classifier. Applying ML requires robust data set items with patterns to train a classifier implementing AI algorithms to predict SQLI accurately. Unfortunately, as there is no single standardised data set that can be used in training an AI model for all scenarios of web application vulnerabilities, researchers have presented various approaches for extracting data sets with most proposals suffering from limited data engineering (data set with patterns to enhanced ML prediction). The HTTP dataset CSIC 2010 [66] does not present a robust, versatile pattern-driven data set as it was a purpose-built data set following a classic data set extraction using automated tools to log web requests with repeating query strings and occasional variation in query strings. These data sets are repeatedly found in research work as test data for evaluations and have come under scrutiny over the years of being dated and irrelevant to building a real business application.

There have been various ML algorithm proposals by researchers over the years to mitigate web applications SQLIA vulnerabilities. We reviewed literature on the popular ML algorithms to detect and prevent SQLIA which are: [71], [72], [74], [128]–[133], [100], [134]–[136],[137]–[141] to observe drawbacks in the data set devoid of patterns as they share a commonality in reliance on URL query strings, SQL internal query structure and static generated data set which is often repeated strings. We selected for further discussion some of these approaches employing SVM, LR and NN algorithms in subsequent paragraphs.

Bockermann et al. [75] proposed the tree kernels for analysing SQL statement in addition to exploring feature vectorisation of data input to an SVM classifier, but found there to be drawbacks in the tree-kernels computational overhead. Also, their data set extraction was dependent on URL strings which are repeating strings that are lacking in patterns.

Choi et al. [72] train an SVM classifier using feature vectorisation by N-Grams employing bigrams, trigrams and 4-grams for queries comparison. Their approach would need various patterns to improve the accuracy of the approach as the data set used has few distributions in the training data set of 608 attribute values (normal and malicious) and 306 attribute values of test data. Also, the approach requires access to full source code.

Kar et al. [137] proposed using a SQL query of normalised query elements to tokens to build a graph of tokens and centrality of nodes to train an SVM classifier, though an ML approach it mirrors a classical queries comparison between a normalised tokenised and runtime query to detect SQLI at database firewall. The approach looks promising in an intranet and extranet but not applicable to cloud-hosted web applications SDN endpoints. As the SQLIA mitigation is at the database level, the injected queries if not intercepted by other SQLIA protection during the transition are left to bypass firewall before be enumerated for tokens used in the ML query comparison approach. Also, this enumeration or tokenisation of schema changes takes 30 minutes, which will not scale in big data web traffic analysis.

Wang and Li [73] proposed an SQL query program tracing in which related queries are grouped based on the runtime program trace. However, the approach's drawback is the reliance on the source code for the SQL tracing grouping that is hashed to the vector matrices for a classifier implementing the SVM algorithm.

Pinzón et al. [74] present a multi-agent approach that uses various classifiers including SVM and neural networks to predict SQLIA from SQL queries behaviour that is stored as cases. Though an ML approach, it mirrors a classical queries comparison. The architecture named CBR cycle is computationally expensive and needs access to source code.

Kim and Lee [138] proposed an approach that uses the SVM classifier for binary classification of internal query representation known as query trees. The training data comprised of feature vectors from transformed query trees of the internal query structure. Its drawback is that it requires access to the source code and is computationally complex because of the size of the query trees.

We reviewed existing literature on SQLI to establish the research gap and defined a research goal to present in this thesis a supervised learning model that uses a data set input from patterns of both expected data and SQLIA types including SQL tokens to train classifiers. We started with an investigation of numerically encoding of both valid data input and patterns of SQLIA types including SQL tokens to test the performance metrics of vector matrices of data set input derived from patterns [61], [62]. We further our approach by applying ML predictive analytics to SQLIA prediction and prevention [63], [64]. The approach presented in this thesis relies on the web requests or data input at runtime to detect SQLIA as against raw queries comparison as proposed in most runtime analysis.

3.5 Conclusion

In the preceding sections, we reviewed existing research work proposed by various authors over the years on the topic of SQLIA mitigation. The existing literature is categorised as SQLIV testing and detection; defensive coding in web application code sanitisation for SQLI prevention; and dynamic runtime analysis. We can summarise our critique of these existing works as follows:

- They were SQLIA mitigation approaches that existed before the emerging computing of cloud-hosted web services where the traditional web server no longer resides on the organisation's intranet or extranet.
- The existing approaches displayed a need for pattern-driven data set to train the classifier in AI approaches as against repeating strings of SQL queries and URL query string.
- The available antiquated data sets of ECML/PKDD 2007 [65] and HTTP CSIC 2010 [66] on SQLI were purpose-built not idea to build mitigation against SQLIA in real-life scenario business application.
- The reliance on source code access which is now restricted in cloud-hosted services.
- The dependence on static code access scanning for SQLIA mitigation in most existing work as against the need for web requests to cloud-hosted services to be intercepted for analysis where access to source code scanning is restricted.
- The use of the traditional string lookup approach which is not known to be scalable to SQLIA mitigation when compared with using AI platforms in emerging computing big data.

On account of the literature review of the existing work, we identified the dynamic runtime analysis that applies AI techniques provided a functional and scalable approach to SQLIA mitigation in emerging computing. The few existing works [72]–[75], [136] that do employ SVM are lacking in data engineering (text pre-processing). Also, to date, none has discussed applied ML in predicting SQLIA in a context of big data, focusing on patterns and text pre-processing including empirical evaluation on MAML platform.

Also, an old competition-driven data sets were only good for the detecting the anomalies in the competition in which they were generated. These academic competitions-driven data sets do not benchmark performance metrics for an AI trained model in the real-world diverse web applications. On account of the lack of all-purpose data set of the diverse web applications that exist; there is a need for a paradigm shift from relying on old competition-driven data sets to a pattern-driven data set. The pattern-driven data set which varies in web application types is generated from the pattern that exists in the expected legitimate web request input, including illegitimate web requests of

SQL attack signatures and tokens to a web application which is one of the novel contributions of this thesis.

We focused on existing work by Choi et al. [72] that shares a similarity in vectorisation. While Choi et al. hashed URL query strings using more than one N-Grams (number of words in a string) to obtain vector matrices as to train an SVM classifier, the experiment presented in this thesis employs vectorisation to hashed the pattern-driven data set using unigram (a unit of word in a string) to obtain vector matrices. These vector matrices are used to train various classifiers (SVM, LR and ANN). The N-grams of more than one word was applied by Choi et al. in vectorisation could introduce prediction errors as an intruder may try to circumvent the SQLIA mitigation with SQL token comments and white spaces. We examined hashing more than one word in a group of strings and observed unigram of hashing word by word offers a minimal prediction error.

We suggest patterns exist in any data input to a web application to generate pattern-driven data set and by applying text pre-processing to such learning data improves the prediction accuracy of the resultant trained model. To further our approach, we applied ML predictive analytics to SQLIA for prediction and prevention of malicious web requests being fulfilled and this is the subject of discussion in the successive chapters.

4 Numerical Encoding to Tame SQLIA

Figure 4-1 presents the organisation chart layout of this chapter with the sections.

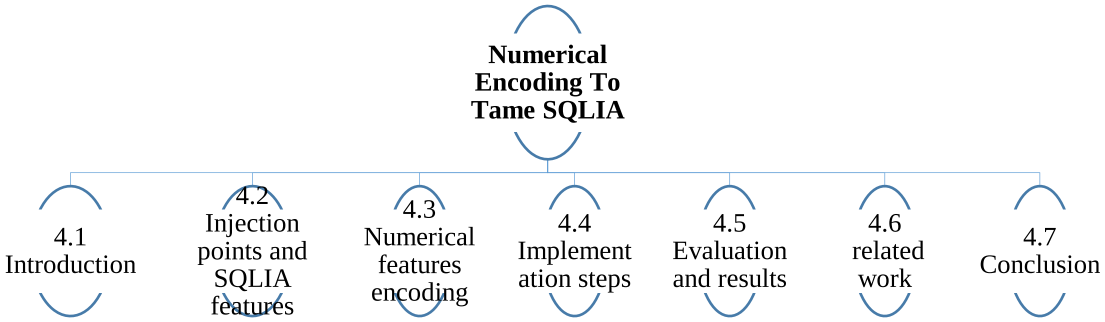

Numerical Encoding To Tame SQLIA
4.1 Introduction
4.2 Injection points and SQLIA features
4.3 Numerical features encoding
4.4 Implement ation steps
4.5 Evaluation and results
4.6 related work
4.7 Conclusion

Figure 4-1 The chapter organisation chart: The chapter organisation chart presents the layout of the chapter's sections.

4.1 Introduction

Figure 4-2 presents this section organisation chart with the associated subsections.

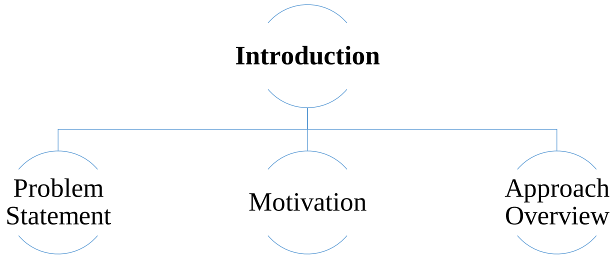


Figure 4-2 The section organisation chart: A figure of the section organisation chart that provides the layout of this section with the associated subsections.

This chapter answers the research question of can a pattern-driven data set be derived and validated from the expected legitimate web requests including illegitimate SQL tokens and injection signatures? The ML technique requires data set with numerical attributes as input to train a supervised learning model. The procurement of such historical learning data set vectors is straightforward in the natural sciences where observations and measurements over time are recorded to produce a large volume of data set with labelled feature vectors. Such historical data sets containing collated labelled observations of feature vectors have been made available by the University of California, Irvine (UCI) ML repository which currently maintains about 438 data set that is widely used by the ML community [3], [158] with the number of available data sets continually changing.

Unfortunately, the SQLI related data set that exist are purpose-built for competition [65], and the HTTP CSIC 2010 data set [66] is motivated by the outdated KDD CUP 1999 [69], [161]. Because of the lack of all-purpose data set suitable to train an AI model to mitigate SQLIA in the diverse web application types that exist; there is a need for a paradigm shift from relying on antiquated competition-driven data sets to a pattern-driven data set inferred from the desired web application type context that the SQLIA mitigation is being provided. To derive a pattern-driven data set, the human system expert providing SQLIA ML mitigation need to know the web application type domain and the expected input type as to produce relevant data set accordingly.

In this thesis chapter, we investigated the issue of availability of historical learning data set to train an AI model. We observed limitations exist in a purpose-built competition-driven HTTP CSIC 2010 data set [66] sponsored by the Spanish Research National Council motivated by the outdated DARPA KDD CUP 1999 [69], [161] which was used to test a department web application vulnerability. These data sets are out of date, especially in the context of emerging computing of big data and cloud-hosted services predictive analytics to address SQLIA issue. Also, HTTP CSIC 2010 data set is in the Spanish language which also implies doing some language translation pre-processing, if it is to be meaningful to a non-Spanish speaker to make sense of the data set. In our first approach to obtain a data set where none exist to train a classifier, we explore R language string replication, distance and RegEx patterns to derive a data set containing patterns of expected legitimate web requests, SQL tokens and known SQLIA type signatures.

ML falls into the broad area of AI. ML is a technique employed in data science to provide input of historical learning data to computer algorithms to predict outcomes. In applying ML, there is a need for the availability of historical learning data. The historical learning data are often referred to as a data set. The data set would contain columns and rows of data, termed as attributes or independent *x*-variables.

Figure 4-1 above provides the layout organisation chart of this chapter. This chapter starts with the introduction in section 4.1 and ends with a conclusion in section 4.7. Sections 4.2 discusses SQLIA injection points and features while section 4.3 presents the numerical encoding techniques of features. Section 4.4 discusses the implementation steps in MAML studio. Section 4.5 discusses the evaluation and the results while section 4.6 discusses the proposal in contrast and comparison to existing work.

4.1.1 Problem statement

In our research question, we ask if a pattern-driven data set can be derived from a web application type to be protected using the features artefact of the expected legitimate web requests, including illegitimate SQL tokens and injection signatures, and if the derived pattern-driven data set can be validated?

OWASP developer-centric SQLIA mitigation techniques require access to source code during the SDLC which include sanitisation of input in every tier (web client and server) [113]. These input validations and string lookup are techniques that were scalable within the organisation's intranet application domain boundary where there is no source code access restriction. Access to the source code for validation is an issue in emerging computing with a shift in the organisation's application domain boundary because of cloud-hosted services and cloud SDN where there is a restriction to source code access.

The magnitude of data will only increase, and an intruder may exploit SQLIA types with many signature variations to evade pattern matching techniques, e.g. SQLIA type tautological attack of *1=1* can also be written as *1 > 1, 'a'='a'*, etc. [157] to achieve the same attack. These variations in new SQLIA attack signatures could create significant issues for signature-based detection and prevention methods in recognising new signatures that intruders continuously evolve. In this thesis chapter, we account for these variations in SQLIA types with the assignment of random decimal attribute values named rndrisk in the generation of large data items of any size.

There is a strong argument for a bio-inspired approach that is functional in big data mining as against the existing traditional strings look-up in signature methods which are not known to be scalable. Applying AI approach is faced with the lack of existing data set to train a supervised learning model which is addressed in this chapter.

4.1.2 Motivation

The recent advancement in AI with readily available web platforms like MAML [41], [175], Amazon Web Services (AWS) [176] and IBM Watson Analytics [177] has refocused AI platforms as an enabling tool to build smart prediction models. These trained models are consumed by real-world applications in real-time to solve classification problems. This recent advancement in emerging computing has changed the perception of ML domain from the traditional confines of scientific research laboratories to an enabling ready to deploy tool for data mining. The better understanding of AI techniques in recent years has rejuvenated data science in the ability to train a classifier from a featured engineered pattern-driven data set to a ready to deploy the web application to be consumed as a Web Service (WS) available in MAML studio. ML models require data set of vector matrices as learning data which unfortunately is not readily available in securing web applications.

Applying ML approaches can adequately protect against new SQLIA signatures by classifying new attack signatures not trained for as unknown and thereby dropping or referring such requests at the interim. Exploring this method has met with the issue of lacking an existing robust data set to train a supervised learning model which is addressed here as one of the contributions of this thesis.

In a big data scenario, where string lookup approaches are not known to be a scalable option, and there is no pre-existing data set to apply AI methods; there is a need for a technique to generate significant data set of any size for SQLIA prediction of any magnitude. Also, the need for an empirical evaluation of the learning data to validate its suitability for SQLIA detection and prevention is also presented in this chapter.

4.1.3 Approach overview

In addressing non-availability of the data set in SQLIA detection and prevention domain, we draw our inspiration from the approaches at the UCI. The UCI maintains a repository of data sets which are a collection of observations and scalar measurements (labelling and numerical encoding of features) over a period to build a historical learning data. These available data sets from UCI are contributed to the scientific ML community and are used in training classifiers to address the real-world issue that needs cognitive automation.

The Wisconsin breast cancer data set (commonly used by researchers in ML binary classification) is one of such clinical observation examples. The data set contains *699* patterns defined by nine attributes with an integer value range from *1-10* which determined the labelling of being benign (*458* patterns) or malignant (*241* patterns) [158], [178]. Unfortunately, up till now, no comparable data set exists for SQLIA features. The SQLI related data sets that exist are purpose-built for competition [65]. Also, the HTTP CSIC 2010 data set [66] (motivated by the out of date KDD CUP 1999 data set) was purpose-built and tested on a department web application vulnerability.

Thus, that begs the question if the pattern that exists in web requests that are deemed valid, and historical studies of different SQLIA types including SQL tokens can provide artefact to be used to generate a data set to train a supervised learning model and be validated? In this chapter, we provide the detail of a pattern-driven technique for data set extraction. The extracted data set is used to train a supervised learning model which is empirically evaluated with statistical measures to gauge the performance metrics of the trained classifier. The successful outcome of the evaluation led to further development of this encoding by employing hashing or vectorisation of features to obtain the vector matrices needed to train a supervised learning model detailed in Chapters 5 and 6 of this thesis in obtaining massive learning data to mitigate SQLIA.

In creating the pattern-driven data set, we provide scalar observations to both patterns of legitimate input data (expected input data in web requests), SQL tokens and SQLIA types to derive a pattern-driven data set. These features are numerically encoded based on patterns of expected valid web requests, and SQLIA types including SQL tokens to obtain vectors require to train a classifier algorithm.

The scheme presented in this thesis uses, a web proxy to intercept web requests of any intent and applies supervised learning algorithms of a trained model to predict SQLIA at the SQL injection points. A web proxy is the most suitable to intercept web requests, including those originating from any injection mechanisms to SDN cloud applications' endpoints. An application-level proxy performs better in intercepting and decrypting of obfuscated web requests than low-level network packet interception tools which suffer from message fragmentation in a large volume of gigabits per seconds of packets in the wire. Injection mechanisms to a vulnerable application can originate from web page forms, second-order injection, exploiting web-enabled server variables, query strings, and through cookies.

An intruder would employ the following techniques to carry out SQLIA at the injection points in any combination. These SQLIA exploit techniques are Tautology; Union; Piggyback; Invalid/Logical queries; Time-based; Obfuscation encoding and Stored procedure. These SQLIA types discussed in 2.4 above are also a source of SQLIA positive labelled feature values in the scheme presented in this thesis.

4.2 Injection points and SQLIA features

Figure 4-3 presents this section organisation chart with the associated subsections.

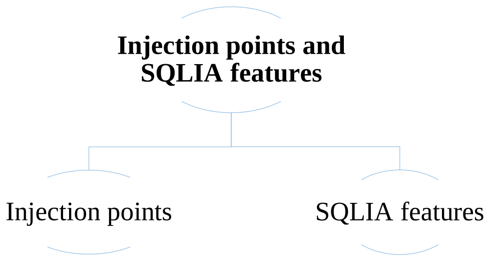


Figure 4-3 The section organisation chart: A figure of the section organisation chart that provides the layout of this section with the associated subsections.

4.2.1 Injection point

The WHERE clause controls the level of access to the back-end database from the front-end web applications. The scope of access could be any of validating a login account; filtering or setting criteria for the results from a database query; inserting a new record; deleting a record and updating an existing record. This injection point location after the WHERE clause has been explored to detect and prevent SQLIA [5], [9], [125]. In this research work, we analyse the intercepted web request substitution at the predicate and expression part of the SQL query structure for SQLIA. Figure 4-4 contains query strings which are intercepted for the field's input to be validated and extracted for analysis to predict SQLIA at the SQL query structure expression hotspot as shown in the figure with a red font.

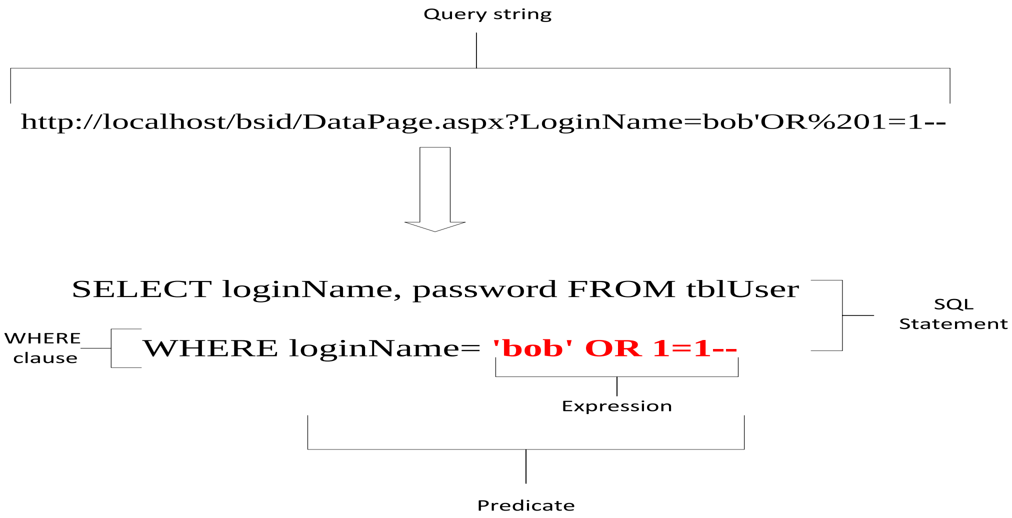

Query string
http://localhost/bsid/DataPage.aspx?LoginName=bob'OR%201=1--
SELECT loginName, password FROM tblUser
WHERE loginName= 'bob' OR 1=1--
WHERE clause
SQL Statement
Expression
Predicate

Figure 4-4 The injection point of SQL query structure: A figure of the SQL query structure expression hotspot which is the location for the SQL injection.

4.2.2 SQLIA features

A new web application does not have pre-existing robust data set of SQLIA features available to train a supervised learning model. We determined in this thesis that web application type, the research literature on SQLIA types and intent (as discussed in Chapter 2) combined with the presence of SQL tokens at the vulnerable hotspot provide a baseline for generating a pattern-driven data set of any size.

We explore these signatures usually injected into predicate’s expression highlighted in Figure 4-4 above. These attack features that substitute the expected valid input at the expressions can be special characters, SQL reserved keywords and known SQLIA signatures. Table 4-1 below are examples of symbols that are present in SQLIA signatures with much more that can be found in SQL reserved keywords [179] which are the subject of SQLIA positive attributes labelling in the numerical encoding of feature values. Table 4-2 illustrates some examples of SQL keywords that are being labelled positive when substituted into the SQL query expression.

Table 4-1 SQL Symbols: The table illustrates some examples of SQL symbols that are present in SQLIA signatures.

Symbol	Name	Symbol	Name
‘	Single Quote	|	Pipe
,	Comma	||	Logical OR
.	Period	%	Percentage
;	Semi-colon	?	Question Mark
:	Colon	&	Ampersand
“	Double Quote	&&	Logical AND
=	Equals	[	Opening Square Bracket
!	Exclamation	]	Closing Square Bracket
<	Less Than	{	Opening Curly Bracket
>	Greater than	}	Closing Curly Bracket
<=	Less Than Equal	\	Back Slash
>=	Greater than Equal	/	Forward Slash
!=	Not Equals	-	Dash/Minus
<>	Not Equals	--	Double Dash/Comment
+	Addition	)	Closing Parenthesis
-	Subtraction	\	Back Slash
*	Multiplication	(	Opening Parenthesis
\	Back Slash	/*	Comment Start
/	Forward Slash	*/	Comment End
-	Dash/Minus	/**/	Comment
--	Double Dash/Comment		

Table 4-2 SQL Keywords: The table illustrates some examples of SQL keywords that are being labelled positive when substituted into the SQL query expression.

SQL KEYWORD	SQL KEYWORD	SQL KEYWORD	SQL KEYWORD	SQL KEYWORD
ADD	EXTERNAL	PROCEDURE	CONTINUE	KILL
ALL	FETCH	PUBLIC	CONVERT	LEFT
ALTER	FILE	RAISERROR	CREATE	LIKE
AND	FILLFACTOR	READ	CROSS	LINENO
ANY	FOR	READTEXT	CURRENT	LOAD
AS	FOREIGN	RECONFIGURE	CURRENT_DATE	MERGE
ASC	FREETEXT	REFERENCES	CURRENT_TIME	NATIONAL
AUTHORIZATION	FREETEXTTABLE	REPLICATION	CURRENT_TIMESTAMP	NOCHECK
BACKUP	FROM	RESTORE	CURRENT_USER	NONCLUSTERED
BEGIN	FULL	RESTRICT	CURSOR	NOT
BETWEEN	FUNCTION	RETURN	DATABASE	NULL
BREAK	GOTO	REVERT	DBCC	NULLIF
BROWSE	GRANT	REVOKE	DEALLOCATE	OF
BULK	GROUP	RIGHT	DECLARE	OFF
BY	HAVING	ROLLBACK	DEFAULT	OFFSETS
CASCADE	HOLDLOCK	ROWCOUNT	SOME	TSEQUAL
CASE	IDENTITY	ROWGUIDCOL	STATISTICS	WHERE
CHECK	IDENTITY_INSERT	RULE	SYSTEM_USER	WHILE
CHECKPOINT	IDENTITYCOL	SAVE	TABLE	WITH
CLOSE	IF	SCHEMA	TABLESAMPLE	WITHIN GROUP
CLUSTERED	IN	SECURITYAUDIT	TEXTSIZE	WRITETEXT
COALESCE	INDEX	SELECT	THEN	PIVOT
COLLATE	INNER	SEMANTICKEYPHRASETABLE	TO	PLAN
COLUMN	INSERT	OUTER	TOP	PRECISION
COMMIT	INTERSECT	SEMANTICSIMILARITYTABLE	TRAN	PRIMARY
COMPUTE	INTO	SESSION_USER	TRANSACTION	PRINT
CONSTRAINT	IS	SET	TRIGGER	PROC
CONTAINS	JOIN	SETUSER	TRUNCATE	END
CONTAINSTABLE	KEY	SHUTDOWN	TRY_CONVERT	ERRLVL
DELETE	ON	UNION	ESCAPE	PIVOT
DENY	OPEN	UNIQUE	EXCEPT	PLAN
DESC	OPENDATASOURCE	UNPIVOT	EXEC	PRECISION
DISK	OPENQUERY	UPDATE	EXECUTE	PRIMARY
DISTINCT	OPENROWSET	UPDATETEXT	EXISTS	PRINT
DISTRIBUTED	OPENXML	USE	EXIT	PROC
DOUBLE	OPTION	USER	PERCENT	OVER
DROP	OR	VALUES	WHEN	WAITFOR
DUMP	ORDER	VARYING	ELSE	VIEW

4.3 Numerical features encoding

Figure 4-5 presents this section organisation chart with the associated subsections.

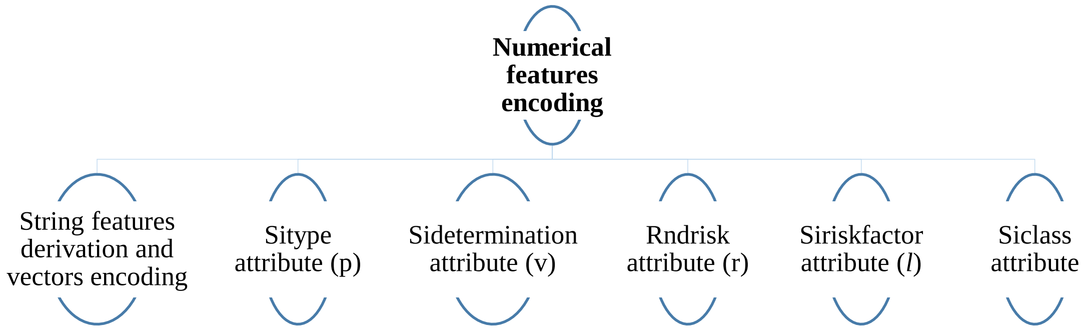

Numerical features encoding
String features derivation and vectors encoding
Sitype attribute (p)
Sidetermination attribute (v)
Rndrisk attribute (r)
Siriskfactor attribute (*l*)
Siclass attribute

Figure 4-5 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

A typical data set structure would contain columns and rows. The columns are often referred to as attributes while the rows are the attribute values. Also, there is a column that contains what to predict termed as the y-attribute with the content of the rows being the labelled y-attribute values. The labelled y-attribute values are the expected outcome to be predicted in a supervised learning model.

In ML approaches, scoping of the x attributes provides a method to capture the required artefact to predict the output y accurately which is the reason for the attributes scoping presented in the sections below. The ML algorithms required input type of numeric vectors which is the content of x-attribute values (data set rows to train a classifier).

The conventional approaches for extrapolating these numeric attribute values to train a classifier in biological sciences are through scalar observations and measurements. Take an example of the Iris data set (Anderson’s Iris data set) which Edgar Anderson collected the measurements of the sepals and petals of three Iris flowers to quantify the morphological variations of the related species [180]. This Iris data set has become a classic test case data set in ML statistical classification based on Fishers’ linear discriminant model [181]. The Iris data set has the name of the iris flower types as the labelled category and the associated attribute values (rows) containing the lengths of the sepals and petals measurements. The Iris data set procurement technique described above forms the idea behind the string features encoding presented in the sections below.

4.3.1 String features derivation and vectors encoding

In our first attempt to obtain a data set relevant to an intended web application as to provide an applied ML technique in SQLIA mitigation, we derived strings and the corresponding numeric scalar observations (vectors) as the attribute values in the data set used in this section. We further the approach in subsequent Chapters (5 and 6) to use string vectorisation as against manual vector assignments. Hashing or string vectorisation is a technique where string features are transformed into a numeric matrix required as input to a classifier to be discussed in Chapters 5 and 6. Also, we observed during the process of data set procurement that we could generate as much data set to assert SQLIA of any size.

The approach presented here uses ML techniques integrated with a proxy API to intercept web request as to predict SQLIA at SQL injection point (hotspot) which is a paradigm shift from SQL queries structure comparisons to some extent in most existing approaches. In inferring a data set ahead for SQLIA ML mitigation to a web application type, there is a need to have all possible permutations of features as to derive a pattern-driven data set capable of training a classifier.

To obtain a massive data set (learning data) with patterns, we explore string package of the R programming language replication method and a package named stringi [38] to derive as many related member strings as required. We obtained significant learning data by methods of replicating and shuffling (string transposition) of the strings to generate all possible member strings including the anagrams of the original string given.

Take an example of an expected valid input of *bob,* which can be shuffled to have a derivation of *bbo, obb,* etc. However, as the transposition of strings presents more patterns of the original strings; this becomes an invaluable step to detect shuffled strings by an intruder to bypass mitigation being provided in the web request to circumvent the SQLIA mitigation at hotspots. For example, an intruder may intend to carry out SQLIA with a shuffled *select* SQL keyword, e.g., *select, seeclt, lesetc, lseect, cetlse, eetcls, etc* in place of expected valid input of *bob* at the SQL predicate's expression value.

There is a need to procure data set with the relevant artefact in thinking ahead of ML SQLIA mitigation for a real-world web application where there is no precedence of the data set. The web application types are so diverse and dynamic to rely on out of date competition-driven data sets [65], [66] that were only good for the detecting of anomalies in the competition and project for which they were generated. Having to start from scratch implies a requirement to have a learning data to train an intended classifier relevant to the web application type to be protected to require the following artefact.

- Patterns of expected valid input data.
- Patterns of established SQLIA signatures which the background knowledge of SQLIA types provides.
- Patterns of SQL tokens (the query keywords, constants, symbols and delimiter identifiers), like an intercepted query strings with SQL tokens, can only mean a likelihood to substitute valid input data at SQL injection hotspot to effect SQLIA.

An example of a web application logic expecting from a web form; a login name, but the query strings analysis presents SQL tokens as a substitution indicates an issue. We need to be able to predict such anomalies in a large volume of a web request in cloud SDN by generating massive learning data to train a classifier to the maximum with cross-validation.

Figure 4-6 below shows the semantics of how the numeric attribute values are extrapolated. In replicating the approach, depending on the web application type, the scope of the attribute depends on the desired artefact required to predict SQLIA accurately. The approach presented here provides a paradigm shift towards procurement of the data set based on the web application type at the point of need as against having a static data set that is likely to be antiquated over time. Below are the steps.

- The data set extraction starts depending on the web application type, a set of strings is generated. In the example provided in this chapter, a list of expected input strings to a web application are conceptually labelled SQLIA negative while SQL tokens including SQLIA type signatures are labelled positive. Alternatively, string pattern can be generated with RegEx.
- These original string features are duplicated by methods of replication and transposition to generate related member strings by using the R language replicate method and random transposition of R stringi library (stringi::stri_rand_shuffle). The features are labelled with SQL tokens and SQLIA type signatures assigned a numeric value of between *1-8* in labelling while valid input string is labelled *9*.

- We assign an attribute named rndrisk which is a random value of $0.0_o \leq x \leq 0.0_n$ to account for the variation in signatures that occurs in SQLIA types. It is observed with this rndrisk that we can even simulate SQLIA of any size without the string features.
- We further measure the string distance to establish related member strings anagram using R package stringdist [182]. If the qgram method value is *0*, it confirms the string is an anagram or transposed version of the original string.
- For SQLIA positive, the sum values are less than *9* which is assigned a value *1* while *9* and above is SQLIA negative which is assigned a value *0*. Table 4-3 provides a snippet of how the class labelling is assigned to craft the labelled data set vectors.

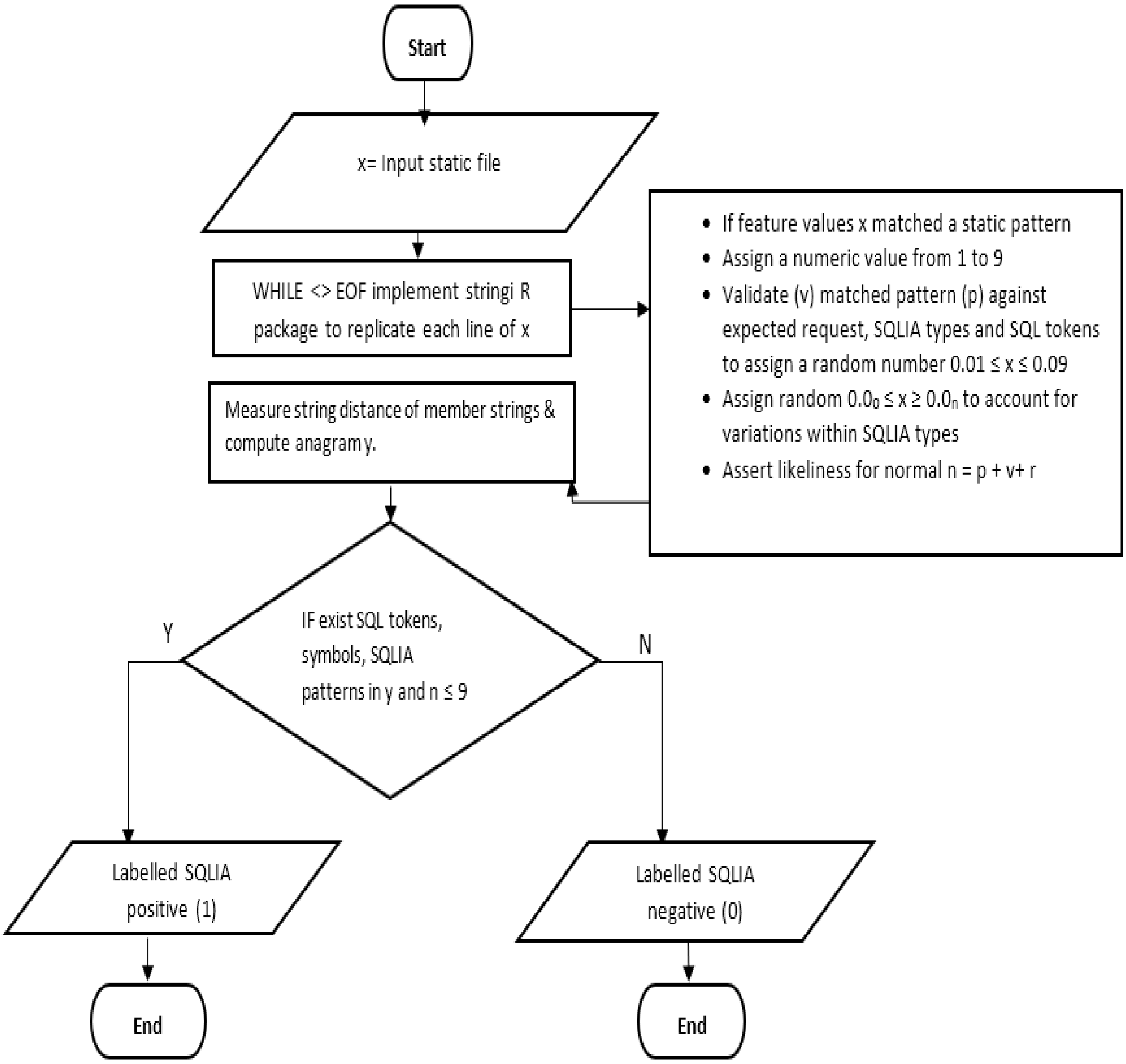

Start
x= Input static file
WHILE <> EOF implement stringi R package to replicate each line of x
Measure string distance of member strings & compute anagram y.
If feature values x matched a static pattern
Assign a numeric value from 1 to 9
Validate (v) matched pattern (p) against expected request, SQLIA types and SQL tokens to assign a random number 0.01 ≤ x ≤ 0.09
Assign random 0.0₀ ≤ x ≥ 0.0ₙ to account for variations within SQLIA types
Assert likeliness for normal n = p + v+ r
IF exist SQL tokens, symbols, SQLIA patterns in y and n ≤ 9
Y
N
Labelled SQLIA positive (1)
Labelled SQLIA negative (0)
End
End

Figure 4-6 A flowchart of the numerical encoding of features: A Flowchart illustrating numeric encoding of features to obtain scalar numeric values required to train a supervised learning model.

Table 4-3 Labels/classes to predict: A table illustrating the labelling of the SQLIA types including expected valid request.

SQLIA Types	**Class/Labels**	**Snippet of signature**
Tautology	1	`--,1=1,1<1, ‘a’=’a’ etc.
Piggyback	2	;
Union	3	UNION SELECT
Invalid/Logical Incorrect Query	4	SELECT, IF, ELSE, dbo,1/0
Alternate encoding obfuscation	5	), %, Hex values
Time-based	6	Wait for delay '00:00:10'
Store Procedure	7	; EXEC, XP_CMDSHELL, ATTRIB
Second order	8	‘x’=x’--"
Valid web request		
Expected collated input/non-SQLIA	9	Bob, bbo, obb etc

To simplify the sections explanation, throughout this chapter a few strings and SQL token features will be presented in the table examples. A set notation for many features attribute values (rows) or vectors will be represented with $\{x_o \ldots x_n\}$ for the list of items of vectors. We used the set notation in the table in the subsequent sections to represent the big picture of the attributes (columns) and associated attribute values (rows).

Table 4-4 presents set of attributes scoping with original strings that are replicated and transposed to obtain a derived member string which is then filtered to ensure the data set string features derived are patterns of related member strings (anagram). The subsequent sections would see a gradual build up of more attributes in the associated tables presented.

Table 4-4 Derived member strings: A table illustrating derivation of related member strings with the anagram value of *0* in the data set attributes scoping.

Original strings attribute	**Derived member string attribute values (String Features)**	**Related member strings (anagram)**
bob	bob	0
	bbo	0
	obb	0
select	select	0
	seeclt	0
	lesetc	0
	lseect	0
	cetlse	0
	eetcls	0
`	`	0
$\{x_o \ldots x_n\}$	$\{x_o \ldots x_n\}$	$\{x_o \ldots x_n\}$

4.3.2 Sitype attribute (p)

The sitype attribute $p = \{x_0 \ldots x_n\}$ represent encoded numeric SQLIA types, SQL tokens including a valid expected pattern of input data. Signature of SQLIA types with a combination of SQL tokens is assigned a number between one to eight *(1-8)* while a valid web request is assigned the value nine (*9*) and above. This encoding is aimed to provide a data set for supervised learning classifier with class attribute values that support both binary predictions of *1 / 0* and the multiclass classification of the SQLIA types.

The flowchart presented in the Figure 4-6 above illustrates the procedure for attributes extraction. If the static defined pattern (*p*) finds a match in the derived member strings and known signatures, then a number is assigned from a range of *1 to 9*. The value of nine (*9*) and above is a legitimate web request value assigned arbitrarily, while a value range of *1* to *8* is any of the SQLIA types including second-order injection mechanisms as highlighted and shown in Table 4-5. Refer to Table 4-3 above which provides a snippet of the translation of SQLIA types and SQL tokens including valid input in the labelling. The threat threshold is the cut-off value for labelling SQLIA risk or not. In this implementation presented in this thesis, it implies a threshold of collated value equal to or greater than nine will be labelled SQLIA negative feature value. Conversely, feature values assigned *1-8* and collated to be less than nine are labelled SQLIA positive.

Table 4-5 Sitype attribute vector assignments: A table illustrating the Sitype attributes vector assignments in labelling.

String Features	Related member strings	**Sitype (p)**
bob	0	9
bbo	0	9
obb	0	9
select	0	4
seeclt	0	4
lesetc	0	4
lseect	0	4
cetlse	0	4
eetcls	0	4
`	0	1
$\{x_0 \ldots x_n\}$	$\{x_0 \ldots x_n\}$	$\{x_0 \ldots x_n\}$

4.3.3 Sidetermination attribute (v)

The sidetermination attribute $v = \{x_0 \ldots x_n\}$ involves establishing if the patterns being matched are present in known SQLIA signatures; are patterns partly present in matched features; e.g. implementing NFA backtracking [183] for the tautological attack will be a litmus test for escape character (‘), OR, patterns of *1=1* and its derivations.

Sidetermination attribute is an NFA backtracking (cross-checking) to test if the signature of the assert SQLIA type exists to satisfy each assigned number of between a range *1-9* (refer to Table 4-3 above for the snippet of the vectors assignment). In obtaining the vectors in Table 4-6 below, we run a loop against the sitype, whenever a condition is satisfied for either of a valid web request or SQLIA types, a two decimal place value of $0.0_1 \leq x \leq 0.0_\mathrm{n}$ is assigned.

This feature encoding provides the maiden attempt to investigate if the artefact to provide SQLIA mitigation can be inferred from a web application type. In determining the SQLIA type, we take the closest match (similarity) of the feature being encoded. For example, the tautological attack will be a litmus test for escape character (‘), OR, patterns of *1=1*, etc. The result of the experiment being presented here led to a further work to replace this manual encoding to strings vectorisation (hashing) that is presented in Chapters 5 and 6. We recognised the limitation in emerging computing of cloud-hosted web applications with restriction to the source code in constructing the full lexical or syntactic structure of a SQL query to detect SQLIA. We went with the approach of a string by string analysis using ML techniques focusing on big data web traffic at the cloud SDN endpoints for pattern analysis of the permutations of SQLIA signature in transitions.

Table 4-6 Sidetermination attribute vector assignments: A table illustrating attribute vector assignments for Sidetermination.

String Features	Related member strings	Sitype (p)	**Si determination (v)**
bob	0	9	0.09
bbo	0	9	0.09
obb	0	9	0.09
select	0	4	0.04
seeclt	0	4	0.04
lesetc	0	4	0.04
lseect	0	4	0.04
cetlse	0	4	0.04
eetcls	0	4	0.04
`	0	1	0.01
$\{x_o \ldots x_n\}$	$\{x_o \ldots x_n\}$	$\{x_o \ldots x_n\}$	$\{x_o \ldots x_n\}$

4.3.4 Rndrisk attribute (r)

The rndrisk attribute (*r*) accounts for the variation that exists in both valid web requests and SQLIA types matching, e.g. an SQLI tautological attack of *1=1* can also be written as *1 > 1*, *'xyz'='xyz'* and so on, to achieve the same attack. Also, a valid data input expecting a pattern of *'bob'* can also accept a pattern of '*bbo'*. The rndrisk attribute is assigned a random value of $0.0_1 \leq x \leq 0.0_n$. These rndrisk random values account for the variations in SQLIA types when generating large data items. It is imperative the random number stay within the random values of $0.0_1 \leq x \leq 0.0_n$ as a large value will shift the threshold cut-off point. Table 4-7 presents the current progress of the attributes scoping with the rndrisk attribute values.

Table 4-7 Rndrisk attribute vector assignments: A table illustrating Rndrisk attribute vector assignments.

String Features	Related member strings	Sitype (p)	Sidetermination (v)	**Rndrisk (r)**
bob	0	9	0.09	0.08125
bbo	0	9	0.09	0.03384
obb	0	9	0.09	0.02436
select	0	4	0.04	0.08771
seeclt	0	4	0.04	0.03712
lesetc	0	4	0.04	0.06701
lseect	0	4	0.04	0.05996
cetlse	0	4	0.04	0.07636
eetcls	0	4	0.04	0.06436
`	0	1	0.01	0.05511
$\{x_o \ldots x_n\}$	$\{x_o \ldots x_n\}$	$\{x_o \ldots x_n$	$\{x_o \ldots x_n\}$	$\{x_o \ldots x_n\}$

4.3.5 Siriskfactor attribute (*l*)

The sirikfactor *l* is collated to obtain the likeliness of features being a risk factor that can either be *-1* likeliness or *1* for remote likeliness. The y-variables or what to predict (commonly known as a class) can be a possible *1 (1 0)* or *0 (0 1)* for SQLIA and valid web request respectively. It is the sum of x-attributes of the sitype (p), sidermination(v), rndrisk (r), for example (p + v + r) to determine if a data item is to be labelled an SQLIA positive or negative. Following this encoding scheme presented, more or other attributes deemed required depending on the web application type can be inferred as various web application types would vary in patterns that exist in them. However, we further our approach to replace these features encoding to features vectorisation or hashing in a further work presented in Chapters 5 and 6. Table 4-8 illustrates the current progress in the feature encoding with the highlighted Siriskfactor attribute values.

In Equation 4-1, let *i* and *n* be the size of attribute values; a sum of the total attribute *x*= (*p* + *v* + *r*) values less than *9* is a likeliness of SQLIA risk factor assigned *1*. Table 4-3 provides a snippet of how the class labelling is assigned to conceptualised the labelled data set vectors. Refer to row attribute values sum highlighted in red in Table 4-8.

$$l = \sum_{i=1}^{n} x = (p+v+r) < 9$$

Equation 4-1

In Equation 4-2, let *i* and *n* be the size of attribute values; a sum of the total attribute *x*= (*p* + *v* + *r*) value of *9* and above is a remote likeliness for SQLIA risk factor assigned *-1*. Refer to row attribute values sum highlighted in green in Table 4-8.

$$l = \sum_{i=1}^{n} \mathrm{x} = (p+v+r) \geq 9$$

Equation 4-2

Table 4-8 Siriskfactor attribute vector assignments: A table illustrating Siriskfactor attribute vector assignments.

String Features	Related member strings	Sitype (p)	Sidetermination (v)	Rndrisk (r)	**Siriskfactor (*l*)**
bob	0	9	0.09	0.08125	-1
bbo	0	9	0.09	0.03384	-1
obb	0	9	0.09	0.02436	-1
select	0	4	0.04	0.08771	1
seeclt	0	4	0.04	0.03712	1
lesetc	0	4	0.04	0.06701	1
lseect	0	4	0.04	0.05996	1
cetlse	0	4	0.04	0.07636	1
eetcls	0	4	0.04	0.06436	1
`	0	1	0.01	0.05511	1
$\{x_0 \ldots x_n\}$	$\{x_0 \ldots x_n\}$	$\{x_0 \ldots x_n$	$\{x_0 \ldots x_n\}$	$\{x_0 \ldots x_n\}$	$\{x_0 \ldots x_n\}$

4.3.6 Siclass attribute

The class attribute y-variable or what to predict can be a possible *1 (1 0)* for SQLIA positive or *0 (0 1)* for SQLIA negative. These attribute values are collated from the sum of independent predictor variables (x-attributes) of *p, v, r as* illustrated in Equation 4-1 and Equation 4-2. The dependent y-variable or what to predict (commonly known as a labelled class) is represented as a possible *1* for SQLIA or *0* for non-threat. Table 4-9 presents a complete conceptual labelled data set with attribute values that are required to be fed into MAML studio as a data set to train a supervised learning model presented in the subsequent sections.

We introduced this section with a motivation of the Iris data set that forms the idea behind the string features encoding presented. Note this conceptual maiden technique in to obtain a data set is replaced in Chapters 5 and 6 by string hashing vectorisation.

Table 4-9 Siclass attribute binary labelling: A table illustrating the encoding of strings and SQL tokens to numerical attribute values required to train a supervised learning model.

String Features	Related member strings	Sitype (p)	Sidetermination (v)	Rndrisk (r)	Siriskfactor (*l*)	**Siclass**
bob	0	9	0.09	0.08125	-1	0
bbo	0	9	0.09	0.03384	-1	0
obb	0	9	0.09	0.02436	-1	0
select	0	4	0.04	0.08771	1	1
seeclt	0	4	0.04	0.03712	1	1
lesetc	0	4	0.04	0.06701	1	1
lseect	0	4	0.04	0.05996	1	1
cetlse	0	4	0.04	0.07636	1	1
eetcls	0	4	0.04	0.06436	1	1
`	0	1	0.01	0.05511	1	1
$\{x_o...x_n\}$	$\{x_o...x_n\}$	$\{x_o...x_n$	$\{x_o...x_n\}$	$\{x_o...x_n\}$	$\{x_o...x_n\}$	$\{x_o...x_n\}$

4.4 The implementation steps on MAML studio

Figure 4-7 presents this section organisation chart with the associated subsections.

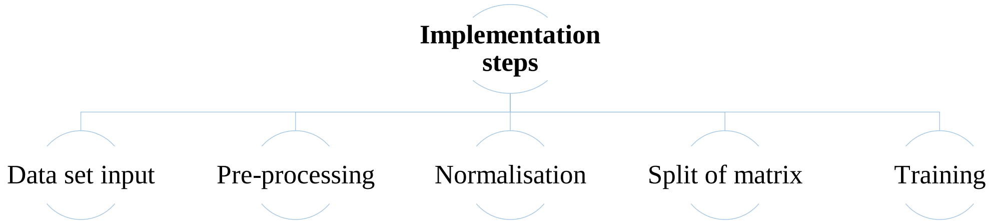
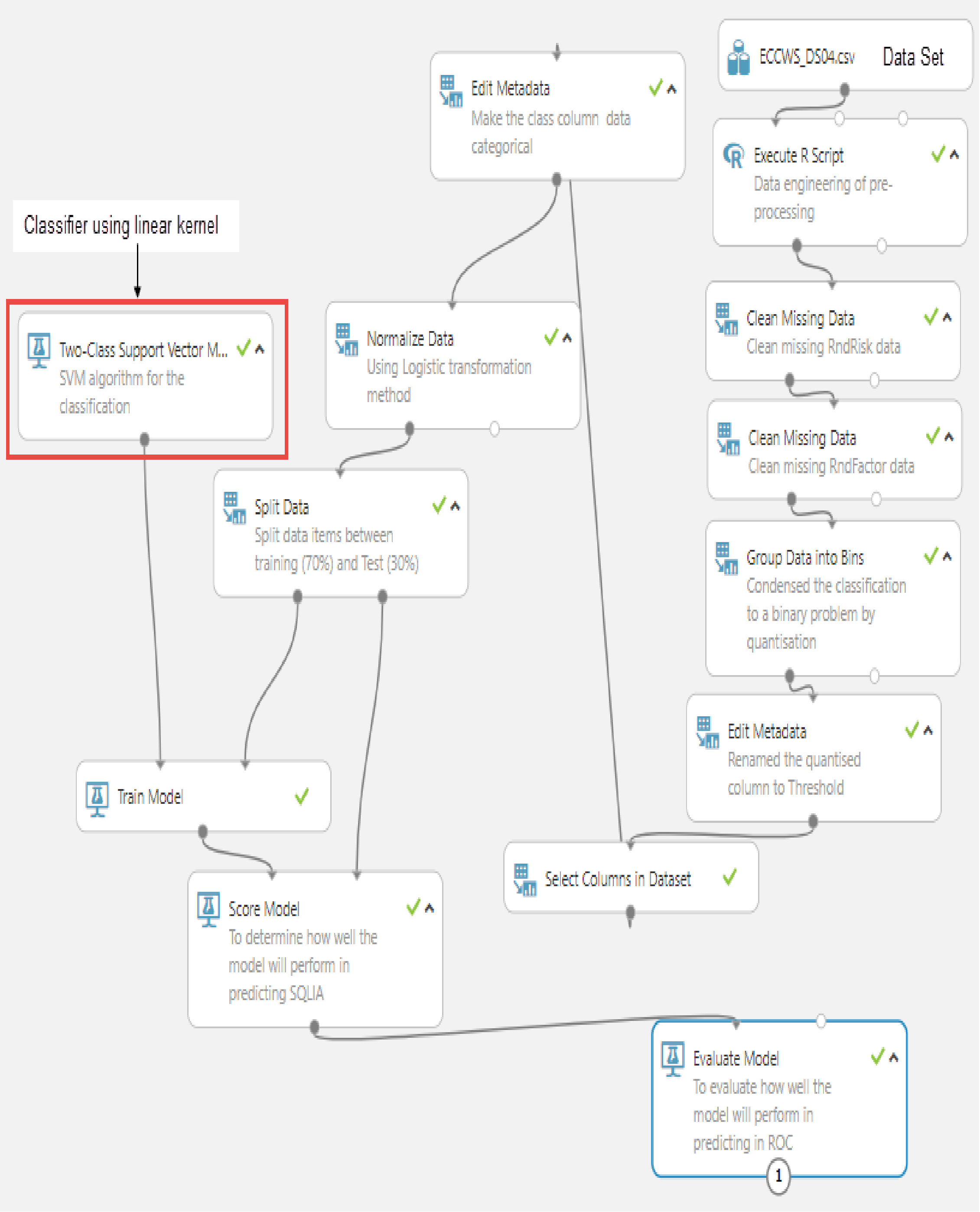


Figure 4-7 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

Figure 4-8 below is the MAML experiment screen capture that is carried out for the model proposed. The steps include: pre-processing of the data set attribute values;

classification of SQLIA features; validation of the supervised learning model which is then exposed as web services in ongoing detection and prevention. Below is the high-level overview of the key steps and with further detail presented in this section.



ECCWS_DS04.csv
Data Set
Edit Metadata
Make the class column data categorical
Execute R Script
Data engineering of pre-processing
Classifier using linear kernel
Two-Class Support Vector M...
SVM algorithm for the classification
Normalize Data
Using Logistic transformation method
Clean Missing Data
Clean missing RndRisk data
Clean Missing Data
Clean missing RndFactor data
Split Data
Split data items between training (70%) and Test (30%)
Group Data into Bins
Condensed the classification to a binary problem by quantisation
Edit Metadata
Renamed the quantised column to Threshold
Train Model
Select Columns in Dataset
Score Model
To determine how well the model will perform in predicting SQLIA
Evaluate Model
To evaluate how well the model will perform in predicting in ROC
1

Figure 4-8 The MAML studio predictive experiment: The figure is a screen capture of MAML studio to illustrates experimental steps for the proposed model.

1. The pre-processing of pattern-driven data set is done using R scripting language and RegEx patterns matching as detailed in 4.3.1. This step is followed by

scrubbing of the missing data and column casting of the input data set. Missing attribute values are aligned to the presence or absence of SQLIA risk in labelling to achieve better performance metrics that can accurately predict SQLIA. Data set scrubbing helps remove over-fitting from both binary and multiclass classification in the work presented in this thesis.

2. The next step is employing normalisation which is the rescaling of the numeric data as to constrain the vectors [184] for a proper fitting in the statistically trained model. In the scheme presented here, we rescaled the vectors of the presented pattern-driven data set from *[-1,9]* to a range of *[-1,1]* using z-score normalisation and *[0,1]* while using logistic normalisation to present a detailed comparison result in section 4.5.2 below.
3. The normalisation is followed by splitting of data between training and testing data. Through repeated training of the statistical model with different split data ratios, it is observed in the scheme presented in this thesis, employing *70% (0.7)* training data to *30 % (0.3)* testing data gave the highest prediction rates in the binary classification while *80% (0.8)* training data to *20% (0.2)* testing data gave the highest prediction rates in multiclass classification.

4. The supervised learning model was trained using the following classification algorithms:

   - Two Class Logistic Regression (TC LR).
   - Two Class Support Vector Machine (TC SVM).
   - Multiclass Neural Network (MNN).
   - Multiclass Logistic Regression (MLR).

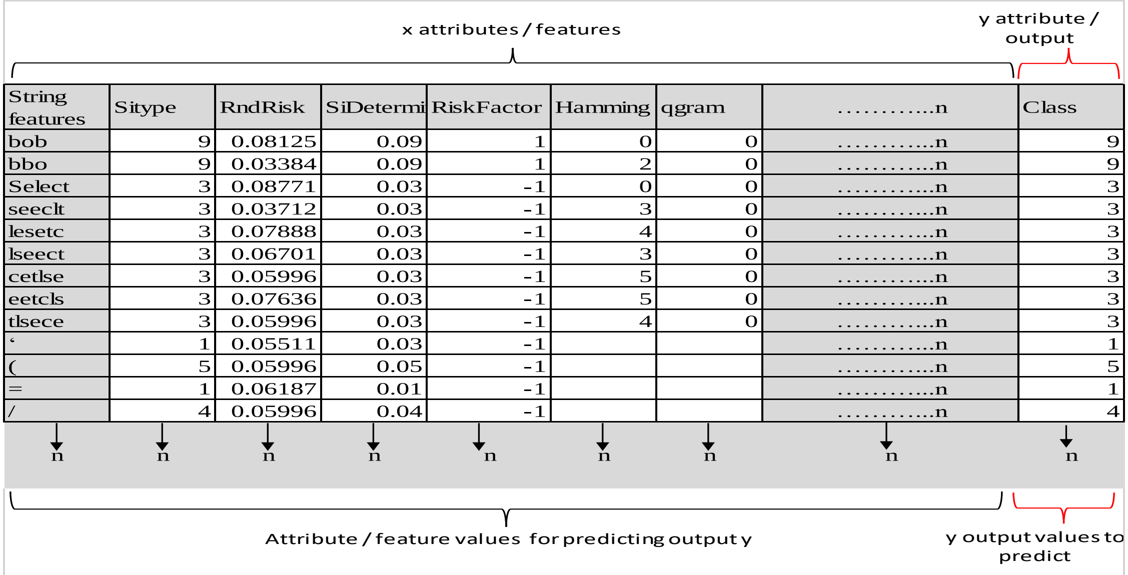
The MAML studio provides an easy technique to test drive classifiers by drag and drop with the promising algorithm validated and cross-validated for reduced biases to unknown independent data in the real-world. The cheat sheet [185] (How to choose a Microsoft ML algorithm) is an additional resource that gives prescriptive guidance in selecting a classifier to use. Both TC LR and TC SVM were used to determine which classification algorithm performs better in a binary classification of the pattern-driven data set. The trained models were validated and cross-validated with detail provided later in the chapter. We further the classification to use MNN and MLR to determine which classification algorithm performs better in SQL types multiclass classification.

5. These algorithms above were scored and empirically evaluated to establish how well the prediction of the supervised learning classification models is performing. The model with the best performance metrics in statistical measures (accuracy, precision, recall and F1score) including ROC curve, AUC, CM and k-fold cross validation is deployed as the predictive model.

6. Finally, a predictive web service was generated from the trained; supervised learning model built using MAML studio to obtain an API code to integrate into the Fiddler proxy API and web forms.

4.4.1 Data set input data items

The Figure 4-9 below displays a snippet of the attributes extrapolated as detailed in 4.3.1 above with data set containing *59702* data set row items (attribute values).


x attributes / features
y attribute / output
n
n
n
n
n
n
n
n
n
Attribute / feature values for predicting output y
y output values to predict

String features	Sitype	RndRisk	SiDetermi	RiskFactor	Hamming	qgram	...........n	Class
bob	9	0.08125	0.09	1	0	0	...........n	9
bbo	9	0.03384	0.09	1	2	0	...........n	9
Select	3	0.08771	0.03	-1	0	0	...........n	3
seeclt	3	0.03712	0.03	-1	3	0	...........n	3
lesetc	3	0.07888	0.03	-1	4	0	...........n	3
lseect	3	0.06701	0.03	-1	3	0	...........n	3
cetlse	3	0.05996	0.03	-1	5	0	...........n	3
eetcls	3	0.07636	0.03	-1	5	0	...........n	3
tlsece	3	0.05996	0.03	-1	4	0	...........n	3
‘	1	0.05511	0.03	-1			...........n	1
(	5	0.05996	0.05	-1			...........n	5
=	1	0.06187	0.01	-1			...........n	1
/	4	0.05996	0.04	-1			...........n	4

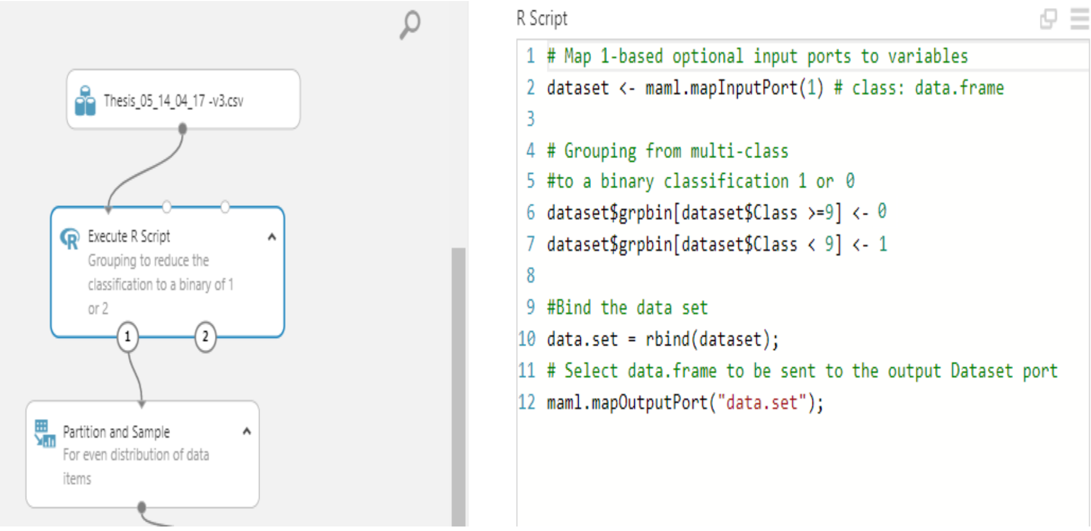
Figure 4-9 A snippet of numeric encoded vectors: A table showing a snippet of numeric encoded data set attribute values inferred from scalar observations of known valid web requests, SQLIA signatures and SQL tokens.

These extrapolated attributes serve as input vectors to a supervised learning model experiment carried out in the MAML studio to evaluate the performance metrics of applying the pattern-driven data set presented in this thesis. ML techniques require data set of historical attributes that have patterns to predict a predetermined labelled output, which in this case is whether a web request data input is SQLIA negative or positive.

4.4.2 Pre-processing or data engineering

In gathering raw data, there can be irregularities in the form of missing values, outliers and discrepancies [186]. Apart from cleansing irregularities in data set items, other data engineering steps can be carried out during pre-processing to improve prediction performance. These data engineering improvements could include: using RegEx constraints by implementing R-Script modules; grouping of the class y-attribute values to reduce dimensionality, e.g. multiclass to a binary classification; and, sampling for even distribution of attribute values.

We evaluated the suitability of numeric encoded vectors input data set in both binary class and multiclass classification in the ROC curve and CM respectively. We reduce the original multiclass labelled data set of encoded assigned values of *1-9* in y-attribute values to a binary class of *1-8* grouped as SQLIA positive with an assigned value of *1*, while *9* onward is assigned *0* as illustrated in Figure 4-10. Using the data set to train binary and multiclass classifier algorithms provided an empirical evaluation of the suitability of the learning data obtained from the scheme presented in this thesis to train the SQLIA prediction model.

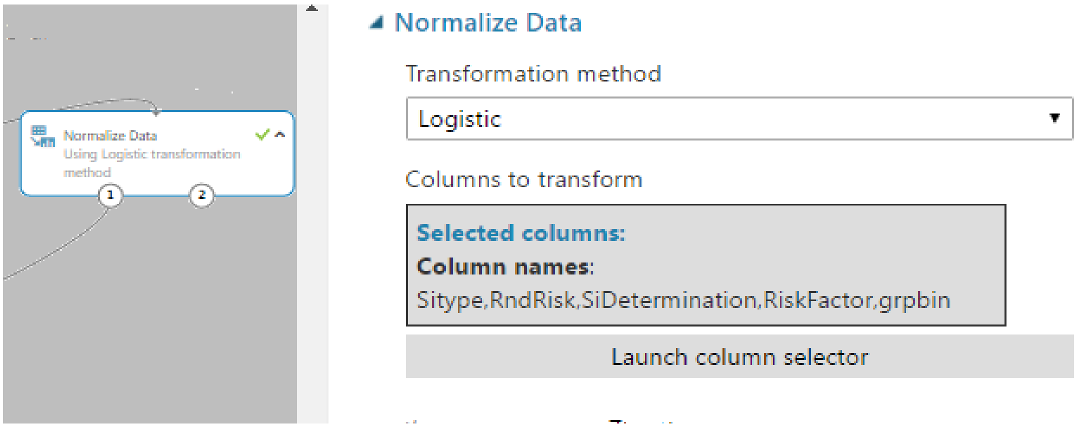

Thesis_05_14_04_17 -v3.csv
Execute R Script
Grouping to reduce the classification to a binary of 1 or 2
1
2
Partition and Sample
For even distribution of data items
R Script

```
# Map 1-based optional input ports to variables
dataset <- maml.mapInputPort(1) # class: data.frame

# Grouping from multi-class
#to a binary classification 1 or 0
dataset$grpbin[dataset$Class >=9] <- 0
dataset$grpbin[dataset$Class < 9] <- 1

#Bind the data set
data.set = rbind(dataset);
# Select data.frame to be sent to the output Dataset port
maml.mapOutputPort("data.set");
```

Figure 4-10 Grouping of multiclass to a binary class: A figure of screen captures of reducing multiclass labelled data set to a binary data set.

4.4.3 Normalisation

Normalisation scheme involves attribute values rescaling scheme in ML to standardised x-attributes value to a range of closely confined values. Normalisation provides standardising data or z-score which is a method to rescale or reduce numeric data items to a standard range [184]. In this experiment, we rescaled x-attribute values from *[-1,9]* to a range of *[-1,1]* using z-score normalisation and *[0,1]* while using logistic normalisation. Figure 4-11 illustrates normalisation using a logistic transformation method.



Normalize Data

Transformation method: Logistic

Columns to transform

Selected columns:
Column names: Sitype,RndRisk,SiDetermination,RiskFactor,grpbin

Figure 4-11 A MAML studio normalisation module to rescale attribute: A figure of the MAML studio normalisation module to rescale attribute values to a standardised range of value through logistic normalisation.

We rescaled the attribute values employing z-score and logistic normalisation modules that are implemented in Equation 4-3 and Equation 4-4 respectively.

Let *z* be the z-score or logistic normalisation (standardisation of data items)

- *x* is the attribute values or each data point observation
- μ is the mean (average) of *x*
- σ standard devaiation of *x*
- *e* is the exponential constant value of *2.71*

$$\mathrm{z} = \frac{(x - \mu)}{\sigma} \qquad \textbf{Equation 4-3}$$

The MAML studio offers various normalisation configurable properties. Equation 4-3 is the formula used in the z-score normalisation scheme which MAML studio offers. Let *z* be the normalised value, the normalisation module of the MAML studio takes the data point *x* minus the mean represented by $(x - \mu\ )$ divide by the standard deviation of *x* denoted by σ [184]. This calculation rescaled the attribute values (vectors) of range *[-1,9]* to vector values range of *[-1,1]*.

$$z = \frac{1}{1 + e^{-x}} \qquad \textbf{Equation 4-4}$$

Equation 4-4 is a logistic normalisation formula which is a configurable module in the MAML studio. Where z is the normalised vectors, using the mathematical integration equation, the original attribute values are rescaled from *[-1,9]* to *[0,1]*.

We evaluated the impact of z-score and logistic normalisation on prediction errors (FN and FP) to have observed the logistic normalisation improves the prediction rate as further presented in section 4.5.2.

4.4.4 Split of the vector matrices between training and testing set

We divided the data set matrices into different split ratios to determine the impact of the split ratio of the training to testing set has on the classifier performance. The first run was *70%* of the data set vector matrices were used for the training of the classifier, and the remaining *30%* were used as testing data. In the second run, we used a ratio of *80%* of the data set vector matrices for the training of the classifier, and the remaining *20%* were used as testing data. This testing data is the subject of the empirical evaluation presented in section 4.5 below. Also, we provided cross-validation, which provides a better gauge of the performance metrics with ten folds possible permutations of training to testing data partitions.

4.4.5 Training a supervised model

In this experiment, we trained binary classifiers named TC LR and TC SVM. We also trained multiclass classifiers (MNN and MLR) to provide evaluation results presented in 4.5. The binary classifiers predict one of possible outcomes of between SQLIA negative or positive, while the multiclass classifier is extended to predict the various SQLIA types including valid web requests.

Both TC LR and TC SVM algorithms employ linear kernel which offers a binary prediction at the proxy. The default kernel used in the experiment offers a linear separation between SQLIA positive and negative in web requests. This linearity of the classes which can be demarcated in a straight line makes algorithms (classifiers) using linear kernel a preferred choice in binary classification.

The two classifiers were observed with good accuracy and fast training times in performance metrics in using the linear kernel. Also, MLR and MNN classifiers offer us a method to predict different SQLIA types. TC SVM, TC LR, MLR and MNN algorithms are trained with the training data and evaluated with the testing data of the partitioned matrix values. The trained classifiers are evaluated in the ROC curve and CM with the better performing trained, supervised learning model under cross-validation deployed as a web service for the ongoing SQLIA detection at the web proxy.

4.5 Evaluation and Result

Figure 4-12 presents this section organisation chart with the associated subsections.

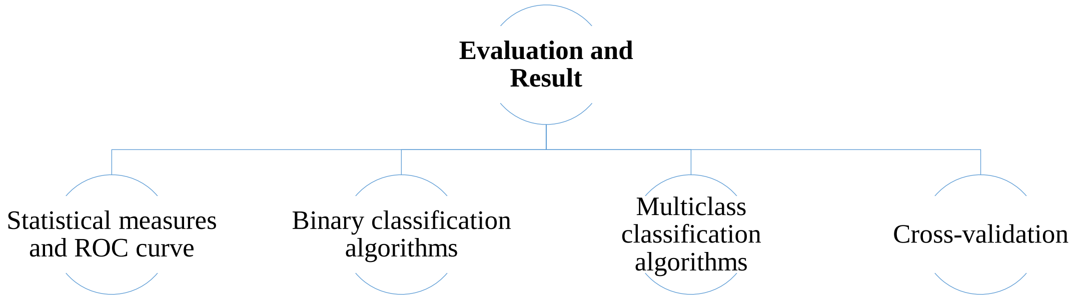

Evaluation and Result
Statistical measures and ROC curve
Binary classification algorithms
Multiclass classification algorithms
Cross-validation

Figure 4-12 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

The statistical measures (accuracy, precision, recall and F1 score), ROC curve and CM are widely used by data scientists to measure the performance metrics in ML analytics. It provides a valid empirical statistical measure for the evaluation of results. ROC curve, which has its origin in World War II to predict radar images for threats, has been widely accepted in interpreting medical test results [50], [51] and recently in computing data science.

The attribute values or x-variables (sitype, sidetermination, rndrisk, siriskfactor, etc.) including the labelled output attribute or y-variable (siclass) forms the input data set to a supervised learning model. ML approaches need an extensive data set of numeric encoded (vectors) from the behaviour or patterns of normal and SQLIA features for accurate prediction of SQLIA in a web request. The numerical attributes encoding ontology detailed in 4.3 above provides the technique to derive as many desired data set attribute values for the supervised learning model presented in this thesis.

The observations are inferred by scoring the model to generate score probabilities for the testing data (the split data unseen by the classifiers) against the training data used to train the classifiers. This section provides the evaluation and performance metrics of the model under various normalisation, training to testing split data ratios and statistical measures. Also presented in this section is the binary and multiclass classifier performance metrics in the ROC curve and CM respectively.

4.5.1 Statistical measures and ROC curve

Figure 4-13 presents this section organisation chart with the associated subsections.

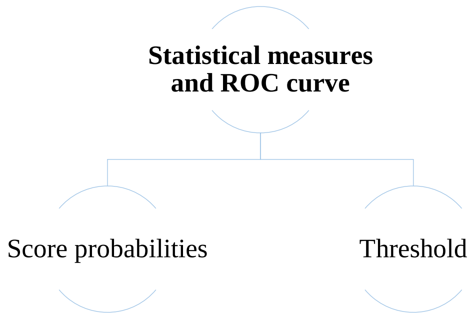


Figure 4-13 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

4.5.1.1 Score probabilities

The testing data input variables are scored to generate score probabilities of a range $0 \leq x \leq 1$ where x is the input vector values to predict output y. The score probabilities provide a measure of observations that are correctly predicted as TP and TN including the two prediction errors of FP and FN within the range $\{0,1\}$. These prediction observations of TP, TN, FP and FN are presented in the tables below, are calculated to determine how many of these observations score probability values fall within each score bins set of between 0 and 1 (expressed in the set as $\{0,1\}$).

Therefore, the expected output is $y = 0$ or SQLIA negative if the function of the predictor variables x is closer to 0 expressed as $f(x) \approx 0$. Conversely, the prediction output is $y = 1$ or SQLIA positive when the function of predictor variables x is closer to 1 denoted as $f(x) \approx 1$. Also, taken into consideration are the prediction errors rate of FP and FN. The prediction errors are observations of predicted output that have been wrongly classified which is used to gauge the performance of a classifier.

4.5.1.2 Threshold

The MAML studio sets by default the cut-off threshold for the prediction of TP and TN including the errors of FP and FN to be 0.5. This cut-off of 0.5 is a predetermined threshold employed by classification algorithms; it is a cost function of x to predict y.

Therefore $f(x) < 0.5$ score probability value is predicted as SQLIA negative (*0*) while $f(x) \geq 0.5$ is predicted as SQLIA positive (*1*). The score probabilities are partitioned into ten score bins of *0.1* increments of the set *{0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0}*.

The collation of the prediction variables of TP, TN, FP and FN falls on these cut-offs between the set *{0.0,1.0}*. These prediction observations are used to calculate the accuracy, precision, recall and F1 score statistical measures. Table 4-10 is a formula table that includes the abbreviation used in statistical measures throughout this thesis.

Table 4-10 Formula table: A table detailing how the performance metrics are calculated.

Terminology	Formula
True Positive (TP)	-
True Negative (TN)	-
False Positive (FP)	-
False Negative (FN)	-
Positive events (PE)	TP+FN
Negative events (NE)	FP+TN
Positive observations (PO)	TP+FP
Negative Observations (NO)	FN+TN
Total events (TE)	PO+ NO

4.5.1.3 Statistical measures

Figure 4-14 presents this section organisation chart with the associated subsections.

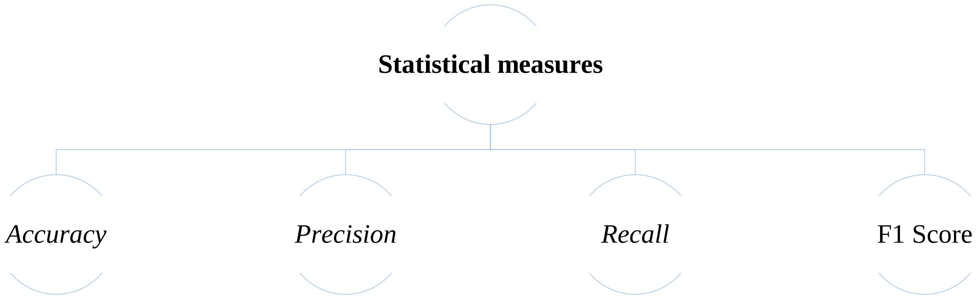


Figure 4-14 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

The statistical measurements of a trained model are provided by accuracy, precision, recall and F1 score. These statistical measures are calculated by collating the score probability values of TP, TN, FP and FN under various cut-off of *0.1* increments between $0.0 \leq x \leq 1.0$.

4.5.1.3.1 Accuracy

Accuracy is the proportion of the actual accurate results from the total cases. It has relevance in interpreting the performance of a model trained as it costs the same in a data set with an even distribution of features [46]. The accuracy is calculated by the formula elements in Table 4-10 above with the Equation 4-5 below.

Accuracy (A) = (TP + TN) / TE **Equation 4-5**

4.5.1.3.2 Precision

Precision is the proportion of true overall positive results returned by a trained model. In an even and uneven distribution of features, it will cost more in FP for a wrong prediction of a feature. In this scenario, it is the administrative overhead of wrongly predicting SQLIA positive. The precision is calculated by the formula elements in Table 4-10 above with the Equation 4-6 below.

Precision (P) = TP / PO **Equation 4-6**

4.5.1.3.3 Recall

The recall is the TP rate, which is the fraction of total correct results returned by the model. In an even and uneven distribution of features, it will cost more for prediction error of FN. In this scenario, it is the security loophole of wrongly predicting actual SQLIA as negative, which implies the system is unprotected against data pilfering with damaging ramifications. The recall is calculated by the formula elements in Table 4-10 above as the prediction rate of a trained classifier with the Equation 4-7 below.

Recall (R)= TP / PE **Equation 4-7**

4.5.1.3.4 F1 Score

The F1 score is a measure of accuracy that balances precision and recall. It has relevance in interpreting the performance of a trained model as it costs the same in a dataset with an even or uneven distribution of features [46]. The F1 score is calculated by the formula elements in Table 4-10 with the Equation 4-8.

F1 Score (F) =2 x (R x P) / (R+P) **Equation 4-8**

4.5.1.4 ROC curve and AUC

ROC curve is a graphical plot using variation in threshold discrimination to illustrate the performance of the binary classifier with a curve towards the upper left corner indicating a better performing model. AUC or area below the curve in the graph plot is a measure of TP Rate (TPR) or recall on the y-axis against FP Rate (FPR) on the x-axis. An excellent prediction model achieved through this thesis is interpreted from the AUC value of *0.9 ≤ x ≤ 1*. The ROC curve AUC provides a measure of the performance of a classifier which is graded as presented in Table 4-11.

Table 4-11 AUC grading: A table showing the grading of AUC that set the benchmark for the measure of performance metrics of a trained model [51].

AUC Values	Grade
0.9 ≥ x ≤ 1.0	Excellent
0.8 ≥ x < 0.9	Good
0.7 ≥ x < 0.8	Fair
0.6 ≥ x < 0.7	Poor
x< 0.6	Random prediction/worthless

4.5.2 Binary classification algorithms

Figure 4-15 presents this section organisation chart with the associated subsections.

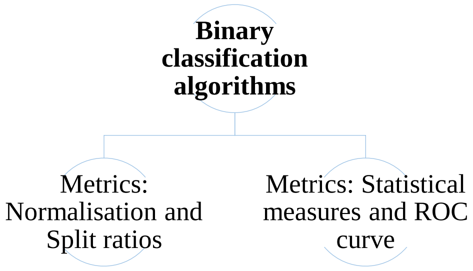

Binary classification algorithms
Metrics: Normalisation and Split ratios
Metrics: Statistical measures and ROC curve

Figure 4-15 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

In this implementation of applying ML to mitigate SQLIA, the ratio of SQLIA negative to positive is *1* to *8* in the data set used. We used feature values (*59702*-row items) that are split at various ratios between training to testing data. These partition data sets are used to train the classifiers named above. We split the data set matrices to ratios of *70%* (*41791* rows of training data) to *30%* (*17911* rows of testing data) for an experiment.

We also carried out a split ratio of *80%* (*47762* rows of training data) to *20%* (*11940* rows of testing data). In this experiment, there were majority SQLIA positives over negatives in binary classification, but a balanced data set in multiclass across the nine classes of the features; as the same data set was used in the multiclass classification of SQLIA types. However, the imbalanced data set (majority SQLIA positives over negatives) were corrected in future binary classification using Synthetic Minority Over-

Sampling Technique (SMOTE) [187] in 5.3.2. These results obtained from the experiments using various split data ratio are presented below. The original data set created as input to the multiclass classifier was reduced by a grouping of the multiclass labelling to binary class labelling as also to become an input to the binary classifier as described in section 4.4.2 above.

4.5.2.1 Metrics: Normalisation and Split ratios of binary classification

The normalisation principle is presented in 4.4.3 above. Table 4-12 presents repeated training of the supervised learning classifier using different normalisation and split ratios to infer the results of the prediction errors. It was observed using logistic normalisation of *70:30* (training: testing data) split ratio gave the lowest prediction errors as compared to other split data ratios and normalisation. Figure 4-16 below, presents an optimum classification that was achieved at normalisation using logistic data transformation methods. The data set split ratio of *70* to *30* between the training to testing data at this normalisation achieved an excellent result of FN *=43* and FP=*50* in the performance metrics using the TC SVM classification algorithm.

The prediction errors of FN and FP provides an insight into the performance of a trained model. A wrong prediction of SQLIA to be FN would imply the intruder compromises our protected web application. Also, a high FN of SQLIA threats would mean a failed protection which implies there is a need to keep FN from low to nil to improve the SQLIA mitigation. Conversely, a wrong prediction of SQLIA negatives to be FP is not as detrimental, but an administrative overhead of further reviewing of the referring web requests. FP does offer the opportunity for a second review of any suspect web requests, thereby enhancing the security of the protected web application using the approach presented in this thesis. It is observed as shown in Figure 4-16 below using the logistic normalisation and training to testing data ratio of *70* to *30*%, offers an enhanced classification with low FP and FN.

Table 4-12 Normalisation observations between training and testing data: Observations of prediction errors (FN and FP) in using various normalisation schemes and data set partition between training and testing data at various split ratios in binary classification.

Normalisation types/ratios	False Negative (FN)	False Positive (FP)
Logistic - 70:30	43	50
Z-Score - 70:30	92	254
Z-Score - 80:20	224	276
Logistic - 80:20	170	604

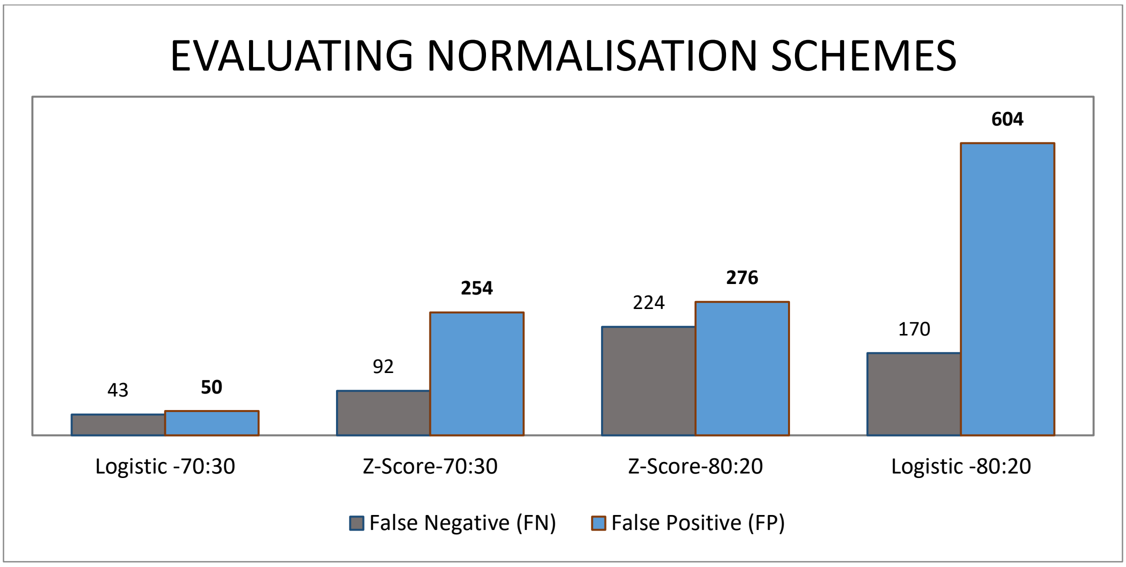


Figure 4-16 A Bar Chart of Evaluation of ZScore and Logistic normalisation: A Bar Chart of evaluation of ZScore and Logistic normalisation at different split ratios in binary classification.

4.5.2.2 Metrics: Statistical measures and ROC curve

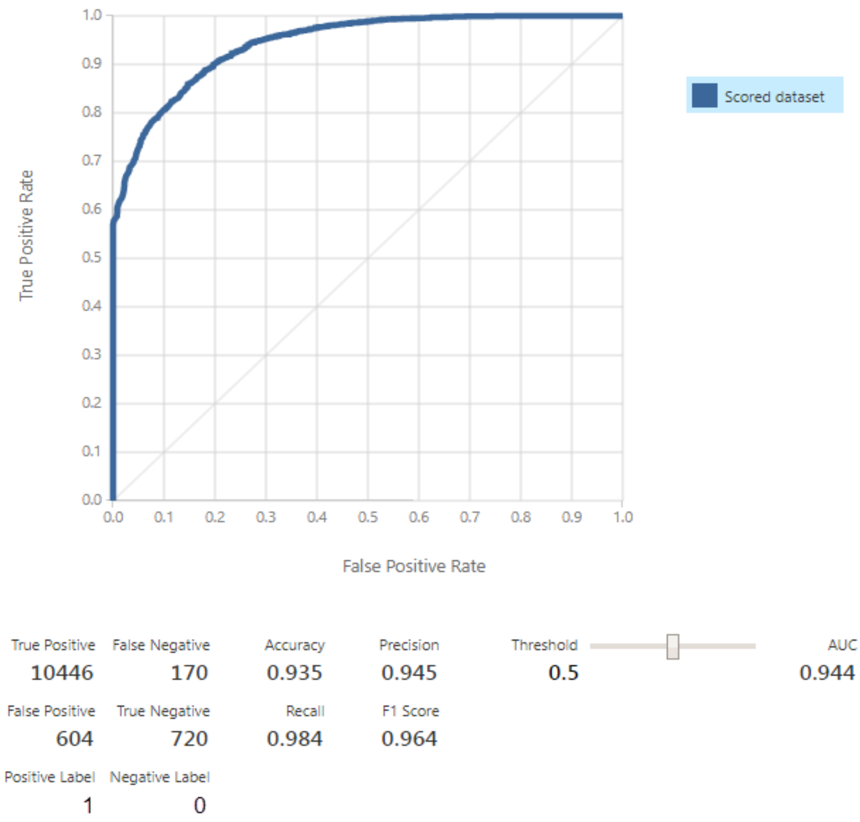
The section below presents the performance metrics of the suitability of a pattern-driven data set in towards predicting SQLIA. The tables presented below has the formula for calculating the statistical measures along with the collated observations similarly presented in Table 4-10 of section 4.5.1.2 above.

Table 4-13 below contains the collated prediction variables of TP, TN and the prediction errors of FP and FN including prediction observations at different thresholds (score bins) using TC SVM algorithms with the calculated statistical measures of using *80* to *20* split data ratios. Figure 4-17 below presents a ROC curve of the evaluation with the AUC plot. The AUC graph is plotted with values from Table 4-13 with FPR on the x-axis and TPR/Recall on the y-axis.

We achieved an overall performance of AUC value of *0.944* using the TC SVM classifier as presented in ROC curve shown in Figure 4-17 below. The performance metrics of normalisation, split ratio, statistical measures and AUC overall performance is compared in the section below to select a better performing classifier to be deployed as a web service in ongoing SQLIA mitigation.

Table 4-13 TC SVM of observations at various thresholds-80:20 data ratio: A table of observations at various thresholds/score bins using TC SVM algorithms, including the statistical measures for 80% training to 20% testing data ratio.

Score Bins	TP	FN	FP	TN	PE	NE	PO	NO	FPR	TPR/Recall	Accuracy	P	FI Score
1	0	10616	0	1324	10616	1324	0	11940	0	0	0.110887772	0	0
0.9	9854	762	19	1302	10616	1321	9873	2064	0.01438304	0.92822155	0.934338358	0.998076	0.961881985
0.8	10090	526	75	1249	10616	1324	10165	1775	0.05664653	0.95045215	0.949664992	0.992622	0.971079351
0.7	10220	396	127	1197	10616	1324	10347	1593	0.09592145	0.96269781	0.956197655	0.987726	0.975051281
0.6	10302	314	202	1122	10616	1324	10504	1436	0.15256798	0.970422	0.95678392	0.980769	0.975568182
0.5	10392	224	276	1048	10616	1324	10668	1272	0.20845921	0.97889977	0.958123953	0.974128	0.976508175
0.4	10451	165	357	967	10616	1324	10808	1132	0.26963746	0.98445742	0.956281407	0.966969	0.975634802
0.3	10508	108	435	889	10616	1324	10943	997	0.32854985	0.98982668	0.954522613	0.960249	0.974813303
0.2	10557	59	566	758	10616	1324	11123	817	0.42749245	0.99444235	0.947654941	0.949114	0.971249827
0.1	10606	10	742	582	10616	1324	11348	592	0.56042296	0.99905803	0.937018425	0.934614	0.965762156
0	10616	0	1324	0	10616	1324	11940	0	1	1	0.889112228	0.889112	0.941301649
Abbrevations & Formula	True Positive (TP)	Faslse Negative (FN)	False Positive (FP)	True Negative (TN)	Positive Event (PE) =TP+FN	Negative Event (NE) =FP+TN	Positive Observ ations(PO) =TP+FP	Negative Obser vations (NO) =FN+TN	False Positive Rate (FPR) =FP / (FP+TN)	True Positive Rate (TPR) =TP / PE	(TP+TN) / Total events (TE) TE = 11940	Precision (P) =TP / PO	FI Score =2*(TPR*P) / (TPR+P)



True Positive	False Negative	Accuracy	Precision	Threshold	AUC
10446	170	0.935	0.945	0.5	0.944
False Positive	True Negative	Recall	F1 Score		
604	720	0.984	0.964		
Positive Label	Negative Label				
1	0				

Figure 4-17 An AUC plot for TC SVM - 80:20 data ratio: A figure of TC SVM performance metrics trained with 80% training to 20% testing data ratio.

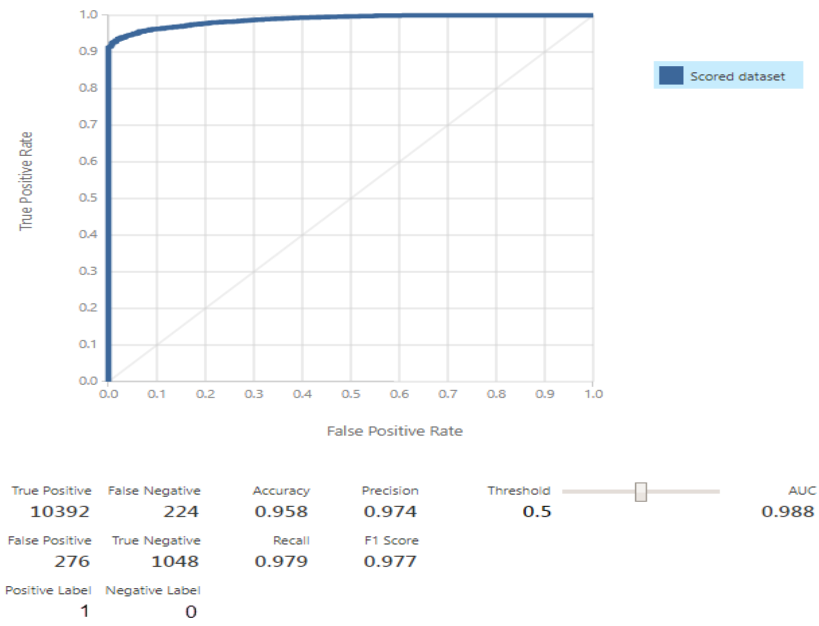
The MAML studio sets by default the optimised cut-off threshold for the prediction of TP and TN including the errors of FP and FN to be *0.5*. This cut-off value of *0.5* is a predetermined threshold employed by classification algorithms in MAML studio. Therefore *f(x) < 0.5* score probability value is predicted as SQLIA negative *(0)* while *f(x)*

≥ *0.5* is predicted as SQLIA positive *(1).* The AUC Figure 4-17 above shows a curve at the default optimised threshold of 0.5 which is calculated using the formula in Table 4-13.

In Table 4-14 below, we present the performance metrics using the same normalisation and split data as used above, but with the TC LR classifier algorithm as against TC SVM. There is a slight improvement on FN prediction errors with values of *170*, but with a higher FP observation of *604* at the default threshold values. Also, there is an overall improvement of AUC with a value of *0.988* shown in Figure 4-18 below.

Table 4-14 TC LR observations at various thresholds-80:20 data ratio: A table of observations at various thresholds using a TC LR algorithm, including the statistical measure calculations.

Score Bins	TP	FN	FP	TN	PE	NE	PO	NO	FPR	TPR/Recall	Accuracy	P	FI Score
1	0	10616	0	1324	10616	1324	0	11940	0	0	0.110887772	0	0
0.9	8996	1620	187	1137	10616	1324	9183	2757	0.14123867	0.84740015	0.848659966	0.979636	0.908732764
0.8	9731	885	302	1022	10616	1324	10033	1907	0.22809668	0.91663527	0.900586265	0.969899	0.942515376
0.7	10113	503	398	926	10616	1324	10511	1429	0.30060423	0.95261869	0.924539363	0.962135	0.95735315
0.6	10310	306	506	818	10616	1324	10816	1124	0.38217523	0.97117558	0.9319933	0.953217	0.962112729
0.5	10446	170	604	720	10616	1324	11050	890	0.45619335	0.98398644	0.935175879	0.945339	0.964275824
0.4	10517	99	691	633	10616	1324	11208	732	0.52190332	0.99067445	0.933835846	0.938348	0.96380132
0.3	10562	54	785	539	10616	1324	11347	593	0.5929003	0.99491334	0.929731993	0.930819	0.96179939
0.2	10600	16	890	434	10616	1324	11490	450	0.67220544	0.99849284	0.924120603	0.922541	0.959015652
0.1	10616	0	1054	270	10616	1324	11670	270	0.79607251	1	0.911725293	0.909683	0.952705735
0	10616	0	1324	0	10616	1324	11940	0	1	1	0.889112228	0.889112	0.941301649
Abbrevations & Formula	**True Positive (TP)**	**Faslse Negative (FN)**	**False Positive (FP)**	**True Negative (TN)**	**Positive Event (PE) =TP+FN**	**Negative Event (NE) =FP+TN**	**Positive Observations(PO) =TP+FP**	**Negative Observations (NO) =FN+TN**	**False Positive Rate (FPR) =FP / (FP+TN)**	**True Positive Rate (TPR) =TP / PE**	**(TP+TN) / Total events (TE) TE = 11940**	**Precision (P) =TP / PO**	**FI Score =2*(TPR*P) / (TPR+P)**



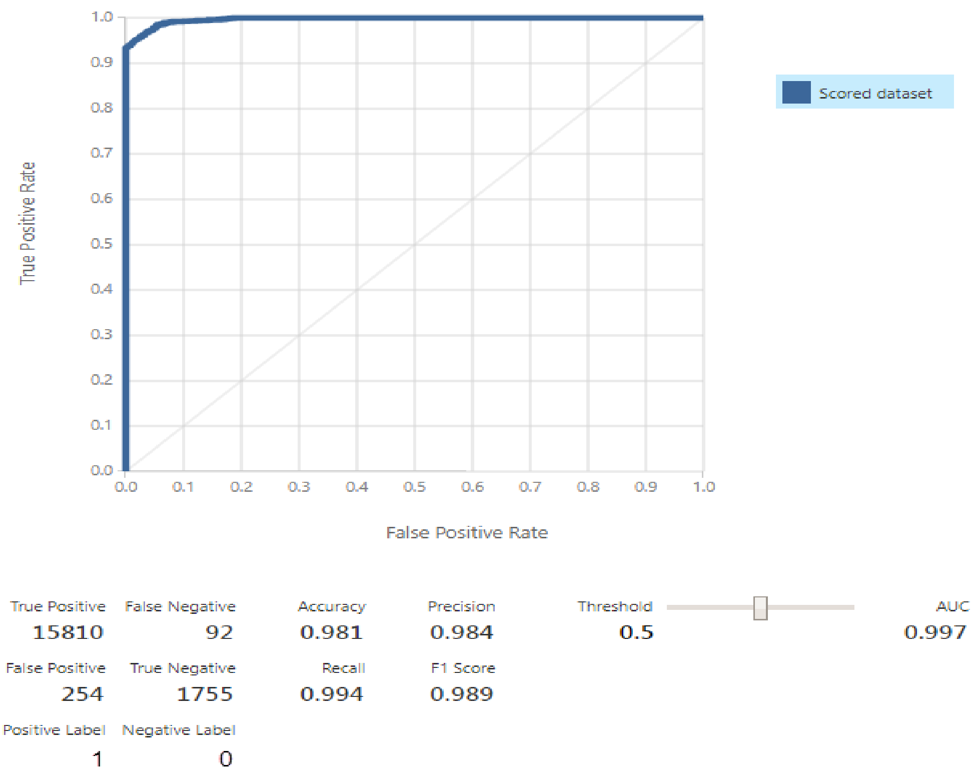
True Positive 10392
False Negative 224
Accuracy 0.958
Precision 0.974
Threshold 0.5
AUC 0.988
False Positive 276
True Negative 1048
Recall 0.979
F1 Score 0.977
Positive Label 1
Negative Label 0

Figure 4-18 An AUC plot for TC LR - 80:20 data ratio: A figure of an AUC plot for TC LR performance metrics for 80% training to 20% testing data ratio at the default threshold of 0.5.

The performance metrics shown in Table 4-15 are observations at various thresholds-using *70:30* split data ratios employing the TC LR classification algorithm. The results present an improvement on prediction errors of FP and FN with values of *92* and *254* respectively at the default threshold. Also, there is an overall improvement on AUC with a value of *0.997* shown in the ROC curve presented in Figure 4-19.

Table 4-15 TC LR observations at various thresholds-70:30 data ratio: A table of observations at various thresholds using a TC LR algorithm, including the statistical measures calculations for 70% training to 30% testing data ratio.

Score Bins	TP	FN	FP	TN	PE	NE	PO	NO	FPR	TPR/Recall	Accuracy	P	FI Score
1	0	15902	0	2009	15902	2009	0	17911	0	0	0.168257956	0	0
0.9	15253	649	53	1956	15902	2009	15306	2605	0.02638128	0.95918752	1.441289782	0.996537	0.977505768
0.8	15643	259	111	1898	15902	2009	15754	2157	0.05525137	0.98371274	1.469095477	0.992954	0.988311852
0.7	15723	179	138	1871	15902	2009	15861	2050	0.06869089	0.98874355	1.473534338	0.991299	0.990019834
0.6	15775	127	196	1813	15902	2009	15971	1940	0.09756098	0.99201358	1.473031826	0.987728	0.989866031
0.5	15810	92	254	1755	15902	2009	16064	1847	0.12643106	0.99421456	1.471105528	0.984188	0.989175999
0.4	15832	70	302	1707	15902	2009	16134	1777	0.15032354	0.99559804	1.468927973	0.981282	0.988388063
0.3	15853	49	337	1672	15902	2009	16190	1721	0.16774515	0.99691863	1.467755444	0.979185	0.98797208
0.2	15882	20	362	1647	15902	2009	16244	1667	0.18018915	0.9987423	1.468090452	0.977715	0.988116717
0.1	15902	0	423	1586	15902	2009	16325	1586	0.21055251	1	1.464656616	0.974089	0.98687436
0	15902	0	2009	0	15902	2009	17911	0	1	1	1.331825796	0.887834	0.940584982
Abbreviations & Formula	True Positive (TP)	Faslse Negative (FN)	False Positive (FP)	True Negative (TN)	Positive Event (PE) =TP+FN	Negative Event (NE) =FP+TN	Positive Observations(PO) =TP+FP	Negative Observations (NO) =FN+TN	False Positive Rate (FPR) =FP / (FP+TN)	True Positive Rate (TPR) =TP / PE	(TP+TN) / Total events (TE) TE = 17911	Precision (P) =TP / PO	FI Score =2*(TPR*P) / (TPR+P)



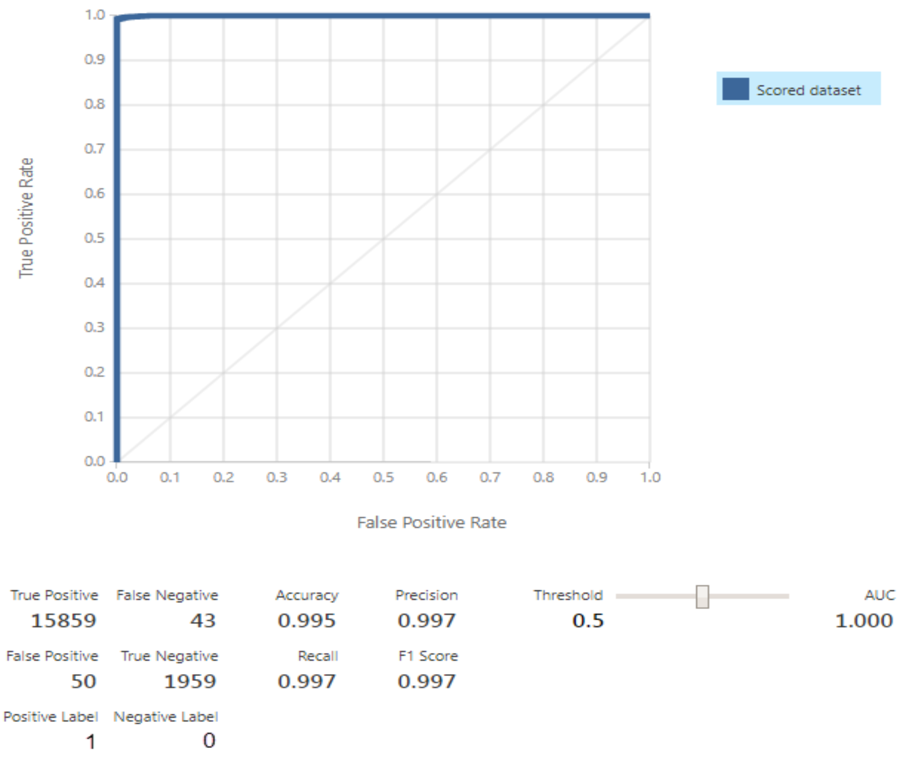
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
15810	92	0.981	0.984	0.5	0.997
False Positive	True Negative	Recall	F1 Score		
254	1755	0.994	0.989		
Positive Label	Negative Label				
1	0				

Figure 4-19 An AUC plot for TC LR - 70:30 data ratio: A figure of an AUC plot for TC LR for 70% training to 30% testing split data ratio with the capture of the performance metrics at the default threshold of 0.5.

The performance metrics shown in Table 4-16 below uses TC SVM classifier with the training to testing split data ratio of *70* to *30*. This evaluation presents an improvement on prediction errors of FN and FP with observations of low values of *43* and *50* respectively at the default threshold as against preceding evaluations. The result presents an excellent performance with an AUC value of *1.0* shown in the ROC curve presented in Figure 4-20.

Table 4-16 TC SVM observations at various thresholds-70:30 data ratio: A table of observations at various thresholds using a TC SVM algorithm, including the statistical measures calculations for 70% training to 30% testing data ratio.

Score Bins	TP	FN	FP	TN	PE	NE	PO	NO	FPR	TPR/Recall	Accuracy	P	FI Score
1	0	15902	0	2009	15902	2009	0	17911	0	0	0.112165708	0	0
0.9	15750	152	0	2009	15902	2009	15750	2161	0	0.990441454	0.991513595	1	0.995197776
0.8	15797	105	10	1999	15902	2009	15807	2104	0.004977601	0.993397057	0.993579365	0.9993674	0.996373269
0.7	15834	68	27	1982	15902	2009	15861	2050	0.013439522	0.995723808	0.994695997	0.9982977	0.997009099
0.6	15851	51	36	1973	15902	2009	15887	2024	0.017919363	0.996792856	0.99514265	0.997734	0.997263204
0.5	15859	43	50	1959	15902	2009	15909	2002	0.024888004	0.997295938	0.99480766	0.9968571	0.997076483
0.4	15869	33	62	1947	15902	2009	15931	1980	0.030861125	0.997924789	0.994695997	0.9961082	0.997015676
0.3	15885	17	91	1918	15902	2009	15976	1935	0.045296167	0.998930952	0.993970186	0.994304	0.996612084
0.2	15896	6	120	1889	15902	2009	16016	1895	0.05973121	0.999622689	0.992965217	0.9925075	0.996052384
0.1	15902	0	155	1854	15902	2009	16057	1854	0.077152812	1	0.9913461	0.9903469	0.995150036
0	15902	0	2009	0	15902	2009	17911	0	1	1	0.887834292	0.8878343	0.940584982
Abbrevations & Formula	**True Positive (TP)**	**Faslse Negative (FN)**	**False Positive (FP)**	**True Negative (TN)**	**Positive Event (PE) =TP+FN**	**Negative Event (NE) =FP+TN**	**Positive Observations(PO) =TP+FP**	**Negative Obser vations (NO) =FN+TN**	**False Positive Rate (FPR) =FP / (FP+TN)**	**True Positive Rate (TPR) =TP / PE**	**(TP+TN) / Total events (TE) TE = 17911**	**Precision (P) =TP / PO**	**FI Score =2*(TPR*P) / (TPR+P)**



True Positive 15859
False Negative 43
Accuracy 0.995
Precision 0.997
Threshold 0.5
AUC 1.000
False Positive 50
True Negative 1959
Recall 0.997
F1 Score 0.997
Positive Label 1
Negative Label 0

Figure 4-20 An AUC plot for TC SVM - 70:30 data ratio: A figure of an AUC plot for TC SVM performance metrics for 70% training to 30% testing data ratio with the capture of the performance metrics at the default threshold of 0.5.

We compared the overall performance metrics of the trained, supervised learning models from the plot of FPR on the x-axis against TPR on the y-axis using the data presented in Table 4-13 -Table 4-16 above. We observed that TC SVM using the logistic normalisation and training to testing split data ratio of *70* to *30* resulted overall in an excellent trained model as graded in Table 4-11 above (refer to section 4.5.1.4 for the AUC performance grading). We achieved an AUC performance of value of *1.0* with low prediction errors of *50* observations for FP and *43* for FN.

The comparisons of various performance metrics of normalisation, split data ratios, and ROC curve's AUC is shown in Figure 4-21. These comparisons coupled with the cross-validation assisted in the selection of the best performing trained supervised learning model to be deployed as a web service in ongoing real-time SQLIA mitigation.

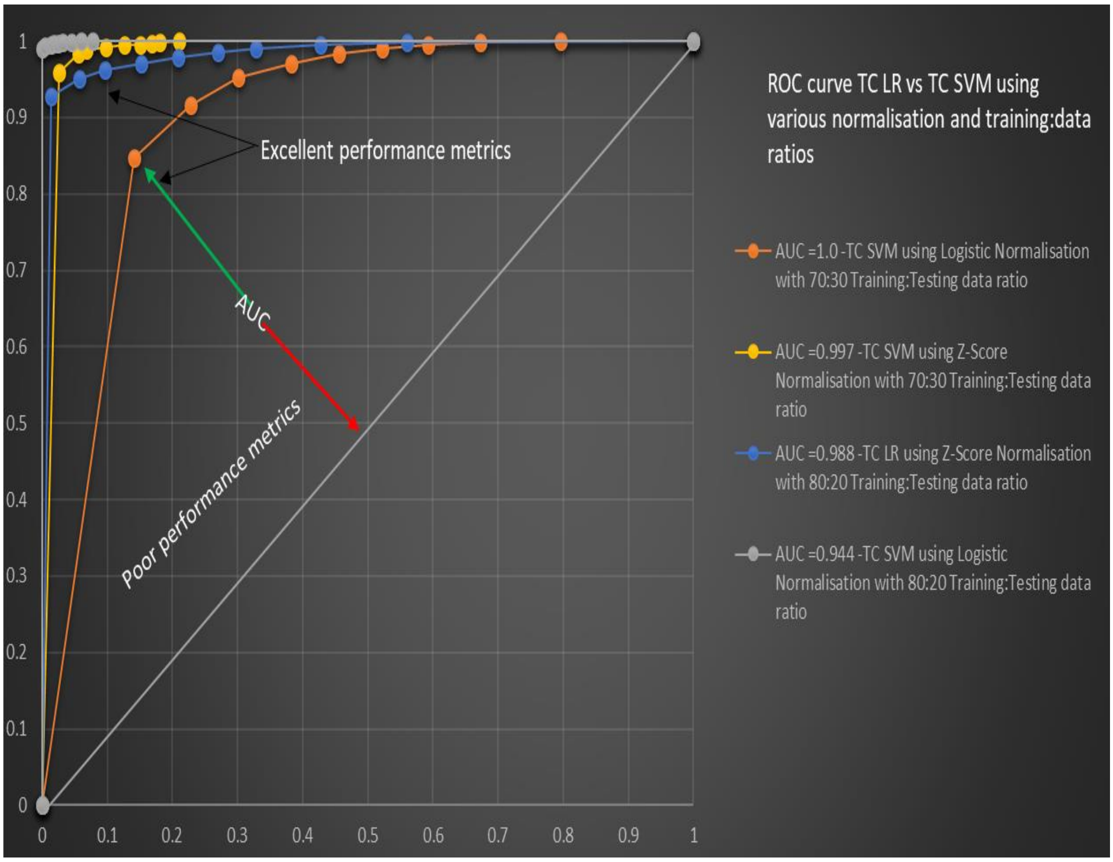


Figure 4-21 AUC comparisons across the algorithms and ratios: A figure of AUC comparisons across the classification algorithms, split data ratios and normalisation schemes to observed a better performance metrics with TC SVM employing logistic normalisation with 70% training to 30% testing data ratio.

4.5.3 Multiclass classification algorithms

Figure 4-22 presents this section organisation chart with the associated subsections.

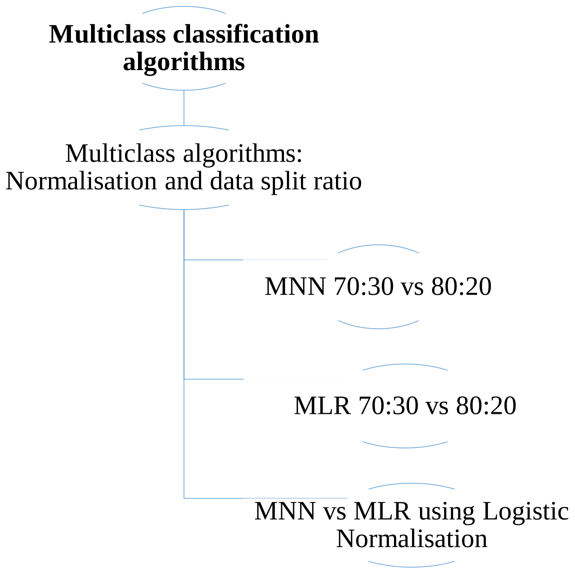

Multiclass classification algorithms
Multiclass algorithms: Normalisation and data split ratio
MNN 70:30 vs 80:20
MLR 70:30 vs 80:20
MNN vs MLR using Logistic Normalisation

Figure 4-22 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

In this section, we evaluated and validated the pattern-driven labelled multiclass data set employing multiclass algorithms. The results of the multiclass classifications employing MNN and MLR are presented in the sections below. We observed the MLR algorithm gave a better performance over MNN in prediction evaluated in CM. The data set contains labelled classes of the various SQLIA types detailed in Table 2-1 above. The numeric labels presented in Table 4-17 below is a snippet of data set obtained by numerical encoding ontology presented in section 4.3 above which provides the technique to derive numeric artefact for SQLIA types.

Table 4-17 Labels/classes to predict: A table illustrating the labelling of the SQLIA types including expected valid request.

SQLIA Types	Class/Labels	Snippet of signature
Tautology	1	`--,1=1,1<1, ‘a’=’a’ etc.
Piggyback	2	;
Union	3	UNION SELECT
Invalid/Logical Incorrect Query	4	SELECT, IF, ELSE, dbo,1/0
Alternate encoding obfuscation	5	), %, Hex values
Time-based	6	Wait for delay '00:00:10'
Store Procedure	7	; EXEC, XP_CMDSHELL, ATTRIB
Second order	8	‘x’=x’--"
Valid web request		
Expected collated input/non-SQLIA	9	Bob, bbo, obb etc

4.5.3.1 Multiclass algorithms: Normalisation and split data ratio

We compared various multiclass classification algorithms under various split ratios. The Logistic normalisation scheme was observed with better results in predicting low FN and FP in the binary classification in section 4.5.2. We used the Logistic normalisation to compare various training to testing data set split ratios.

Also, presented below is a comparison between MNN and MLR algorithms under these various split data schemes to select a better performing trained model to be deployed as a web service in ongoing SQLIA detection and prevention.

The statistical measure calculations are collated from score probabilities to provide observations of correctly predicted features of TP and TN including the prediction errors of FP and FN in a trained model. These statistical measure variables, abbreviations and formula are presented in Table 4-10 above.

The equation to calculate the macro and micro average metrics for comparison of the predicted results of various split data ratios are presented in Equation 4-9. The equations illustrate how the performance metrics results used on the bar chart in Figure 4-23 below and the rest of the section for the multiclass performance metrics comparison are calculated. The calculated macro and micro average metrics in Table 4-19 is extrapolated from Table 4-18 below.

Equation 4-9

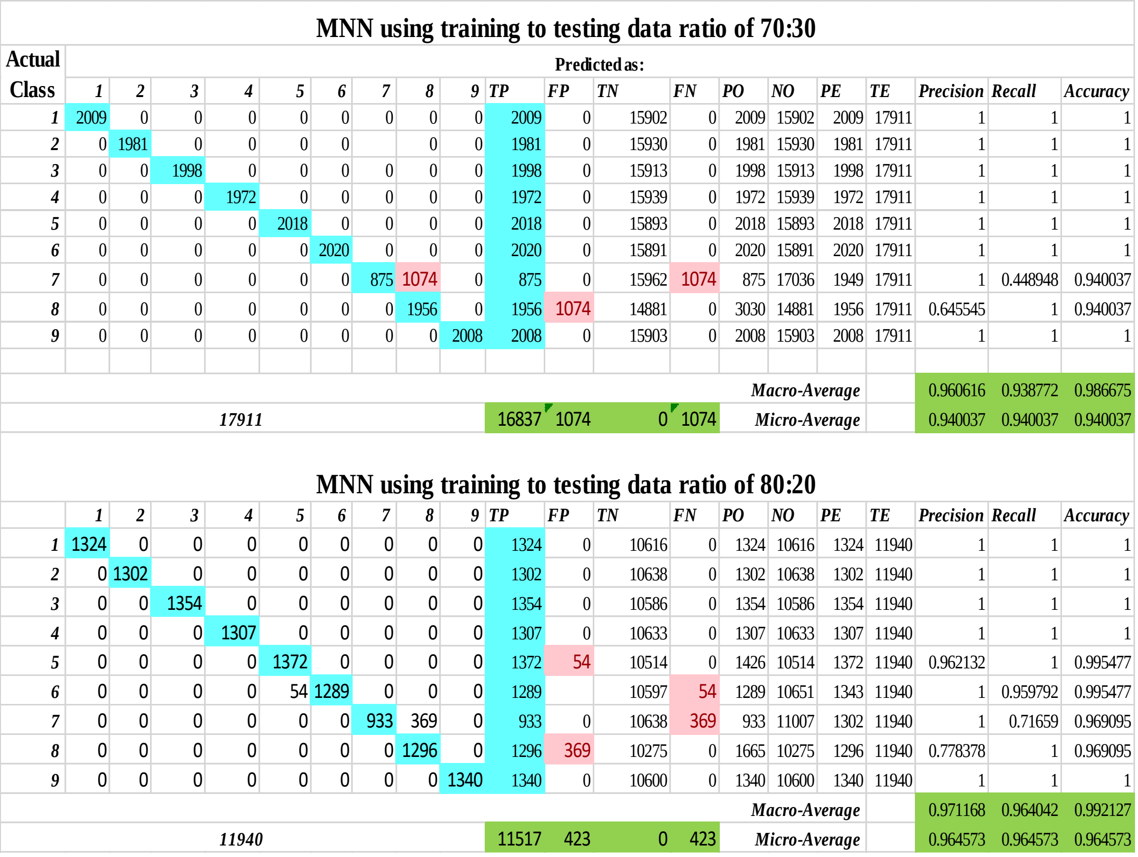
Calculating the multiclass metrics presented across the table

Binary		**Multiclass**	
Accuracy (A)	A = (TP+TN) / (TP+FP+TN+FN)	***Overall accuracy (A_{micro})***	$A_{micro} = A(\sum_{\lambda=1}^{n} \frac{\mathrm{TP + TN}}{\mathrm{TP + FP + TN +}}$
		Average accuracy (A_{macro})	$A_{macro} = \frac{1}{n}\sum_{\lambda=1}^{n} A$
Precision (P)	P = TP / (TP+FP)	***Micro-averaged precision(_{macro})***	$P(\sum_{\lambda=1}^{n} \frac{\mathrm{TP}}{\mathrm{TP + FP}})$
		Macro-averaged precision	$\frac{1}{n}\sum_{\lambda=1}^{n} P$
Recall (R)	R = TP / TP+FN	***Micro-averaged recall***	$P(\sum_{\lambda=1}^{n} \frac{\mathrm{TP}}{\mathrm{TP + FN}})$
		Macro-averaged recall	$\frac{1}{n}\sum_{\lambda=1}^{n} R$
		Where λ is the classes/labels L= { $\lambda = 1 \ldots . n$) or L= 1-9	

4.5.3.1.1 MNN 70:30 vs 80:20 training to testing split data ratio

Table 4-18 below, presents the observations of prediction across the nine classes or labels for MNN algorithms using split data ratios of *70* to *30* and *80* to *20* of training to testing data.

Table 4-18 MNN at 70 to 30 and 80 to 20 split data ratios: A table of performance metrics for MNN at 70 to 30 and 80 to 20 of training to testing split data ratio.



MNN using training to testing data ratio of 70:30

Actual												Predicted as:								
Class	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*TP*	*FP*	*TN*	*FN*	*PO*	*NO*	*PE*	*TE*	*Precision*	*Recall*	*Accuracy*
1	2009	0	0	0	0	0	0	0	0	2009	0	15902	0	2009	15902	2009	17911	1	1	1
2	0	1981	0	0	0	0		0	0	1981	0	15930	0	1981	15930	1981	17911	1	1	1
3	0	0	1998	0	0	0	0	0	0	1998	0	15913	0	1998	15913	1998	17911	1	1	1
4	0	0	0	1972	0	0	0	0	0	1972	0	15939	0	1972	15939	1972	17911	1	1	1
5	0	0	0	0	2018	0	0	0	0	2018	0	15893	0	2018	15893	2018	17911	1	1	1
6	0	0	0	0	0	2020	0	0	0	2020	0	15891	0	2020	15891	2020	17911	1	1	1
7	0	0	0	0	0	0	875	1074	0	875	0	15962	1074	875	17036	1949	17911	1	0.448948	0.940037
8	0	0	0	0	0	0	0	1956	0	1956	1074	14881	0	3030	14881	1956	17911	0.645545	1	0.940037
9	0	0	0	0	0	0	0	0	2008	2008	0	15903	0	2008	15903	2008	17911	1	1	1
																Macro-Average		0.960616	0.938772	0.986675
				17911						16837	1074	0	1074			*Micro-Average*		0.940037	0.940037	0.940037

MNN using training to testing data ratio of 80:20

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*TP*	*FP*	*TN*	*FN*	*PO*	*NO*	*PE*	*TE*	*Precision*	*Recall*	*Accuracy*
1	1324	0	0	0	0	0	0	0	0	1324	0	10616	0	1324	10616	1324	11940	1	1	1
2	0	1302	0	0	0	0	0	0	0	1302	0	10638	0	1302	10638	1302	11940	1	1	1
3	0	0	1354	0	0	0	0	0	0	1354	0	10586	0	1354	10586	1354	11940	1	1	1
4	0	0	0	1307	0	0	0	0	0	1307	0	10633	0	1307	10633	1307	11940	1	1	1
5	0	0	0	0	1372	0	0	0	0	1372	54	10514	0	1426	10514	1372	11940	0.962132	1	0.995477
6	0	0	0	0	54	1289	0	0	0	1289		10597	54	1289	10651	1343	11940	1	0.959792	0.995477
7	0	0	0	0	0	0	933	369	0	933	0	10638	369	933	11007	1302	11940	1	0.71659	0.969095
8	0	0	0	0	0	0	0	1296	0	1296	369	10275	0	1665	10275	1296	11940	0.778378	1	0.969095
9	0	0	0	0	0	0	0	0	1340	1340	0	10600	0	1340	10600	1340	11940	1	1	1
																Macro-Average		0.971168	0.964042	0.992127
				11940						11517	423	0	423			*Micro-Average*		0.964573	0.964573	0.964573

Table 4-19 Extrapolated MNN performance metrics: A table of extrapolated MNN performance metrics for training: testing split data ratios.

Metrics	**Training: Testing data ratio**		**% Improvement** $((\boldsymbol{MNN\ 80{:}20}/\boldsymbol{MNN\ 70{:}30}) - \mathbf{1}) * \mathbf{100}$
	MNN 70:30	***MNN 80:20***	
Overall accuracy	0.940037	0.964573	2.61
Average accuracy	0.986675	0.992127	0.55
Micro-averaged precision	0.940037	0.964573	2.61
Macro-averaged precision	0.960616	0.971168	1.10
Micro-averaged recall	0.940037	0.964573	2.61
Macro-averaged recall	0.938772	0.964042	2.69
Total overall improvement of MNN 80:20 over MNN 70:30			**12.17%**

The predicted classes against the actual classes were compared as presented in the tables and figures to infer the following observations:

- All TP and TN were all correctly predicted as labelled classes $0 < x < 9$ are SQLIA types positive while $x \geq 9$ is SQLIA type negative as shown in Table 4-18 above.
- There were FP and FN, which are misclassification of an SQLIA type positive class with cases of SQLIA type labelled *7* predicted as type *8*. This is expected as there is a close similarity between some SQLIA type signatures and generalisation of such similarity during encoding.
- In comparing the overall performance metrics between split data ratios of *70:30* versus *80:20*, it was observed the split ratio *80:20* gave better overall performance metrics in all statistical measures, but with low FP and FN in SQLIA positive misclassification as illustrated in Figure 4-23 below.
- Figure 4-24 below shows an improvement in misclassification with *28.3%* of SQLIA positive labelled type *7* predicted as type *8* by using split data ratio of *80:20* (but with *4%* of labelled class *6* predicted as class *5);* as against *55.9%* misclassification when using split data ratio of *70:30* training to testing data.
- Table 4-19 presents a performance metrics improvement across the statistical measures using the MNN algorithm with a split data ratio of *80:20* and a total overall improvement of *12.17%* over the MNN algorithm using *70:30*.

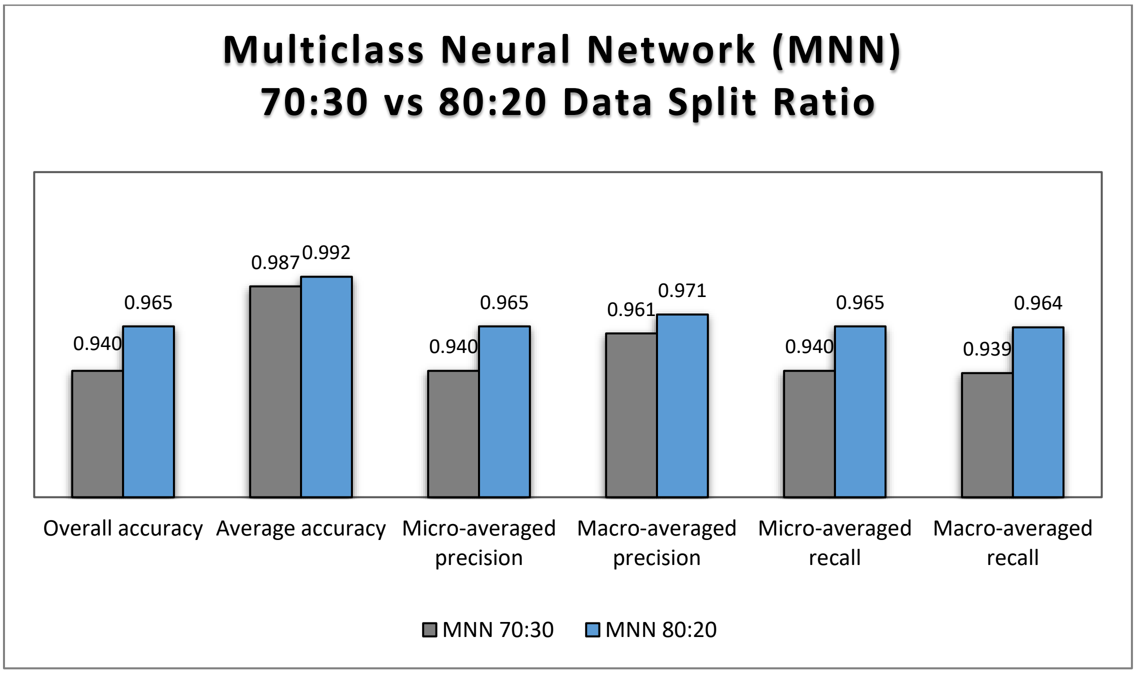
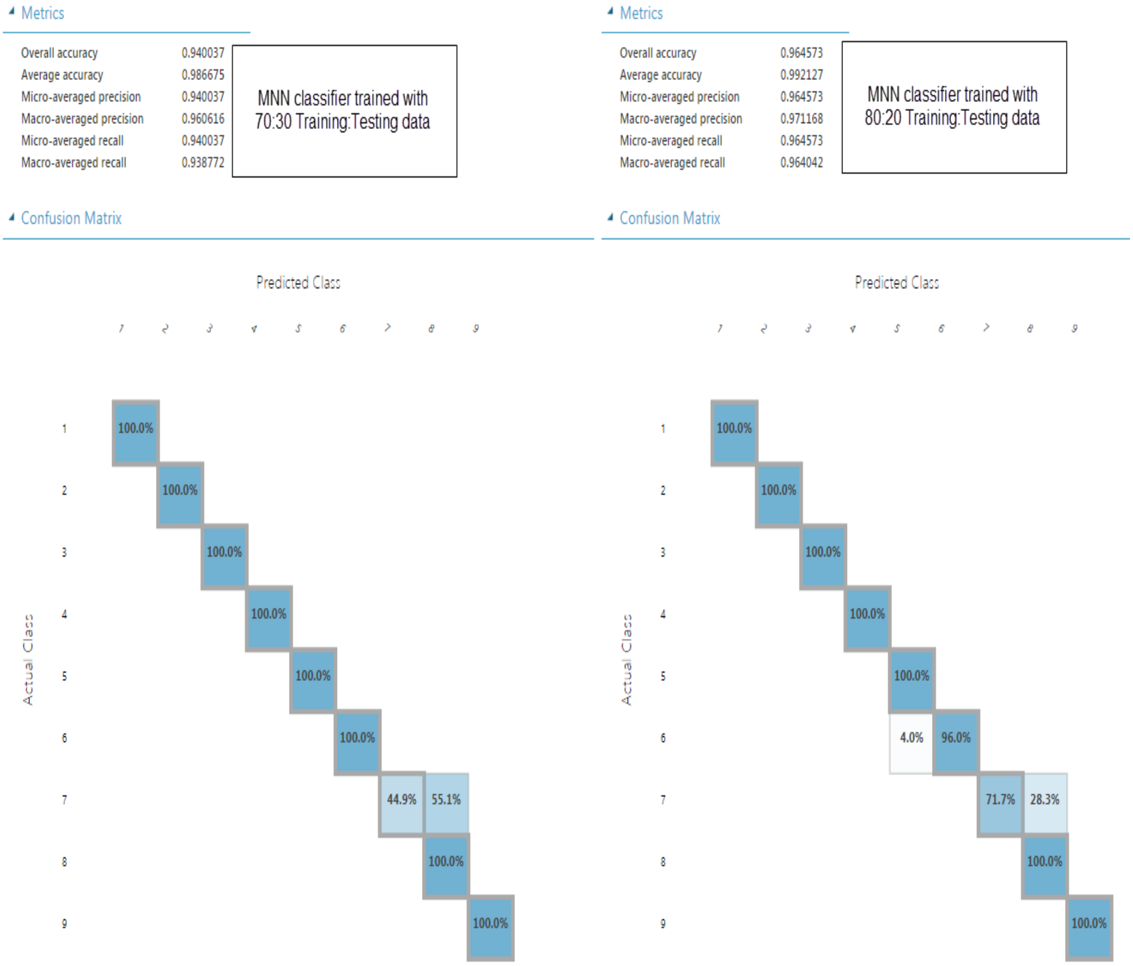


Figure 4-23 Bar Chart: MNN using Logistic Normalisation: A Bar Chart of performance metrics of MNN using Logistic Normalisation 70:30 vs 80:20 Training: Testing Split data ratio.



Metrics

Overall accuracy	0.940037
Average accuracy	0.986675
Micro-averaged precision	0.940037
Macro-averaged precision	0.960616
Micro-averaged recall	0.940037
Macro-averaged recall	0.938772

MNN classifier trained with 70:30 Training:Testing data

Confusion Matrix

Predicted Class

Actual Class	1	2	3	4	5	6	7	8	9
1	100.0%								
2		100.0%							
3			100.0%						
4				100.0%					
5					100.0%				
6						100.0%			
7							44.9%	55.1%	
8								100.0%	
9									100.0%

Metrics

Overall accuracy	0.964573
Average accuracy	0.992127
Micro-averaged precision	0.964573
Macro-averaged precision	0.971168
Micro-averaged recall	0.964573
Macro-averaged recall	0.964042

MNN classifier trained with 80:20 Training:Testing data

Confusion Matrix

Predicted Class

Actual Class	1	2	3	4	5	6	7	8	9
1	100.0%								
2		100.0%							
3			100.0%						
4				100.0%					
5					100.0%				
6					4.0%	96.0%			
7							71.7%	28.3%	
8								100.0%	
9									100.0%

Figure 4-24 MNN using split data ratios of 70:30 vs 80:20: A figure of MNN Confusion Matrix for Training: Testing split data ratios of 70:30 vs 80:20.

4.5.3.1.2 MLR 70:30 vs 80:20 training to testing split data ratio.

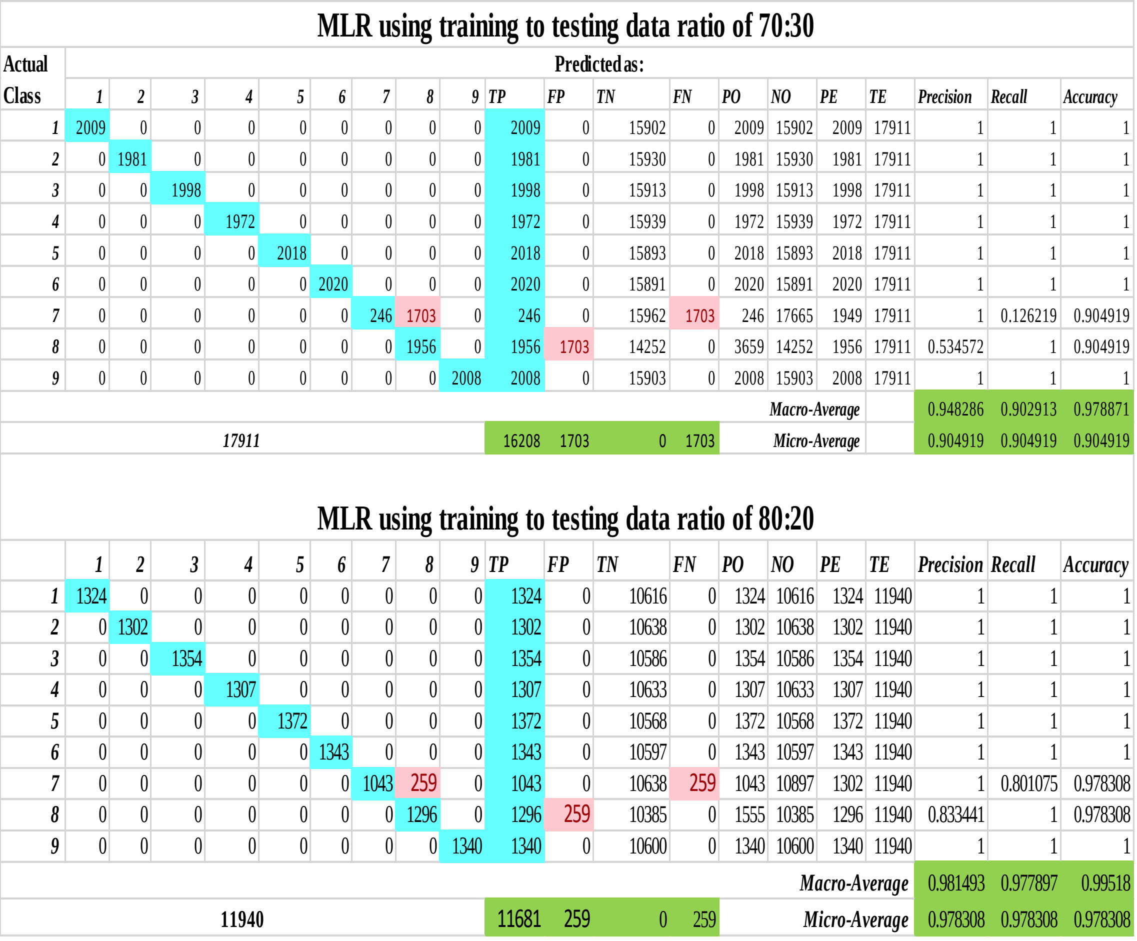
Table 4-20 below present the observations of prediction across the nine classes or labels for MLR algorithms using *70* to *30* versus *80* to *20* of training to testing split data ratios. It is inferred from the CM below that using a split data ratio of *80:20* gave overall better performance metrics by comparing the predicted classes against the actual classes with the following observations below:

- All TP and TN were all correctly predicted as labels $0 < x < 9$ are SQLIA positive while $x \geq 9$ is SQLIA negative as presented in Table 4-20 below.
- There were FP and FN of misclassification of an SQLIA positive class or label from *7* to *8* which is expected due to the close similarity between SQLIA types signatures.
- In comparing the overall metrics between split data ratios of *70:30* versus *80:20*, it was observed the split ratio *80:20* gave overall better performance metrics in all

statistical measures, but with low FP and FN in SQLIA positive misclassification as illustrated in Figure 4-26 below.

- While *87.4 %* of actual labelled class *7* were predicted as label class *8* (misclassified) using split data ratio of *70:30* training to testing data; but in using a split data ratio of *80:20*, only *19.9%* were misclassified as presented in Figure 4-25 below. We observed in the generating the data set vectors there were labelled features that share close similarity between class *7* and *8*. For example, the single quote (‘) to have resulted in a few misclassifications.
- The overall performance metrics indicate an excellent trained model as presented in Figure 4-26 below capable of accurately predicting SQLIA in a web request.
- Table 4-21 presents a performance metrics improvement across the statistical measures using the MLR algorithm with a split data ratio of *80:20* and a total overall improvement of *37.22*% over the MLR algorithm using *70:30*.

Table 4-20 Performance Metrics for MLR at 70:30 and 80:20: A table of performance metrics for MLR at 70:30 and 80:20 of training to testing split data ratio with 80:20 split data having better performance.

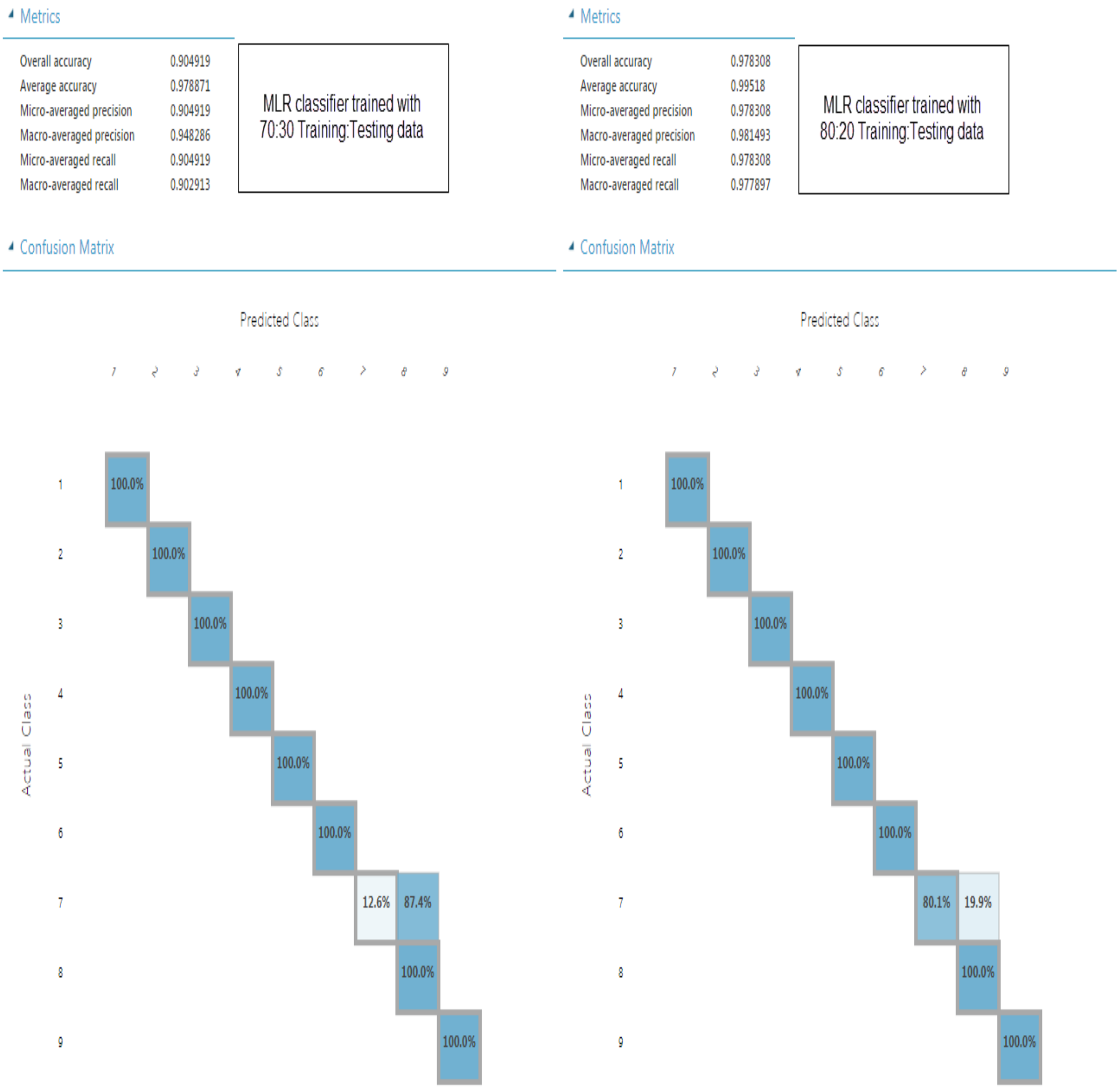


MLR using training to testing data ratio of 70:30

Actual Class	Predicted as:																			
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*TP*	*FP*	*TN*	*FN*	*PO*	*NO*	*PE*	*TE*	*Precision*	*Recall*	*Accuracy*
1	2009	0	0	0	0	0	0	0	0	2009	0	15902	0	2009	15902	2009	17911	1	1	1
2	0	1981	0	0	0	0	0	0	0	1981	0	15930	0	1981	15930	1981	17911	1	1	1
3	0	0	1998	0	0	0	0	0	0	1998	0	15913	0	1998	15913	1998	17911	1	1	1
4	0	0	0	1972	0	0	0	0	0	1972	0	15939	0	1972	15939	1972	17911	1	1	1
5	0	0	0	0	2018	0	0	0	0	2018	0	15893	0	2018	15893	2018	17911	1	1	1
6	0	0	0	0	0	2020	0	0	0	2020	0	15891	0	2020	15891	2020	17911	1	1	1
7	0	0	0	0	0	0	246	1703	0	246	0	15962	1703	246	17665	1949	17911	1	0.126219	0.904919
8	0	0	0	0	0	0	0	1956	0	1956	1703	14252	0	3659	14252	1956	17911	0.534572	1	0.904919
9	0	0	0	0	0	0	0	0	2008	2008	0	15903	0	2008	15903	2008	17911	1	1	1
																Macro-Average		0.948286	0.902913	0.978871
				17911						16208	1703	0	1703			*Micro-Average*		0.904919	0.904919	0.904919

MLR using training to testing data ratio of 80:20

	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*TP*	*FP*	*TN*	*FN*	*PO*	*NO*	*PE*	*TE*	*Precision*	*Recall*	*Accuracy*
1	1324	0	0	0	0	0	0	0	0	1324	0	10616	0	1324	10616	1324	11940	1	1	1
2	0	1302	0	0	0	0	0	0	0	1302	0	10638	0	1302	10638	1302	11940	1	1	1
3	0	0	1354	0	0	0	0	0	0	1354	0	10586	0	1354	10586	1354	11940	1	1	1
4	0	0	0	1307	0	0	0	0	0	1307	0	10633	0	1307	10633	1307	11940	1	1	1
5	0	0	0	0	1372	0	0	0	0	1372	0	10568	0	1372	10568	1372	11940	1	1	1
6	0	0	0	0	0	1343	0	0	0	1343	0	10597	0	1343	10597	1343	11940	1	1	1
7	0	0	0	0	0	0	1043	259	0	1043	0	10638	259	1043	10897	1302	11940	1	0.801075	0.978308
8	0	0	0	0	0	0	0	1296	0	1296	259	10385	0	1555	10385	1296	11940	0.833441	1	0.978308
9	0	0	0	0	0	0	0	0	1340	1340	0	10600	0	1340	10600	1340	11940	1	1	1
																	Macro-Average	0.981493	0.977897	0.99518
				11940						11681	259	0	259				*Micro-Average*	0.978308	0.978308	0.978308



Metrics

Overall accuracy	0.904919
Average accuracy	0.978871
Micro-averaged precision	0.904919
Macro-averaged precision	0.948286
Micro-averaged recall	0.904919
Macro-averaged recall	0.902913

MLR classifier trained with 70:30 Training:Testing data

Metrics

Overall accuracy	0.978308
Average accuracy	0.99518
Micro-averaged precision	0.978308
Macro-averaged precision	0.981493
Micro-averaged recall	0.978308
Macro-averaged recall	0.977897

MLR classifier trained with 80:20 Training:Testing data

Figure 4-25 MLR Confusion Matrix at various split data ratios: A figure of MLR Confusion Matrix for Training: Testing split data ratios of 70:30 vs 80:20.

Table 4-21 MLR at various split data ratios: A table of extrapolated MLR performance metrics for training: testing split data ratios with better MLR 80:20 performance metrics

Metrics	Training: Testing data ratio		% Improvement $((\boldsymbol{MLR}\ 80{:}20/\boldsymbol{MLR}\ 70{:}30) - 1) * 100$
	MLR 70:30	***MLR 80:20***	
Overall accuracy	0.904919	0.978308	8.11
Average accuracy	0.978871	0.99518	1.67
Micro-averaged precision	0.904919	0.97308	7.53
Macro-averaged precision	0.948286	0.981493	3.50
Micro-averaged recall	0.904919	0.978308	8.11
Macro-averaged recall	0.902913	0.977897	8.30
Total overall improvement of MLR 80:20 over MLR 70:30			**37.22%**

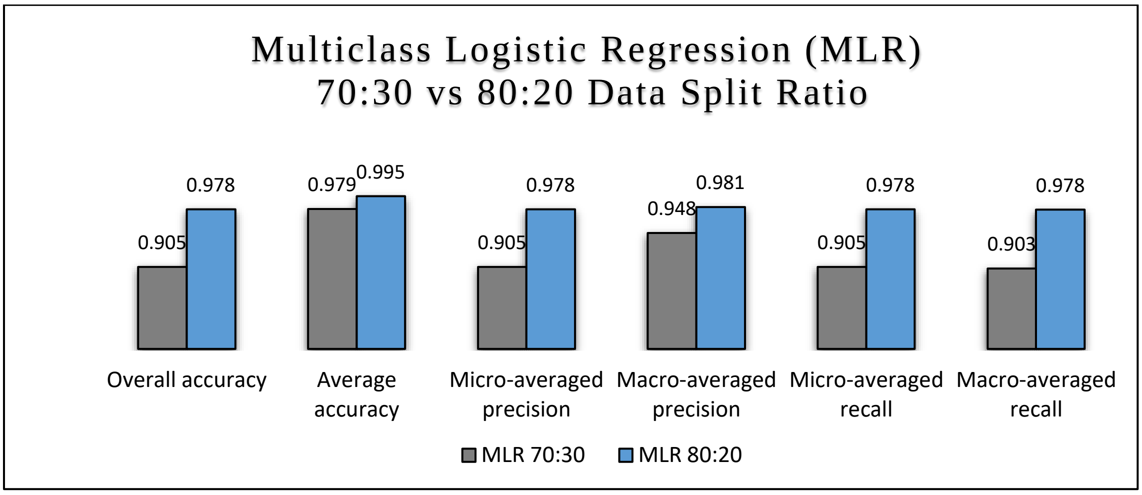


Figure 4-26 Bar Chart: MLR using Logistic normalisation: A Bar Chart of performance metrics of MLR using Logistic normalisation 70:30 vs 80:20 training: testing split data ratio.

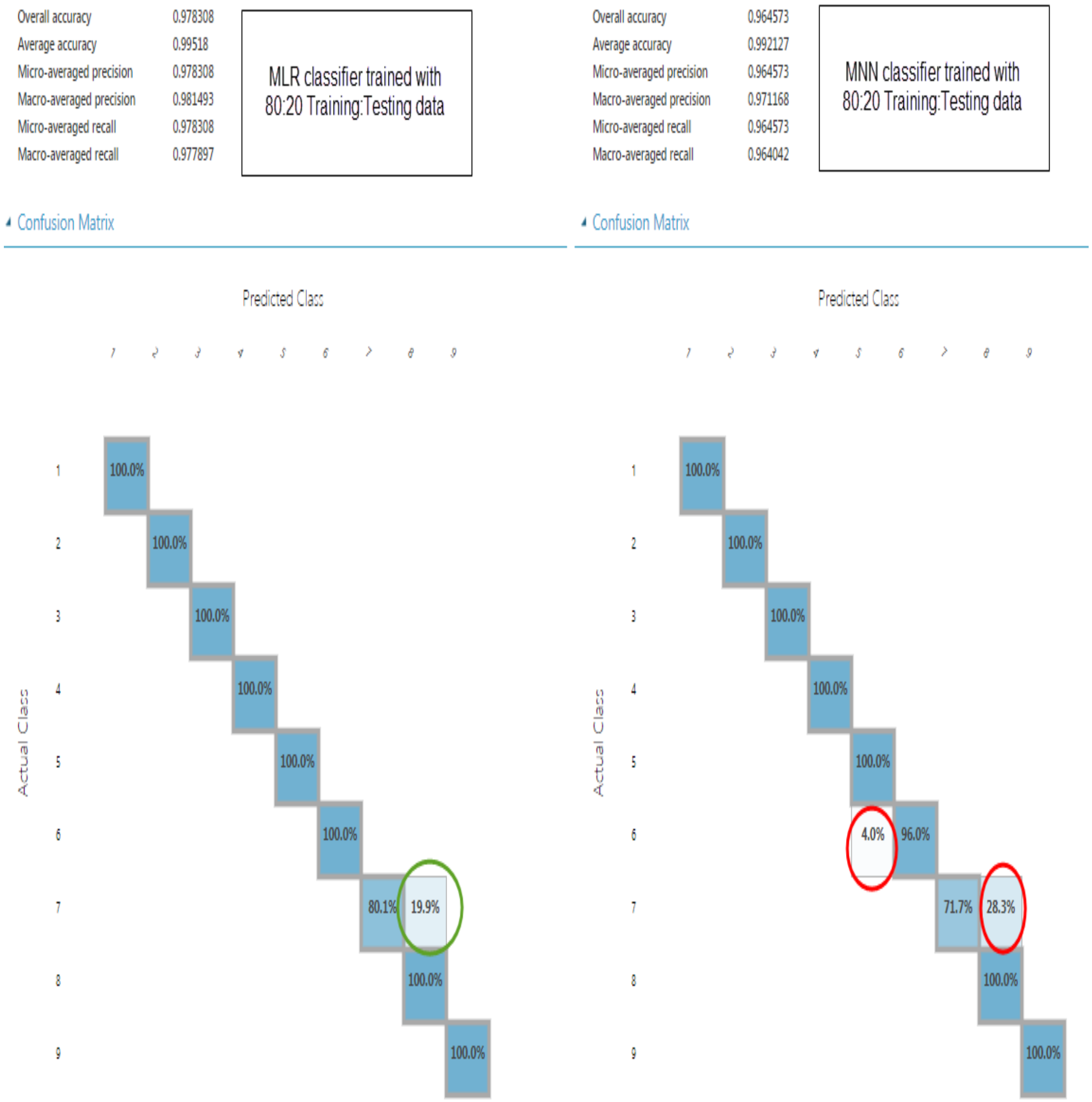
4.5.3.1.3 MNN vs MLR using Logistic Normalisation at 80:20 split data ratio

In the preceding sections (4.5.3.1.1 & 4.5.3.1.2 above), it is observed that using a split data ratio of *80:20* to train MNN and MLR classifiers gave better performance metrics than using a split data ratio of 70:30 training to testing data.

In this section, we compare the performance metrics of both MNN and MLR classifiers with the input of *80:20* training to testing split data ratio. Figure 4-28 below presents the results for both MNN and MLR to have observed MLR classifier have better-performing metrics as extrapolated from Table 4-22 below with a total overall improvement of *7.08%* over the MNN algorithm.

Also, it was observed in the CM presented in Figure 4-27 below that a split data ratio of *80:20* training to testing data in MLR has overall low misclassification of the labelled class *7* with *19.9%*. Conversely, the same split data ratio in MNN has a total of 32.3% misclassification of two labelled classes of which *4%* and *28.3%* were labelled class *6* and *7* respectively. This misclassification is expected as there is a close similarity between some SQLIA type signatures and generalisation of such similarity during encoding.

The performance metrics across the statistical measures show using MLR algorithms with a split data ratio of *80:20* training to testing data delivered a better performance metrics over MNN algorithm using the same split data ratio as presented in Table 4-22 and Figure 4-28.



Overall accuracy	0.978308
Average accuracy	0.99518
Micro-averaged precision	0.978308
Macro-averaged precision	0.981493
Micro-averaged recall	0.978308
Macro-averaged recall	0.977897

MLR classifier trained with 80:20 Training:Testing data

Confusion Matrix

Actual Class \ Predicted Class	1	2	3	4	5	6	7	8	9
1	100.0%								
2		100.0%							
3			100.0%						
4				100.0%					
5					100.0%				
6						100.0%			
7							80.1%	19.9%	
8								100.0%	
9									100.0%

Overall accuracy	0.964573
Average accuracy	0.992127
Micro-averaged precision	0.964573
Macro-averaged precision	0.971168
Micro-averaged recall	0.964573
Macro-averaged recall	0.964042

MNN classifier trained with 80:20 Training:Testing data

Confusion Matrix

Actual Class \ Predicted Class	1	2	3	4	5	6	7	8	9
1	100.0%								
2		100.0%							
3			100.0%						
4				100.0%					
5					100.0%				
6					4.0%	96.0%			
7							71.7%	28.3%	
8								100.0%	
9									100.0%

Figure 4-27 MLR vs MNN using split data ratios of 80:20: A figure of MLR vs MNN Confusion Matrix using training to testing split data ratios of 80:20.

Table 4-22 Performance metrics of MNN vs MLR using logistic normalisation: A table of performance metrics of MNN vs MLR using logistic normalisation 80:20 training: testing split data ratio.

Metrics	Training: Testing Split		% Improvement $((\boldsymbol{MLR\ 80{:}20}/\boldsymbol{MNN\ 80{:}20}) - \mathbf{1}) * \mathbf{100}$
	MNN 80:20	***MLR 80:20***	
Overall accuracy	0.964573	0.978308	1.42
Average accuracy	0.992127	0.99518	0.31
Micro-averaged precision	0.964573	0.978308	1.42
Macro-averaged precision	0.971168	0.981493	1.06
Micro-averaged recall	0.964573	0.978308	1.42
Macro-averaged recall	0.964042	0.977897	1.44
Total overall improvement of MLR 80:20 over MNN 80:20			**7.08%**

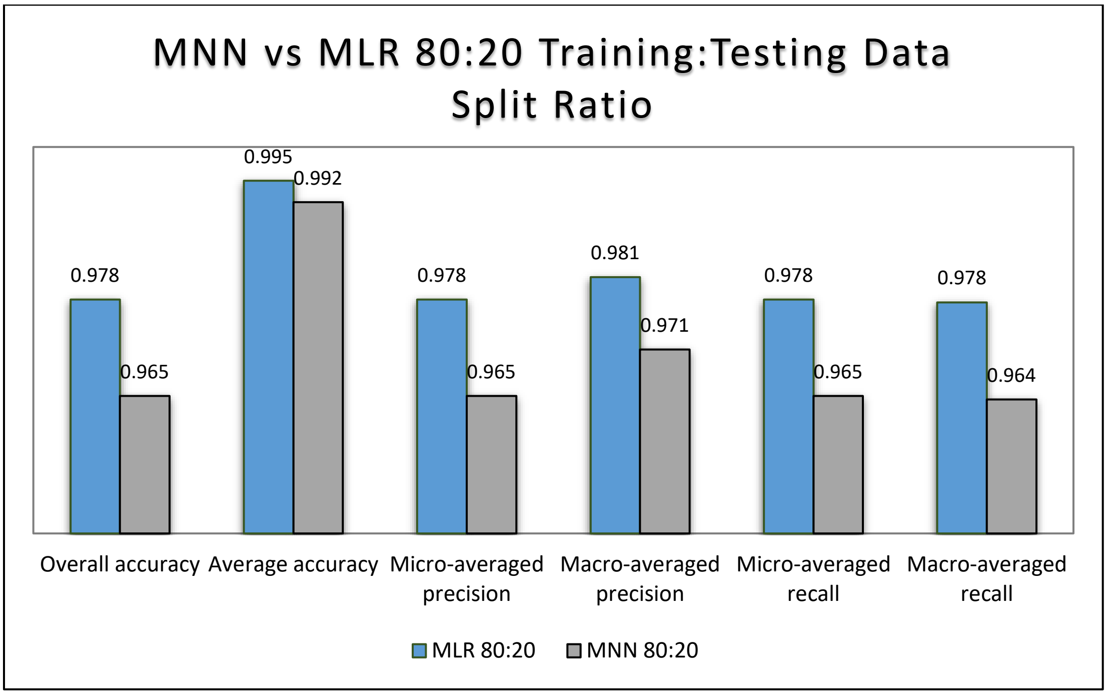


Figure 4-28 Bar Chart: MNN vs MLR using logistic normalisation: A Bar Chart of a better performance metrics of MLR over MNN using logistic normalisation 80:20 training: testing split data ratio.

4.5.4 Cross-validation

Figure 4-29 presents this section organisation chart with the associated subsections.

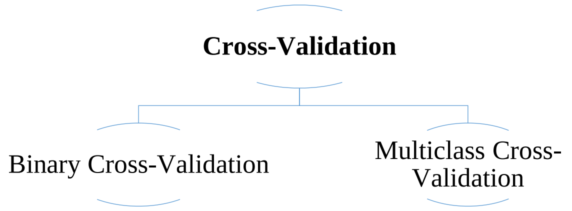


Figure 4-29 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

In the evaluations above, we embarked on the partitioning of the data set into training and testing set by split function at various ratios. The training set of the partitioned data is used to train the classifier while the testing set is the set unseen by the classifier which forms the subject of the empirical evaluations of the scoring methods presented above. The testing set is drawn from the underlying distribution as the training set would have inherent biases on the amount of the distribution that appears in a set at whatever data set split ratio which the cross-validation performance metrics provides insight.

The cross-validation goes beyond this traditional single training to testing set ratio evaluation to provide a more extensive evaluation across multiple folds of training to testing data split ratios. The cross-validation provides a better gauge of the performance metrics of a trained model with unseen data in a real-world scenario.

The Azure ML (MAML studio) provides a cross-validation module which uses by the default *k*-fold cross-validation where *k* is an integer. The MAML studio follows the favoured *k*-fold cross-validation *k =10* [48], [49], [52]. The cross-validation module provides a technique to reduce biases in a trained model that could occur when using a single data set [47]. Figure 4-31 below shows the *10*-fold cross-validation process of MAML studio defaults with the partitioning of the data set matrices into approximately equal folds where possible. Each of the fold is used as a test data across the remaining folds while iterating the number of times of *k* value (*10*). This process of fold held out for test ensures every fold of the observations is used as a testing data (not exposed to the classifier) while a union of the rest of the remaining nine folds are used as a training data (exposed to the classifier). A low standard deviation of accuracy value in MAML studio cross-evaluated model indicates a model with reduced biases capable of generalisation of a new data exposed to the trained model in the real-world.

Also provided in the cross-validation is the logarithmic loss (log loss) or cross-entropy to demonstrate the fitness of the pattern-driven data set proposed in this thesis. The log loss is a popular measure of classifier performance, which is an information theory measure of the cross-entropy between two probability distributions which is the encoded known outcome and the resultant score probabilities. In the Kaggle data science competition, it is the favoured performance metrics measure of a trained classifier [188], [189]. The MAML studio cross-validation module provides a log loss measure of an average log loss which is a single score for measuring the penalty of the result predicted wrongly [190].

The log loss and cross entropy have the same underlying mathematics to measure wrong prediction errors. The log loss term is used in the binary classification while cross entropy is used in multiclass classification. Log loss term is popularly used in interchangeably. The lower the average log loss, the better the prediction outcome [191]. The cross-validations presented below for binary and multiclass classifiers have a low average log loss value for each fold, which indicates a better model capable of generalisation of unknown data in the real-world using the pattern-driven data set to train a classifier proposal of this thesis. In the preceding training to testing data model evaluations above, we presented best-performance metrics to be binary classification algorithm (TC SVM) using training to test data (*70:30*) split ratio, while in the multiclass evaluation, MLR of training to test data (*80:20*) split ratio which is the subject of the cross-validation below.

4.5.4.1 Binary Cross-Validation

We derived a pattern-driven data set containing *59702* vectors with the original multiclass labelled data set of vectors of *1-9* reduced to a binary data set of *0* and *1* as presented in 4.4.2 above. We presented a TC SVM as the best performing trained model under logistic normalisation and training to testing split data ratio of *70* to *30*. We achieved high performance metrics with low prediction errors as shown AUC comparisons in Figure 4-21 above which is the subject of the cross-validation presented below.

Thus, the cross-validation is used to gauge the performance metrics of the unknown data exposed to the trained classifier in a real-world scenario. Figure 4-30 presents the cross-validation integration with the MAML studio predictive experiment presented in Figure 4-8 above. The figure illustrates the normalised data set connected to the cross-validation module that is trained with TC SVM.

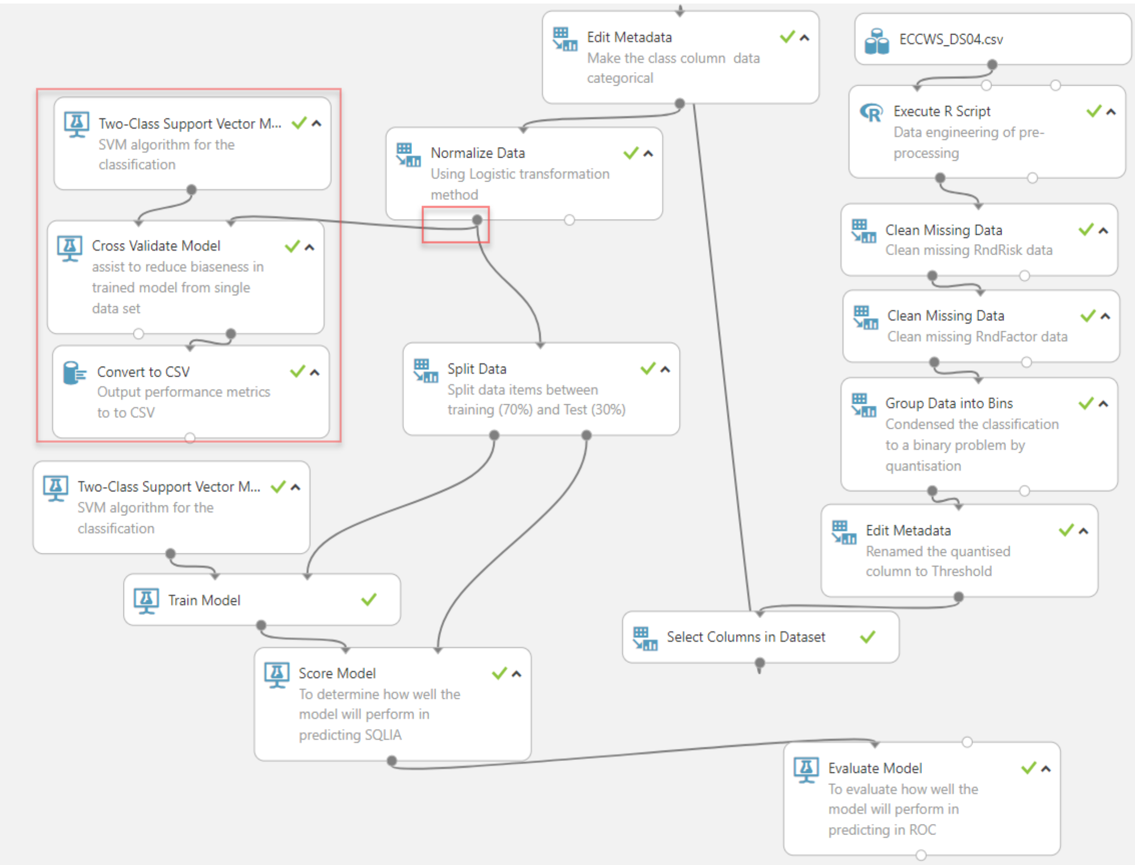
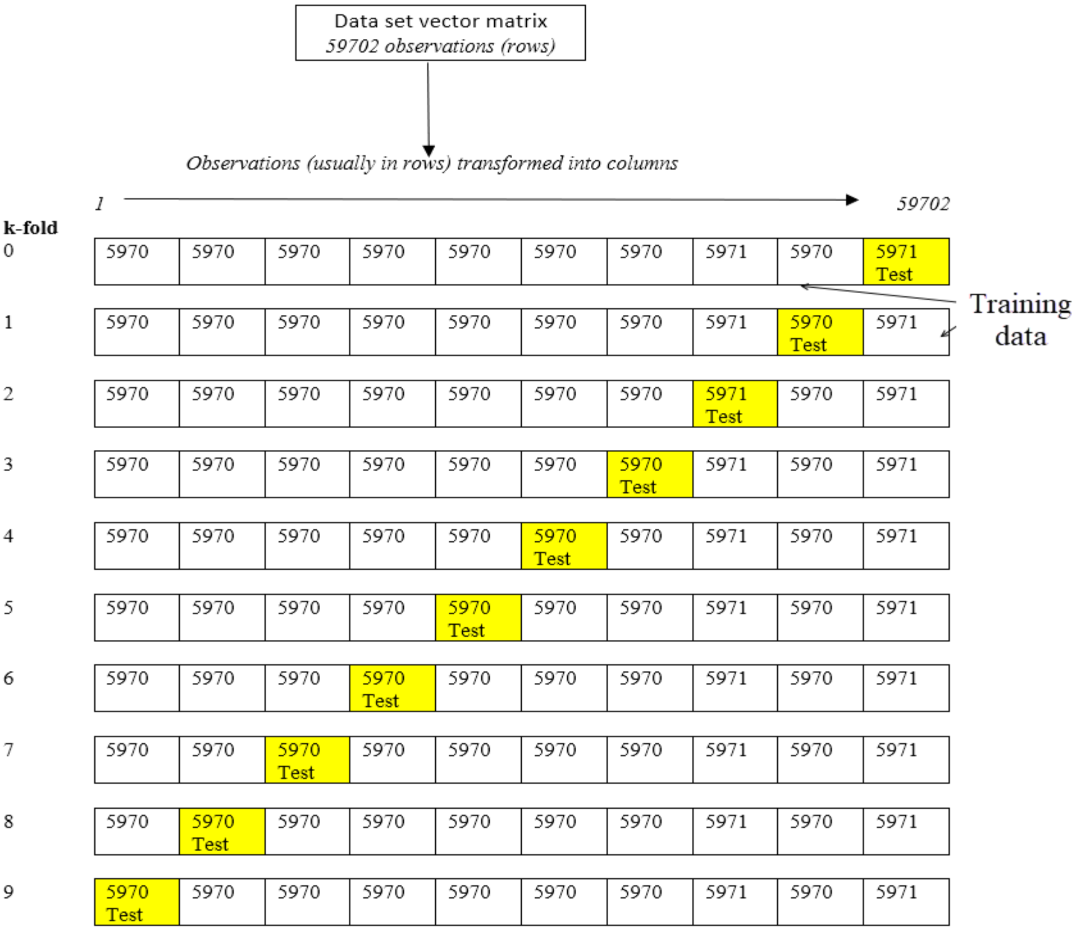

Edit Metadata
Make the class column data categorical
ECCWS_DS04.csv
Two-Class Support Vector M...
SVM algorithm for the classification
Normalize Data
Using Logistic transformation method
Execute R Script
Data engineering of pre-processing
Cross Validate Model
assist to reduce biaseness in trained model from single data set
Clean Missing Data
Clean missing RndRisk data
Clean Missing Data
Clean missing RndFactor data
Convert to CSV
Output performance metrics to to CSV
Split Data
Split data items between training (70%) and Test (30%)
Group Data into Bins
Condensed the classification to a binary problem by quantisation
Two-Class Support Vector M...
SVM algorithm for the classification
Edit Metadata
Renamed the quantised column to Threshold
Train Model
Select Columns in Dataset
Score Model
To determine how well the model will perform in predicting SQLIA
Evaluate Model
To evaluate how well the model will perform in predicting in ROC

Figure 4-30 A cross-validation experiment screen capture: A screen capture of the cross-validation to the MAML studio predictive experiment presented in Figure 4-8 above.


Data set vector matrix
59702 observations (rows)
Observations (usually in rows) transformed into columns
1
59702
Training
data

k-fold										
0	5970	5970	5970	5970	5970	5970	5970	5971	5970	5971 Test
1	5970	5970	5970	5970	5970	5970	5970	5971	5970 Test	5971
2	5970	5970	5970	5970	5970	5970	5970	5971 Test	5970	5971
3	5970	5970	5970	5970	5970	5970	5970 Test	5971	5970	5971
4	5970	5970	5970	5970	5970	5970 Test	5970	5971	5970	5971
5	5970	5970	5970	5970	5970 Test	5970	5970	5971	5970	5971
6	5970	5970	5970	5970 Test	5970	5970	5970	5971	5970	5971
7	5970	5970	5970 Test	5970	5970	5970	5970	5971	5970	5971
8	5970	5970 Test	5970	5970	5970	5970	5970	5971	5970	5971
9	5970 Test	5970	5970	5970	5970	5970	5970	5971	5970	5971

Figure 4-31 A representation of binary k-fold cross-validation: The figure shows the 10-fold cross-validation process of MAML studio defaults with the partitioning of the data set matrices into approximately equal folds where possible.

The cross-validation usually takes the input of the entire data set vectors or vector matrices and then partition it into folds for the rigorous training of the presented algorithms. Figure 4-31 illustrates with each run, a partition (fold) is held out as the testing data, and the union of the rest of the nine folds as the training data. It is a similar concept of training to testing split data at various ratios across the data set split partition, thereby eliminating biases of the distribution of features at any ratio of the data set partition.

The k-fold cross-validation provides performance metrics for measuring bias-variance trade-off. A low variance or standard deviation in accuracy indicates of a successful trained model capable of generalisation of unknown data in the real-world. The k-fold cross validation is akin to classic split training/testing data evaluation model, but with a more accurate estimate of out-of-sample accuracy and efficiency in ensuring every observation is used for both training and testing. Out-of-sample accuracy is a measure of the ability of a trained model to cope with unknown data to the classifier (i.e. akin to testing data) or real-world data generalisation when the trained model is deployed.

We observed a low standard deviation of accuracy across the ten folds indicating an excellent trained model. This result indicates a model with a reduced bias in data set features variation and distribution of the statistical measures that can generalise independent data set in a real-world scenario that verifies this thesis goal of a pattern-driven data set can be used to train a classifier and be validated.

Equation 4-10 presents the formula for calculating the mean μ, where μ is the mean, x_i are the accuracy values across the ten folds (*0.9772*, *0.9995*, *0.9844*, *0.9853*, *0.9625*, *0.966*, *0.9854*, *0.9797*, *0.9739*, *0.9256*), *n* is the iteration times or folds which is *n=10* and therefore μ =9.7395/10 ≈ 0.974. Equation 4-11 presents the equation for calculating the variance or standard deviation σ, where σ is the variance calculated to be *0.02*. We achieved a lower variance of accuracy value of *0.02* which indicates a less biased trained model capable of generalisation of unknown independent data with the results of the test performed presented in Table 4-23. The table also contains a high performance metrics for Precision, Recall, F-Score and AUC, including a low average log loss values to indicate of a trained model with a low penalty for wrong score probabilities.

$$\mu = \frac{\Sigma x_i}{n}$$ **Equation 4-10**

$$\sigma = \sqrt{\frac{\Sigma(x_i - \mu)^2}{n}}$$ **Equation 4-11**

Table 4-23 A table of binary evaluation results by folds: A table containing evaluation results of the data set matrices partitioned into 10 folds with a low standard deviation of accuracy highlighted to indicate of a trained model capable of generalisation of unknown independent data.

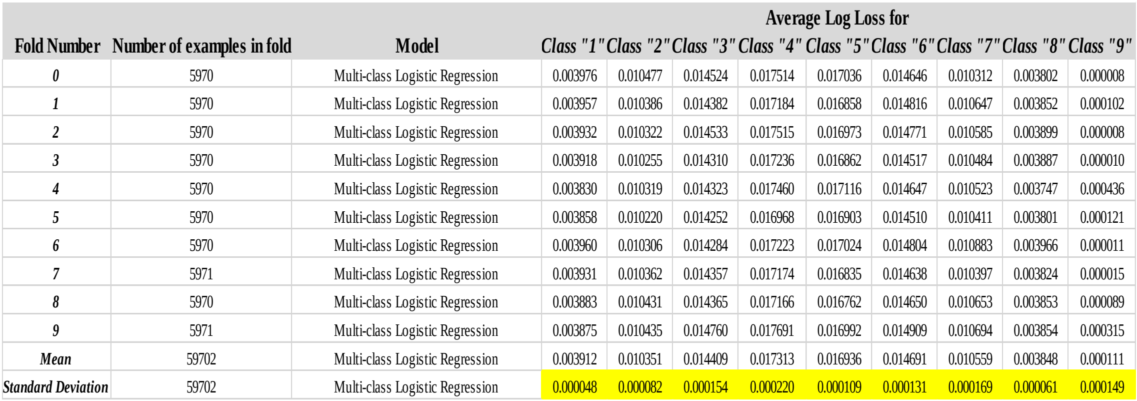
Fold Number	Number of examples in fold	Model	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss
0	5970	SVM (Pegasos-Linear)	0.977	0.989	0.986	0.987	0.997	0.040
1	5970	SVM (Pegasos-Linear)	1.000	1.000	1.000	1.000	1.000	0.002
2	5970	SVM (Pegasos-Linear)	0.984	0.991	0.992	0.991	0.999	0.031
3	5970	SVM (Pegasos-Linear)	0.985	0.992	0.992	0.992	0.999	0.028
4	5970	SVM (Pegasos-Linear)	0.963	0.976	0.982	0.979	0.993	0.068
5	5970	SVM (Pegasos-Linear)	0.966	0.979	0.983	0.981	0.994	0.060
6	5970	SVM (Pegasos-Linear)	0.985	0.991	0.992	0.992	0.998	0.033
7	5971	SVM (Pegasos-Linear)	0.980	0.990	0.988	0.989	0.998	0.039
8	5970	SVM (Pegasos-Linear)	0.974	0.986	0.984	0.985	0.997	0.045
9	5971	SVM (Pegasos-Linear)	0.926	0.928	0.994	0.960	0.866	0.227
Mean	59702	SVM (Pegasos-Linear)	0.974	0.982	0.989	0.986	0.984	0.057
Standard Deviation	59702	SVM (Pegasos-Linear)	*0.020*	*0.020*	*0.006*	*0.011*	*0.042*	*0.062*

4.5.4.2 Multiclass Cross-Validation

In the preceding sections, we derived a pattern-driven data set containing *59702* vectors with multiclass labels of 1-9 that account for SQLIA types signature and valid web requests to train multiclass classifiers. We presented MLR as the best-performing classifier under 80:20 training to testing data split ratio in section 4.5.3.1.3 above. Below are the results of cross-validation across the ten folds using the MAML studio. The underlying mathematics can be found in the following references [188], [189], [191], [192].

Table 4-24 presents the multiclass cross-evaluation across the ten folds with a small average log loss values across the nine classes (*1-9*) highlighted to indicate of a trained model with a low penalty for wrong score probabilities capable of generalisation of unknown independent data.

Table 4-24 A table of multiclass evaluation results by folds: The table presents evaluation results of the data set matrices partitioned into ten folds with a small average log loss values across the nine classes highlighted to indicate of a trained model with a low penalty for wrong score probabilities.



			Average Log Loss for								
Fold Number	Number of examples in fold	Model	*Class "1"*	*Class "2"*	*Class "3"*	*Class "4"*	*Class "5"*	*Class "6"*	*Class "7"*	*Class "8"*	*Class "9"*
0	5970	Multi-class Logistic Regression	0.003976	0.010477	0.014524	0.017514	0.017036	0.014646	0.010312	0.003802	0.000008
1	5970	Multi-class Logistic Regression	0.003957	0.010386	0.014382	0.017184	0.016858	0.014816	0.010647	0.003852	0.000102
2	5970	Multi-class Logistic Regression	0.003932	0.010322	0.014533	0.017515	0.016973	0.014771	0.010585	0.003899	0.000008
3	5970	Multi-class Logistic Regression	0.003918	0.010255	0.014310	0.017236	0.016862	0.014517	0.010484	0.003887	0.000010
4	5970	Multi-class Logistic Regression	0.003830	0.010319	0.014323	0.017460	0.017116	0.014647	0.010523	0.003747	0.000436
5	5970	Multi-class Logistic Regression	0.003858	0.010220	0.014252	0.016968	0.016903	0.014510	0.010411	0.003801	0.000121
6	5970	Multi-class Logistic Regression	0.003960	0.010306	0.014284	0.017223	0.017024	0.014804	0.010883	0.003966	0.000011
7	5971	Multi-class Logistic Regression	0.003931	0.010362	0.014357	0.017174	0.016835	0.014638	0.010397	0.003824	0.000015
8	5970	Multi-class Logistic Regression	0.003883	0.010431	0.014365	0.017166	0.016762	0.014650	0.010653	0.003853	0.000089
9	5971	Multi-class Logistic Regression	0.003875	0.010435	0.014760	0.017691	0.016992	0.014909	0.010694	0.003854	0.000315
Mean	59702	Multi-class Logistic Regression	0.003912	0.010351	0.014409	0.017313	0.016936	0.014691	0.010559	0.003848	0.000111
Standard Deviation	59702	Multi-class Logistic Regression	0.000048	0.000082	0.000154	0.000220	0.000109	0.000131	0.000169	0.000061	0.000149

4.6 Discussion

A review of the previous work on SQLIA up to the time of submitting the manuscripts for two conferences' papers [61], [62] related to this thesis chapter in 2015, to the best of our knowledge, no existing work has leveraged the ML cloud-hosted platform like Azure ML (MAML studio) to address SQLIA issue. One issue that stands out in some existing approaches on SQLIA mitigation [74], [125], [129], [133], [193] is computationally expensive. Availability of ML cloud-based software and platform as services resolves the issue by harnessing these large-scale supercomputers employing AI techniques to deliver ready to use solutions that would have been limited relying on a single or few servers in the intranet or extranet.

The other issue that currently stands out is the lack of the availability of up to date data set which there wouldn’t be as web application types are so diverse to have all-purpose data set to apply AI. However, just as the KDD Cup 1999 [161] has become antiquated in the IDS domain [1], [2], [67], [160], so also has ECML/PKDD 2007 [65] and HTTP CSIC 2010 [66] has become obsolete. These test case data sets served the competition-driven purpose and textbook evaluations at the time, not a benchmark data set to build and evaluate SQLIA mitigation being provided to the real-world application. Also, these competitions-driven data sets are not relevant to benchmarking of any sort in this thesis as it is recognised that web application types are diverse and this thesis provides a pattern-driven approach for deriving a data set on the fly based on the web application type being protected.

Thus, this begs the question if the pattern that exists in web requests that are deemed valid, and historical studies of different SQLIA types including SQL tokens can provide artefact to be used to derive a pattern-driven data set to train a supervised learning model? The ontology to extract encoded features that form a data set earmarked to train an ML model with evaluation for a web application type in towards SQLIA mitigation is a contribution of this chapter.

Web applications and services are now increasingly hosted in the cloud with restricted access to source codes for static and dynamic analysis. Therefore, whilst existing approaches [4], [5], [95]–[104], [6], [105]–[109], [119]–[123], [7], [124]–[127], [174], [194], [195], [8], [10], [82], [84], [94] seemed plausible in the past, they are not scalable in emerging computing with big data. These existing approaches have existed in an era of manageable data size. Also, the shifting of web application domain boundary from the intranet or extranet to the cloud SDN for cloud-hosted services makes approaches that rely on static and dynamic source code scanning not applicable over time.

Also, a review of solely SQLI mitigation over the years is lacking in the big data context. The few that have data mining in the title were database level mitigation applying ML of a sort of query comparison not amenable to cloud SDN with sizeable incoming web requests hit for predictive analytics. These existing approaches [129], [130] gravitate to SQLIA mitigation proposals of query comparison of some sort, mostly driven by the method of the data set procurement through using web crawler of URLs and other automated unit testing tools. These test case data sets that are logged valid SQL queries for comparison between valid and malicious queries during runtime does not have patterns for predictive analytics as presented in this thesis.

In comparing the proposal presented in this chapter with existing approaches; this chapter focused on web application type to provide a pattern-driven data set as against a data set generated with repeating strings as seen in most existing work [71]–[75], [130], [133], [193]. The pattern-driven data set is used to train supervised learning classification algorithms with extensive validation through statistical measures in the context of emerging computing of big data cloud-hosted services as against existing approaches on ML which does not share similarities.

Table 4-25 highlights the pattern-driven data set, code access and emerging computing cloud SDN focused, in towards SQLIA mitigation presented in this chapter with the related academic papers [61], [62] sets apart this chapter proposal from existing approaches that apply AI over the years.

Table 4-25 Related work applying ML comparison: A table comparing the existing ML approaches to SQLIA mitigation in the context of source code access, cloud SDN applicable and pattern-driven data set with related publications to this chapter highlighted.

Authors	Algorithms	Need access to source code	Intranet / intranet domain boundary	Emerging computing applicable: cloud SDN applicable	Pattern-driven data
Bockermann et al. [75]	SVM	Yes	Yes	No	No
Komiya [71]	SVM	Yes	Yes	No	No
Choi et al. [72]	SVM	Yes	Yes	No	No
Wang and Li [73]	SVM	Yes	Yes		No
Pinzón et al. [74], [141], [193]	SVM and NN	Yes	Yes	No	No
Kim and Lee [138]	SVM	Yes	Yes	No	No
Uwagbole et al. [62]	LR and NN	No	Yes	Yes	Yes
Uwagbole et al. [61]	SVM	No	Yes	Yes	Yes

4.7 Conclusion

In conclusion, the work presented in this chapter demonstrates the fitness of a pattern-driven data set to train a supervised learning model employing both statistical binary and multiclass algorithms implemented in MAML studio. We presented statistical measures, ROC curve, CM and cross-validation results of the empirical evaluation of the proposed scheme. The proposal presented in this chapter is towards scalable and functional prediction of SQLIA, demonstrated in further work presented in Chapters 5 and 6.

This approach presented in this thesis is geared towards leveraging recent advancement in the field of AI to build a scalable application that can predict SQLIA in web requests with a high degree of accuracy in TP and TN but with low prediction errors of FP and FN as presented above.

The work presented in this chapter presents the procurement of a pattern-driven data set to extrapolate vectors of any size in towards predicting SQLIA. We leverage the MAML AI cloud-hosted platform employing ML and ANN algorithms to train the pattern-driven data set with empirical statistical measures of high performance metrics presented in the ROC curve, CM and cross-validation. These contributions distinguish the work presented here from previous research proposals by other authors.

On account of the preceding discussion, this chapter answers the research question can pattern in both expected web requests and SQL tokens including existing SQLIA signature extraction be used to create a large volume of learning data of encoded numeric vector variables required to train a supervised learning model and be validated? This research question is answered in the following contributions of this chapter.

- The ontology for crafting a pattern-driven data set from the web application type with a technique to encode the data set into vectors required to train a supervised learning model using emerging cloud-hosted ML platforms of Azure ML (MAML studio).
- We trained a supervised learning classification algorithms with this pattern-driven data and validated the trained model under various classification algorithms with high performance metrics in the ROC curve and CM respectively, including cross-validation as presented in this chapter. The success of this conceptual approach of the web application type as the source of a pattern-driven data set to train a classifier has led to further work in the Chapters 5 and 6.

The further work in the Chapters 5 and 6 involves employing string hashing vectorisation in-place of the numeric encoding presented in this chapter, and a proof of concept of how the proposal will be applied in a real-world application scenario.

5 Applied Predictive Analytics to Mitigate SQLIA

Figure 5-1 presents the organisation chart layout of this chapter with the sections.

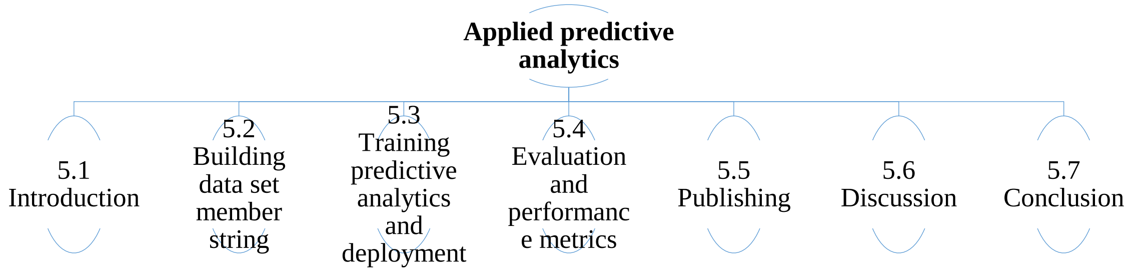

Applied predictive analytics
5.1 Introduction
5.2 Building data set member string
5.3 Training predictive analytics and deployment
5.4 Evaluation and performanc e metrics
5.5 Publishing
5.6 Discussion
5.7 Conclusion

Figure 5-1 The chapter organisation chart: The chapter organisation chart presents the layout of the chapter's sections.

5.1 Introduction

Figure 5-2 presents this section organisation chart with the associated subsections.

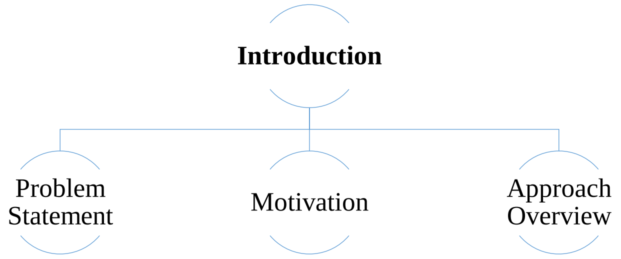


Figure 5-2 The section organisation chart: A figure of the section organisation chart that provides the layout of this section with the associated subsections.

This chapter answers the research question if the pattern-driven data set can be extracted from a web application type context to detect and prevent SQLIA? We present in this chapter a method of applying predictive analytics to predict and prevent SQLIA in both legacy and new web applications in the big data context using a pattern-driven data set. The technique applies AI to a web application domain context that does not rely on access to the web application source code and queries in SQLIA mitigation as proposed by most existing research work on SQLIA.

This chapter starts with the introduction in sections 5.1 and ends with a conclusion in 5.7. Sections 5.2 and 5.3 discuss building the data set member strings and predictive analytics experimental steps while section 5.4 presents evaluations and performance metrics. Section 5.5 discusses the publishing and consuming of the trained, supervised learning model in an application for ongoing SQLIA mitigation while 5.6 discuss the issues with the existing work to highlight the contribution of this chapter.

5.1.1 Problem statement

In this research question, we ask if a pattern-driven data set can be extracted from a web application type being provided with SQLIA mitigation?

SQLIA vulnerability is a sequel to a design fallout of the well-intentioned free text processing of the SQL engine itself, and as a consequence both legacy and cloud deployments lacking sanitisation become vulnerable. A search of “SQLI hall of shame” [21] which reports the recent trends in data pilfering by SQLIA shows the prevalence of this form of attack and so the ability to secure back-end database from SQLIA in an era of big data remains a topical issue.

The SQL language syntax closely resembles plain English [196], and the SQL keywords are also in plain text. Therefore, the SQLIA problem in a big data context is a plausible candidate for predictive analytics of a supervised learning model trained via both known historical attack signatures and safe web request patterns. We propose that string pattern exists in every input data in both legacy and new web applications and that these can be leveraged to generate as many derivations of related member strings. Applying ML techniques requires data set with sufficient learning data arising from patterns that exist in input data provided in any web application domain.

5.1.2 Motivation

The existing solutions of mostly heuristic signature approaches were all before the new emerging computing in big data mining and as such lack the functionality and ability to cope with new signatures concealed in web requests in the context of big data. Also, these existing approaches were aimed at on-premise web application domain boundary, not roaming cloud-hosted applications and services across the internet including SDN.

ML predictive analytics provides functional and scalable mining for big data in the detection and prevention of SQLIA. Unfortunately, lack of availability of a ready-made existing robust data set with patterns and historical attributes to train a classifier are issues well known in SQLIA research in applying AI techniques. The few test case data sets [65], [66] available are obsolete and have served the competition-driven evaluations.

5.1.3 Approach overview

A literature review of existing SQLIA detection and prevention approaches discussed in Chapter 3 reveals three perspectives: static SQLIV detection by peering through the code; code sanitisation prescribed by OWASP during SDLP in SQLIA prevention; and a combined static, and runtime analysis by queries comparison between static and dynamic in the detection and prevention of SQLIA [94], [95], [101],[102], [108]. It is much harder, if not impossible in cloud-hosted services to access free-flowing web application source code which negatively affects some existing solutions in mitigating SQLIA. Also, there is such big data web traffic to these cloud-hosted web applications and services for these existing approaches to be functional. The big data trend will only head one direction which is the upward growth, and as such will require scalable techniques to mine big data for security vulnerabilities.

In this chapter, we further the technique discussed in section 4.3 above in the generation of the data set containing string extraction from patterns in web application type context and known attack patterns including SQL tokens.

Also, as a test case, we build a web application that expects an English dictionary word as vector variables to demonstrate massive quantities of learning data. We derived an equal distributed data set of *725206* data items that is split equally between attack/respondent (*362603* positives) and non-attack/non-respondent (*362603* negatives) from the labelled preprocessed features that are hashed as the input to the classifier. The trained classifier to be deployed as a web service that is consumed in a web form for input validation, and at a custom .NET application implementing a web proxy API to intercept and accurately predict SQLIA in web requests thereby preventing malicious web requests from reaching the protected back-end database. This full proof of concept implementation of an ML predictive analytics and deployment of resultant web service that accurately predicts and prevents SQLIA is the subject of the empirical statistical measures evaluations presented in AUC, ROC curve and cross-validation including comparison with existing work in this chapter.

There is a need to build a prediction model trained on data set of the web application type context expected data to predict SQLIA. The attack signatures presence at injection points will contain patterns of SQL tokens as SQLIA positive while valid web requests would take the form of generating all possible string members by string replication and transposition as discussed in section 4.3.1 and below. We trained a supervised learning model in applying predictive analytics to a test web application with large quantities of learning data derived from related member strings. The learning data are labelled features of both patterns of dictionary word list (SQLIA negative) and SQL tokens (SQLIA positive).

On account of the drawback of most existing solutions, we presented in this thesis an applied predictive analytic approach that does not rely on any source code or query structure to detect and prevent SQLIA in web requests. The work presented here analysed the data input patterns of the web application domain to be monitored and protected as to generate labelled features. The generated labelled data set are preprocessed and vectorised by applying a hashing technique to obtain numeric vector matrix input required to train an SVM classifier.

The published web service is consumed in a proxy API on the server side that intercepts web requests to predict SQLIA thereby preventing malicious web requests from reaching the back-end database. Also, the trained prediction web service is exposed to the client web form validation which will strengthen any existing validation in place in the internet application as shown in Figure 5-3.

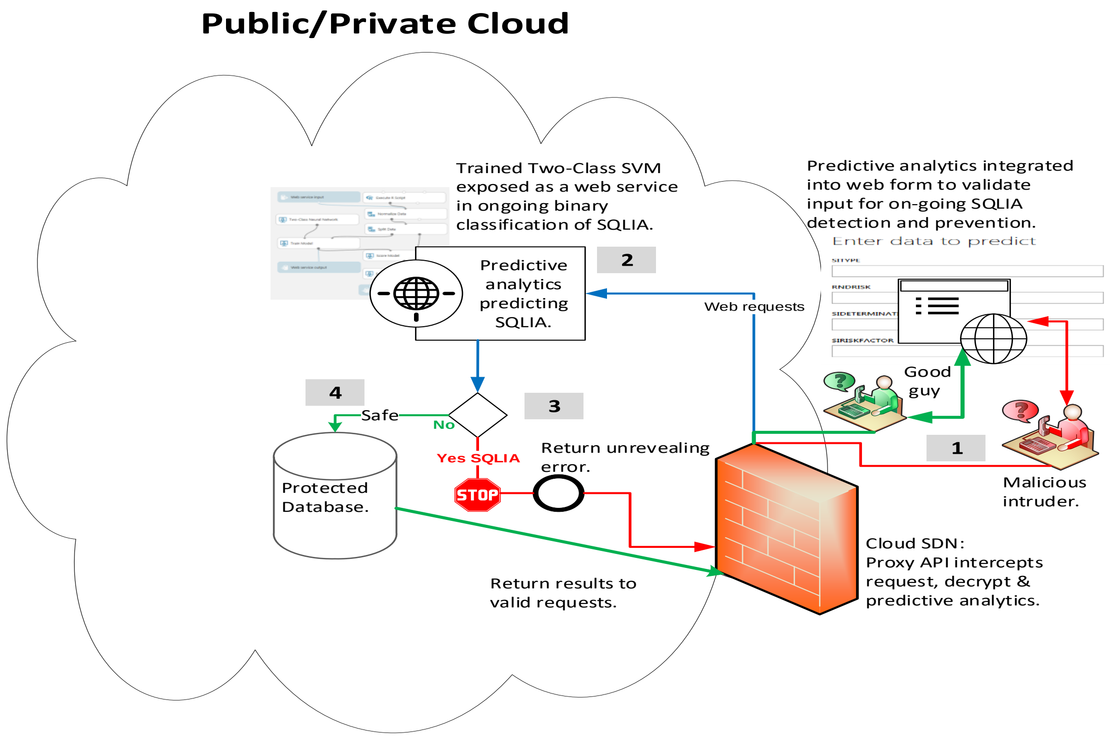

Public/Private Cloud
Trained Two-Class SVM exposed as a web service in ongoing binary classification of SQLIA.
Predictive analytics integrated into web form to validate input for on-going SQLIA detection and prevention.
Enter data to predict
SITYPE
RNDRISK
SIDETERMINAT
SIRISKFACTOR
Predictive analytics predicting SQLIA.
2
Web requests
Good guy
1
Malicious intruder.
4
Safe
No
3
Yes SQLIA
STOP
Return unrevealing error.
Protected Database.
Return results to valid requests.
Cloud SDN: Proxy API intercepts request, decrypt & predictive analytics.

Figure 5-3 A prototype of the proposed model: The figure displays the prototype of the proposed model illustrating the design overview, including consuming a trained ML model in client forms and at the web proxy API in an ongoing SQLIA detection and prevention.

Figure 5-3 is a schematic diagram of the model proposed in this thesis chapter detailing how a trained model exposed as a web service will be consumed in client forms for input validation and at the proxy which intercepts the web requests for SQLIA analytics.

5.2 Building a data set member string

Figure 5-4 presents this section organisation chart with the associated subsections.

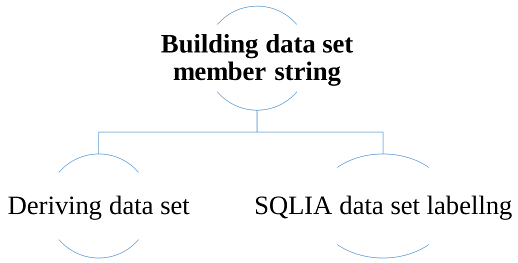


Figure 5-4 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

When new web applications that accept input from web forms are deployed, the initial state of the database is empty until the input data starts being received. Until then, we are left with a blank or empty data set without precedence to train a supervised learning prediction model.

On account of the lack of data set, and the security loophole of directly using a sample input data without patterns (with the potential ramifications of barring confidential data in learning data), we explore string replication and transposition in generating all possible derivations from expected input data including SQLIA signatures and SQL token features to derive a data set.

5.2.1 Deriving the data set

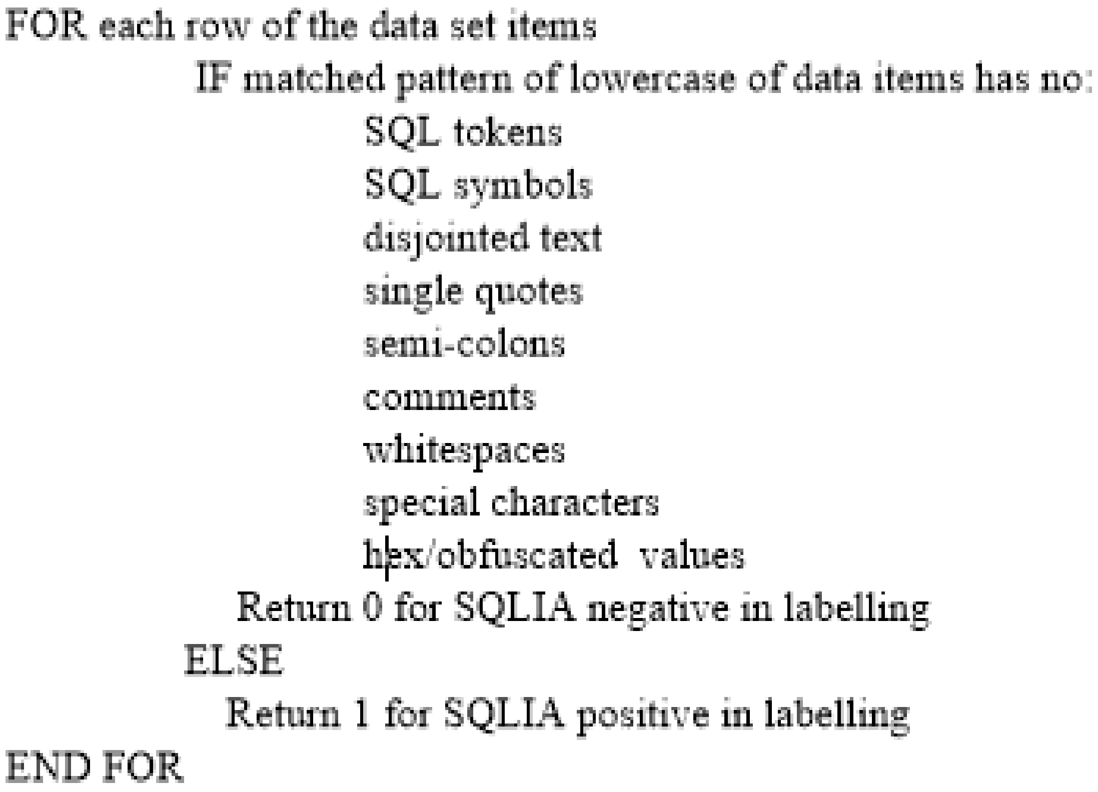
In Chapter 4, we obtained the vectors to train a supervised learning model by numerical attributes encoding ontology of the string attribute values (both legitimate web requests and SQLIA types, including SQL tokens) as to extract a scalable numerical data set. In this chapter, we directly explore hashing or vectorisation instead of scalar observations to derive the vectors required to train an ML model as detailed in section 5.3.3 below.

5.2.2 SQLIA Data set and Labelling

The data set is comprised of an English dictionary word list (labelled SQLIA negative) and SQL tokens (labelled SQLIA positive if present at injection points). The labelling procedure is described in Figure 5-5. When this approach is applied to any web application domain context, a data set can be generated based on the pattern of data input expected. Also, the existing studies on SQLIA types provide additional patterns to labelled SQLIA positive.

The aim is to infer and derive a pattern-driven data set from the intended web application being protected. This pattern-driven data set is then used to train the ML classification algorithm to predict SQLIA in intercepting web requests. The Fiddler proxy API [197] provides the functionality of intercepting and decrypting web requests.

Researchers attempting AI techniques have explored various techniques to extract sample data set with most researchers resorting to network tools to produce test cases of web requests. These test cases learning data extraction techniques often result in mere duplication of the same strings, but lack patterns for improving the performance of the ML models. Most existing SVM implementation lacks completeness in not moving beyond the CM or ROC evaluation to apply the technique in real life scenarios [71], [72], [134]–[141], [74], [100], [128]–[133].



```
FOR each row of the data set items
        IF matched pattern of lowercase of data items has no:
                SQL tokens
                SQL symbols
                disjointed text
                single quotes
                semi-colons
                comments
                whitespaces
                special characters
                hex/obfuscated  values
          Return 0 for SQLIA negative in labelling
        ELSE
          Return 1 for SQLIA positive in labelling
END FOR
```

Figure 5-5 Data set features labelling procedure: A figure illustrating data set features labelling procedure to obtain the labelled data set used in ML.

5.3 Training Analytics experiment and deployment

Figure 5-6 presents this section organisation chart with the associated subsections.

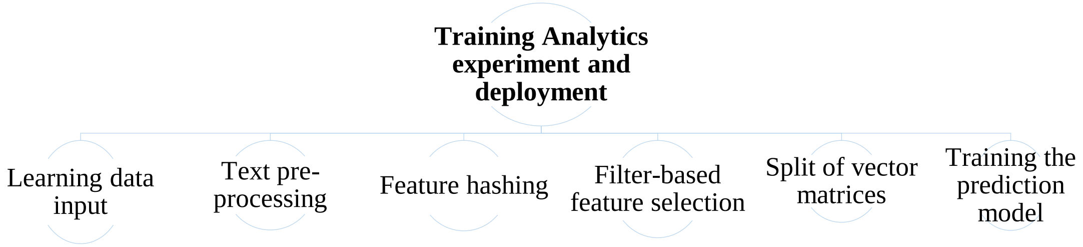
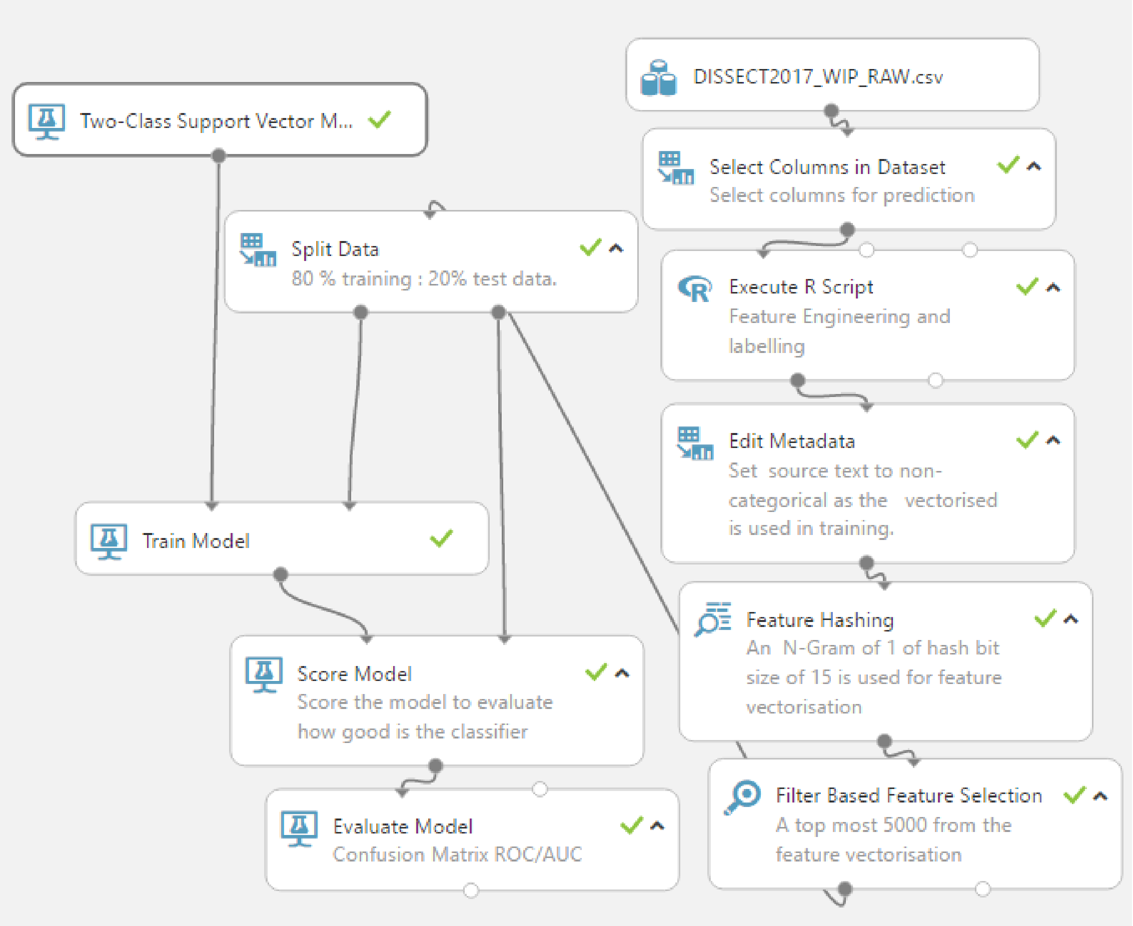

Training Analytics experiment and deployment
Learning data input
Text pre-processing
Feature hashing
Filter-based feature selection
Split of vector matrices
Training the prediction model

Figure 5-6 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

Predictive analytics provide a scalable and functional approach to big data mining. We apply predictive analytics in mitigating SQLIA vulnerabilities. The approach is built on MAML studio which is a cloud-based machine learning platform. The knowledge required to implement our approach is a programming knowledge of C#, R script and RegEx. The experimental steps are listed and shown in Figure 5-7 and are further described below.



DISSECT2017_WIP_RAW.csv

Two-Class Support Vector M... ✓

Select Columns in Dataset ✓
Select columns for prediction

Split Data ✓
80 % training : 20% test data.

Execute R Script ✓
Feature Engineering and labelling

Edit Metadata ✓
Set source text to non-categorical as the vectorised is used in training.

Train Model ✓

Feature Hashing ✓
An N-Gram of 1 of hash bit size of 15 is used for feature vectorisation

Score Model ✓
Score the model to evaluate how good is the classifier

Filter Based Feature Selection ✓
A top most 5000 from the feature vectorisation

Evaluate Model ✓
Confusion Matrix ROC/AUC

Figure 5-7 The experimental steps applying text analytics to SQLIA: A figure of a screen capture of the experimental steps applying text analytics to SQLIA.

5.3.1 Learning data input

This research uses a pattern-driven data set which contains *479,000* English dictionary words (that are transformed and transposed into related member strings) in addition to 862 unique SQL tokens extracted from Microsoft SQL reserved keywords website [179]. The data set items are labelled based on the exhibition of SQLIA type characteristics which, to give examples, are: the presence of SQL tokens in injection point; disjointed text; single quotes; semicolons; comments; and hex values. The data set items labelling is represented in binary values of *0* (SQL negative) or *1* (SQL positive) shown in Figure 5-5 above.

5.3.2 Text pre-processing

This stage involves R Scripting incorporating all the defined regular expression pattern constraints to generate the learning data to train a model for ongoing detection and prevention of SQLIA. In a real-world application context, the data set items are expanded as deemed required with patterns of both valid and malicious requests.

There were *362603* row items after text pre-processing of parsing data set for: patterns, duplicates, normalised to lower cases and the removal of the missing words. The data set is sampled as to provide an even distribution of row items (records). The imbalanced data set (majority negatives over positives) were corrected with SMOTE [187] to have *725206* data items that are split equally between attack/respondent (positives) and non-attack/non-respondent (negatives). These actions improve both the trained model recall and precision.

5.3.3 Feature hashing

ML takes in numerical input as vectors. We reviewed the work by Choi et al. [72] using N-grams to hash query strings which could be circumvented by inserting more empty spaces into a group of strings. In this scheme, we set the hashing bits to 15 and N-grams to 1 (unigram) to analyse every single word item present at the injection point as the trained model is aimed to be deployed in the web form for input validation and at the proxy which intercepts web requests for predicting SQLIA.

The feature hashing provides the technique to translate the data set text items into binary vector matrices of 2^{15} *(32,768)* columns suitable for training a model in ML. The hashing procedure creates a dimensional input vector matrix that does a lookup of feature weights faster by augmenting the string comparison with the hash value comparison. Applying features hashing to text features improves performance and scalability in big data predictive analytics lacking in existing SQLIA signature-based detections that use a string lookup approach by creating several loops in programming logic to detect SQLIA signatures.

5.3.4 Filter-based feature selection for top relevant vectors

Creating a dimension to accommodate the size of data by selecting next hashing bits that fit the data set can sometimes generate too much dimension and sparse data which are reduced by a filter-based feature selection. In this experiment, we used filter-based selection to have reduced computational complexity without affecting the prediction accuracy in classification. The Chi-squared [198] scored function is used to rank the top *5000* hashing features in descending order to return the most appropriate labels to improve SQLIA prediction accuracy.

5.3.5 Split of vector matrices between training and testing data

We divide the vector matrices from the hashed features into *80:20* training to testing data ratio for the evaluations after experimenting with various split ratios which were observed to provide excellent performance metrics. Also provided in this chapter is the cross-validation, which addressed the distribution biases of the training to testing data split ratio.

5.3.6 Training the prediction model

The TC SVM classifier is used to predict the labelled binary outcomes, whether SQLIA is negative or positive in a web request. The SVM algorithm is provided with data set items input of the labelled class of what is being predicted to train the model to an excellent performance to accurately predict SQLIA in web form validation and at proxy intercepted web requests.

5.4 Evaluation and performance metrics of Binary TC SVM

Figure 5-8 section organisation chart presents the layout of this section with the associated subsections.

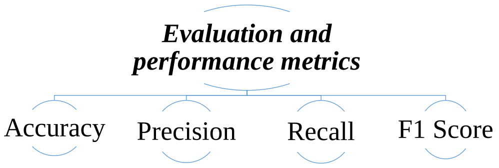


Figure 5-8 The section organisation chart: The section organisation chart provides this section layout with the associated subsections.

The ROC curve and the related AUC provide a way to statistically measure how well a binary learning model is performing [46]. According to Tape "*ROC curve was developed during World War 11 to predict images if a blip on the screen was a threat target, a friendly ship or a harmless noise. Signal detection theory measures the ability of radar receiver operators to make these important distinctions. Their ability to do so was called the Receiver Operating Characteristics. It was not until the 1970's that signal detection theory was recognised as useful for interpreting medical test results*" [50], [51].

The AUC and ROC curve provides a plausible quantitative evaluation of a binary prediction model. In this scenario, whether a web request is malicious SQLIA positive or not SQLIA negative.

An AUC summary value provides a concrete measure for assessing a classifier performance as it presents a unified measure of the separability between the distributions of score probabilities for SQLIA positive and negative. We achieved an AUC of the value of *0.986* in the trained, supervised learning model presented in the chapter experimental results. The ROC curve presents a measure of FPR on the x-axis against TPR on the y-axis with values across different score bins or thresholds.

We use the ROC curve and AUC to measure the performance metrics of our trained, supervised learning model and observed an excellent model with an AUC value of *0.986* with cross-validation of low variance indicating of a model capable of generalisation in a real-world application. This trained model was then deployed as a web service that is consumed in real-time detection and prevention in web applications. The evaluation performance metrics also include the following statistical measures: accuracy, precision, recall and F1 Score.

5.4.1 Accuracy

The accuracy is the statistical measure in a binary classification of the ratio of correctly predicted observations [199] which can be relied on when labelled data items are symmetrical (equal ratio of positively labelled class data items to negative) as in the case presented in this thesis in section 5.3.2. We have *20% (145042)* of the entire data items as testing data items unseen by the classifier which is the subject of this empirical evaluation. The remaining data set used to train the classifier has equal labelled classes distribution of both SQLIA positive or negative while the remaining *80% (580164)* was used as the training data.

At the best performing threshold value of *0.5*, we achieved an accuracy value of *0.986* which is calculated by Equation 5-1 below with values from Table 5-1. The result of the experiment as detailed below demonstrates a high accuracy of TP and TN with few errors of an FN.

The FP observations are the referral web requests for further verification by the system administrator before fulfilling the request, which in SQLIA detection and prevention is a good thing. These errors of FP and FN can be eliminated if we restrict the application to only detect SQLIA within a defined scope of the RegEx pattern constraint presented in the data pre-processing.

Table 5-1 Positive and negative events at 0.5 thresholds: A table of observations of positive and negative events at the 0.5 thresholds.

Predictions	**Values**
True Positives (TP)	72359
False Negatives (FN)	162
False Positives (FP)	1923
True Negatives (TN)	70598

$$\text{Accuracy}=\frac{\text{TP (72359)+ TN (70579)}}{\text{TP (72359)+FP (1923)+FN (162)+ TN (70598)}}=0.986$$

Equation 5-1

5.4.2 Precision

The precision is the statistical measure of the ratio of the correctly predicted positive observations that is calculated using the relevant values in Table 5-1 above as presented in Equation 5-2 with a precision value of *0.974.* We achieved *72539* events correctly predicted true positives on the backdrop of *1923* false positives. In our experiment with equal distribution labelled data items, the precision value of *0.974* achieved gives the actual precision performance of the trained, supervised learning model to be deployed as a web service.

$$\text{Precision}=\frac{\text{TP (72359)}}{\text{TP (72359)+FP (1923)}}=0.974$$

Equation 5-2

5.4.3 Recall

The recall (also referred as TPR) is a statistical measure of the ratio of true positive against all positive events, and so measures the proportion of correctly predicted positive events. The recall is calculated with true positive over positive events which include TP, and FN (also interpreted as positive events).

In this experiment, we had total positive events of *72521* comprising of TP and FN of which *72359* were correctly predicted as TP. We achieved a recall value of *0.997* which is calculated by Equation 5-3 below.

$$\text{Recall}=\frac{\text{TP (72359)}}{\text{TP (72359)+FN (162)}}=0.997$$

Equation 5-3

5.4.4 F1 Score

The F1 Score provides a statistical measure of the accuracy to binary classification. It is a measure of both recall and precision which is calculated as shown in Equation 5-4 below with the F1 Score value of *0.985*. We have an equal distribution of class in the experiment presented in this thesis which is a statistical measure of the F1 Score that helps us determine how well the trained, supervised learning model is performing.

$$\text{F1 Score}=2\text{ x }\frac{\text{Recall (0.997) x Precision(0.974)}}{\text{Recall (0.997)+Precision(0.974)}}=0.985$$

Equation 5-4

The SVM binary classifier evaluation scores are the observations of the *145042* score probabilities of the testing set. These observations are used to calculate the performance metrics between *0.1* and *1.0* score bins as shown in Table 5-2. It uses the threshold value to align the binary classifier to predict values less than *0.5* as less likely to be SQLIA negative (*0*) and a threshold value greater than *0.5* as most likely to be predicted as SQLIA positive (*1*).

Table 5-2 Evaluations scores of TP, TN, FP and FN: A table containing binary SVM classifier score probabilities of evaluation scores of TP, TN, FP and FN at the various thresholds, including FPR and TPR used in plotting the ROC curve.

Threshold	TP	FN	FP	TN	PO	NE	TE	FPR	TPR
1	0	72521	0	72521	72521	72521	145042	0	0
0.9	56446	16075	1505	71016	72521	72521	145042	0.02075261	0.778340067
0.8	63346	9175	1692	70829	72521	72521	145042	0.023331173	0.873484922
0.7	67886	4635	1806	70715	72521	72521	145042	0.024903132	0.936087478
0.6	71088	1433	1887	70634	72521	72521	145042	0.026020049	0.980240206
0.5	72359	162	1923	70598	72521	72521	145042	0.026516457	0.997766164
0.4	72324	197	1921	70600	72521	72521	145042	0.026488879	0.997283545
0.3	72340	181	2179	70342	72521	72521	145042	0.030046469	0.997504171
0.2	72355	166	6175	66346	72521	72521	145042	0.08514775	0.997711008
0.1	72376	145	13108	59413	72521	72521	145042	0.180747646	0.998000579
0	72521	0	72521	0	72521	72521	145042	1	1

The Table 5-3 below details the formula of how the recall, accuracy, precision and F1 Score are calculated at each score bin between *0.1* and *1.0.* This performance metric is to evaluate how well the classifier is performing with the provided testing data *(20%)* that is unseen by the SVM binary classifier against the training data *(80%)* used to train the classifier.

Table 5-3 Formula for calculating the performance metrics: The formula for calculating performance metrics across the different score bins.

Terminology	Formula	Values	Performance metrics	
True Positive (TP)	-	72359	Accuracy(A) = (TP+TN)/TE	0.986
False Negative (FN)	-	162		
False Positive (FP)	-	1923		
True Negative (TN)	-	70598	Precision (P)=TP/(PO)	0.974
Positive events (PE)	TP+FN	72521		
Negative events (NE)	FP+TN	72521	Recall (R)= TP/PE	0.997
+ observations (PO)	TP+FP	74282		
- observations (NO)	FN+TN	70760	F1Score= 2 x (R x P)/(R+P)	0.985
Total events (TE)	PO+ NO	145042		

The Table 5-4 displays a snippet at *0.5* threshold with excellent performance metrics (accuracy: 0.986, precision: 0.974, recall: 0.997 and F1 Score: 0.985). Table 5-3 above provides the formula for calculating the performance metrics that are repeated across the score bins or thresholds between *0.0* and *1.0*. The ROC curve is plotted with a ratio of FPR on the x-axis (specificity) against recall or TPR on the y-axis (sensitivity).

The ROC curve provides a graph performance measure of a trained model. A higher value on the x-axis, implies bad performance, while on the y-axis a higher value indicates an excellent performance. Figure 5-9 illustrates a graph of ROC curve that presents an AUC statistical measure, in which a shift of the curve towards the upper value of y-axis greater than *0.9* indicates an excellent model [50], [51]. We observed in the experiment presented in this thesis an AUC value of *0.986* achieved at a threshold value of *0.5* which indicates excellent performance in SQLIA prediction. We further the evaluation in the next session with cross-validation to gauge the trained model biases in the real-world unknown data generalisation.

Table 5-4 The metrics values across the different thresholds: A table containing binary classifier score probabilities of collated evaluation scores of performance metrics calculation across the various thresholds that are used to plot the ROC curve in Figure 5-9.

Thresholds/ Score bins	Recall	Accuracy	Precision	FI Score
1	0	0.5	0	0
0.9	0.778340067	0.878793729	0.9740298	0.865258446
0.8	0.873484922	0.925076874	0.9739844	0.92100117
0.7	0.936087478	0.955592173	0.974086	0.954708782
0.6	0.980240206	0.977110078	0.9741418	0.977181503
0.5	0.997766164	0.985624853	0.9741122	0.985797293
0.4	0.997283545	0.985397333	0.9741262	0.985568865
0.3	0.997504171	0.983728851	0.9707591	0.983949946
0.2	0.997711008	0.956281629	0.9213676	0.958020801
0.1	0.998000579	0.908626467	0.8466614	0.916122908
0	1	0.5	0.5	0.666666667
	TPR=TP/PE	**A=(TP+TN)/TE**	**P=TP/PO**	**F1=2x(RxP)/(R+P)**

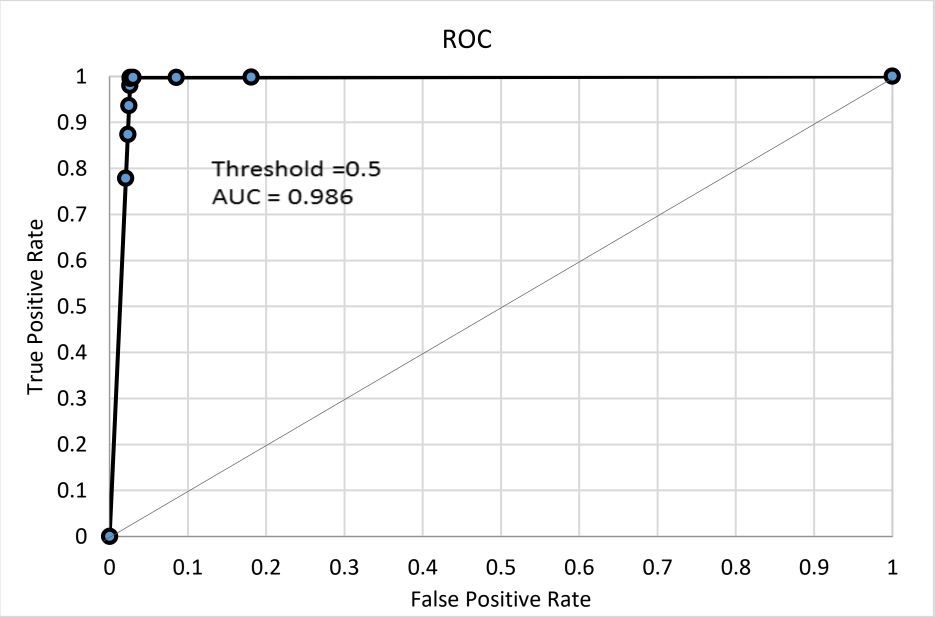


Figure 5-9 ROC curve of a trained model: A figure of a ROC curve of a trained model indicating an excellent model with an AUC value of 0.986.

5.4.5 Cross-validation

In discussing the cross-validation presented in this chapter, we directly picked the best-trained model for the cross-validation. For discussion on introduction to cross-validation refers to section 4.5.4. In the preceding sections, we derived a pattern-driven data set containing *725206* vectors to train a binary classifier with a labelled class of value *1* for SQLIA positive while *0* for SQLIA negative as presented in 5.3.1 above. The hashed data set vector matrices of *725206* observations is used in the cross-validation as illustrated in Figure 5-10.

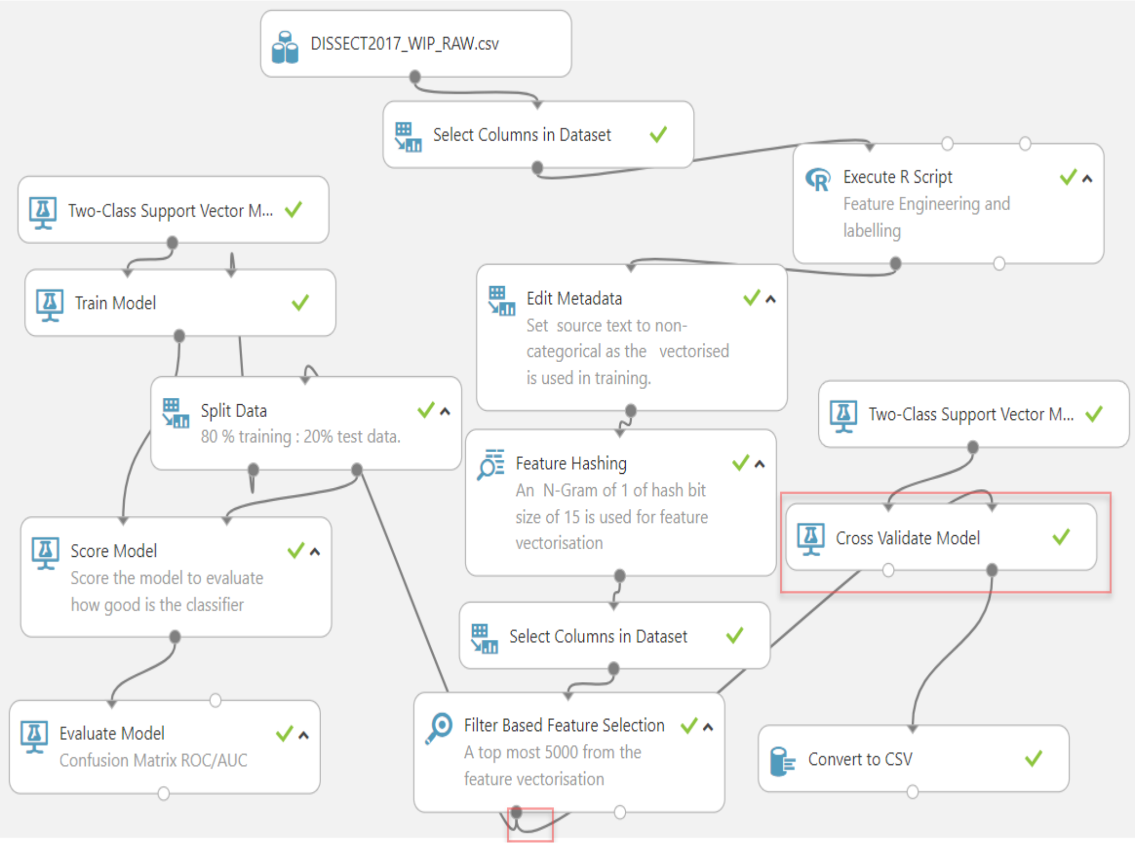

DISSECT2017_WIP_RAW.csv
Select Columns in Dataset
Execute R Script
Feature Engineering and labelling
Two-Class Support Vector M...
Train Model
Edit Metadata
Set source text to non-categorical as the vectorised is used in training.
Split Data
80 % training : 20% test data.
Two-Class Support Vector M...
Feature Hashing
An N-Gram of 1 of hash bit size of 15 is used for feature vectorisation
Score Model
Score the model to evaluate how good is the classifier
Cross Validate Model
Select Columns in Dataset
Evaluate Model
Confusion Matrix ROC/AUC
Filter Based Feature Selection
A top most 5000 from the feature vectorisation
Convert to CSV

Figure 5-10 A cross-validation experiment: A screen capture of the cross-validation to the MAML studio predictive experiment presented in Figure 5-7 above.

We presented a TC SVM trained model using training to testing split data ratio of *80%* to *20%* resulted overall in an excellent trained model as graded in Table 4-11 above (showing AUC grading). We achieved an AUC performance of value of *0.986* with low prediction errors which are the subject of the cross-validation presented below to gauge the performance metrics of the unknown data (akin to testing data in a real-world scenario) to the classifier with reduced biases.

Table 5-5 presents the cross-validation results of the data set matrices partitioned into ten folds with a low standard deviation of accuracy highlighted to indicate a trained model capable of generalisation of unknown independent data. The table also contains a high-performance metrics for Precision, Recall, F-Score and AUC, including a low average log loss values to indicate of a trained model with a low penalty for wrong score probabilities. In the cross-validation results, we observed a low standard deviation in accuracy across the ten folds to have observed an excellent trained model. This result indicates a model with a reduced bias in data set features variation and distribution that can generalise independent data set in a real-world scenario.

Equation 5-5 presents the equation for calculating the mean μ, where μ is the mean, x_i are the accuracy values across the ten folds (*0.9849, 0.9855, 0.9842, 0.9851, 0.9849, 0.9846, 0.9851, 0.9850, 0.9855, 0.9847), n=10* and therefore μ *=9.850/10 ≈ 0.985.* Equation 5-6 presents the equation for calculating the variance or standard deviation σ, where σ is the variance calculated to be *0.0004*. We achieved a low variance of accuracy value of *0.0004* which indicates a less biased trained model capable of generalisation of unknown independent data in the real-world application.

$$\mu = \frac{\Sigma x_i}{n}$$

Equation 5-5

$$\sigma = \sqrt{\frac{\Sigma(x_i - \mu)^2}{n}}$$

Equation 5-6

Table 5-5 A table of evaluation results by folds: A table containing evaluation results of the data set matrices partitioned into *10* folds with a low standard deviation of accuracy highlighted to indicate of a trained model capable of generalisation of unknown independent data.

Fold Number	Number of examples in fold	Model	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss
0	72520	SVM (Pegasos-Linear)	0.9849	0.9736	0.9969	0.9851	0.985	0.0729
1	72521	SVM (Pegasos-Linear)	0.9855	0.9747	0.9969	0.9857	0.9856	0.0699
2	72521	SVM (Pegasos-Linear)	0.9842	0.973	0.9962	0.9845	0.9847	0.0768
3	72521	SVM (Pegasos-Linear)	0.9851	0.9743	0.9966	0.9853	0.9854	0.0722
4	72520	SVM (Pegasos-Linear)	0.9849	0.9738	0.9966	0.9851	0.9851	0.073
5	72520	SVM (Pegasos-Linear)	0.9846	0.9732	0.9965	0.9847	0.9847	0.0743
6	72521	SVM (Pegasos-Linear)	0.9851	0.9742	0.9967	0.9853	0.9853	0.0714
7	72521	SVM (Pegasos-Linear)	0.985	0.9745	0.9961	0.9852	0.9853	0.0721
8	72520	SVM (Pegasos-Linear)	0.9855	0.9751	0.9965	0.9856	0.9856	0.0694
9	72521	SVM (Pegasos-Linear)	0.9847	0.9735	0.9962	0.9847	0.985	0.0726
Mean	725206	SVM (Pegasos-Linear)	0.985	0.974	0.9965	0.9851	0.9852	0.0724
Standard Deviation	725206	SVM (Pegasos-Linear)	0.0004	0.0007	0.0003	0.0004	0.0003	0.0021

5.5 Publishing and consuming the prediction analytics web service

Figure 5-11 section organisation chart presents the layout of this section with the associated subsections.

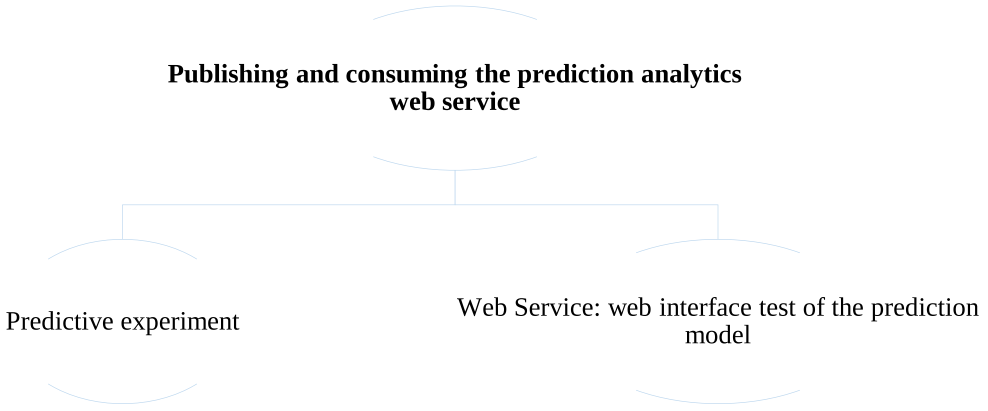

Publishing and consuming the prediction analytics web service
Predictive experiment
Web Service: web interface test of the prediction model

Figure 5-11 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

MAML is a cloud-based platform for testing and deploying ML trained models as a web service that can be consumed in the broad range of front-end web applications. The system requirements regarding RAM and the hard disk are minimal as the one-off workload of training the classifier including retraining is handled in the cloud-based MAML platform.

5.5.1 Predictive experiment

Based on the statistical measure of the performance metrics with AUC value of *0.986,* the trained SQLIA prediction model presented is deemed a model excellently trained from the evaluations to have generated a predictive experiment that is exposed as a web service. The MAML environment equally provides a platform that enables easy deployment of a trained ML model as a web service in ongoing SQLIA prediction [200].

The predictive experiment is a generated copy of the trained classifier detailed in Section 5.3 above. It has a web services input and output that allows an optional module to specify input and output columns attributes. Every intercepted web request at the proxy API are stripped of input data and analysed with the deployed web service used to predict SQLIA. The input data in web requests are the subject of analyses at the predicate's expression detailed in Figure 4-4 above.

To deploy a web service of the trained model, a predictive experiment that is shown in Figure 5-12 below needs to be generated by the utility available within the MAML studio. The solution is scalable, and it is meant to detect and prevent SQLIA in web requests as illustrated in Figure 5-3 above details the architecture of the proposed model.

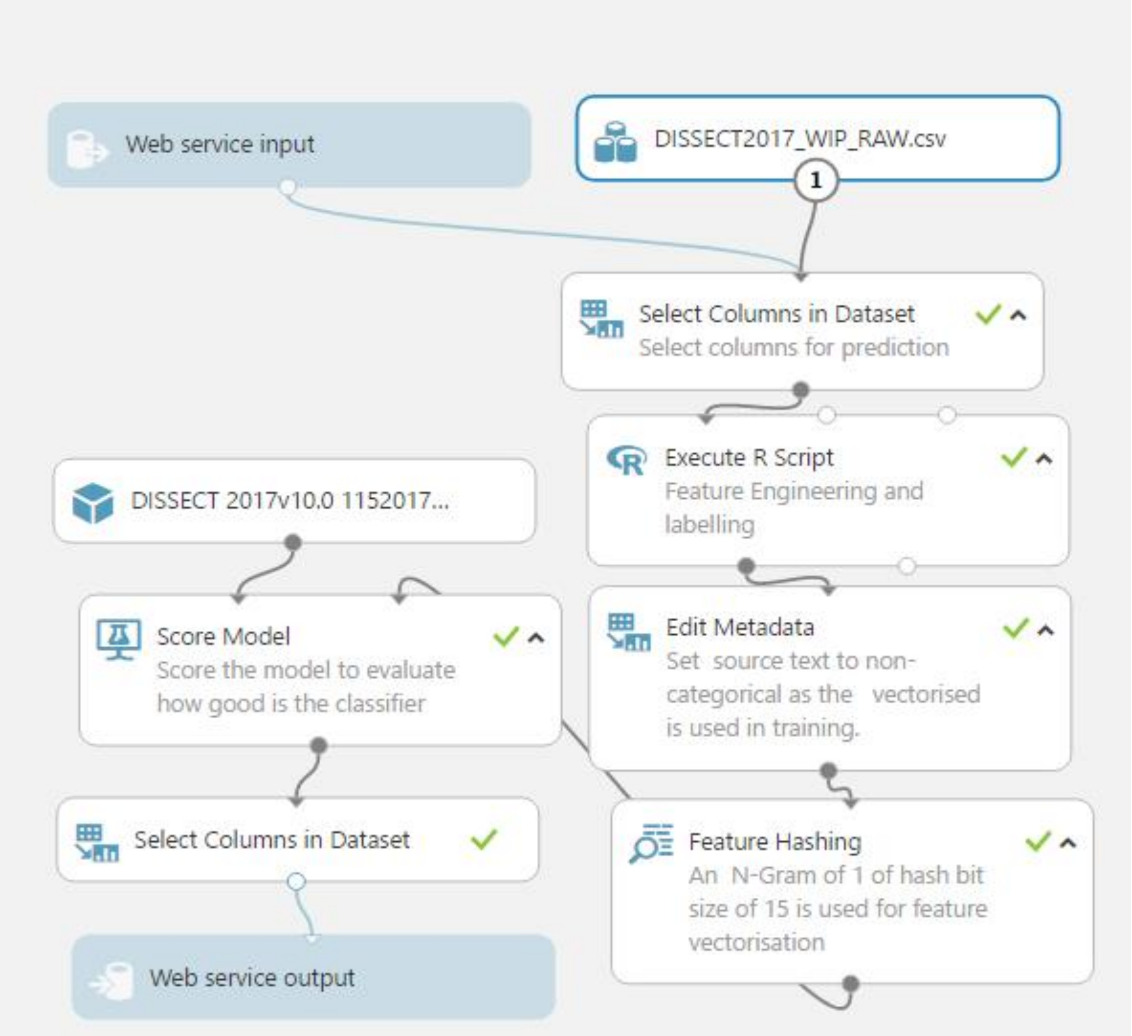
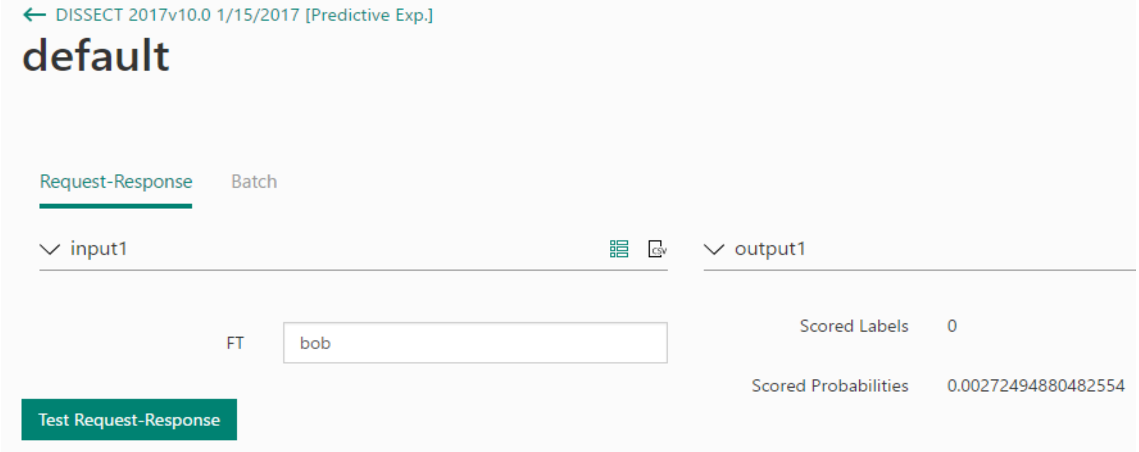

Web service input
DISSECT2017_WIP_RAW.csv
1
Select Columns in Dataset
Select columns for prediction
Execute R Script
Feature Engineering and labelling
DISSECT 2017v10.0 1152017...
Score Model
Score the model to evaluate how good is the classifier
Edit Metadata
Set source text to non-categorical as the vectorised is used in training.
Select Columns in Dataset
Feature Hashing
An N-Gram of 1 of hash bit size of 15 is used for feature vectorisation
Web service output

Figure 5-12 Predictive experiment steps: A figure illustrating predictive experimental steps in creating a web service from the trained, supervised learning model.

5.5.2 Web Service: web interface test of the prediction model

The trained model is exposed as a web service that is called in a custom-built .NET application named NETSQLIA used for ongoing SQLIA detection and prevention. Critical to the deployment in every new web application type, the administrator or system expert needs to feed the data engineering or text pre-processing module with a new rule that matches the patterns present in the new data set to trigger the retraining of the classifier to adapt to the new environment.

This thesis proposal aims to intercept web requests and predict whether it is malicious with the presence of SQLIA or a valid web request. A web application front-end would have a pre-defined expected input pattern, e.g. *bob or bbo* string as a username would be valid input data within the SQLIA hotspot of the predicate' expression after the WHERE clause. Figure 5-13 below is an illustration of a valid web input, which is correctly predicted to be SQLIA true negative with scored label *0* and score probabilities of *0.002725* which indicates the feature input data is SQLIA negative.


← DISSECT 2017v10.0 1/15/2017 [Predictive Exp.]

default

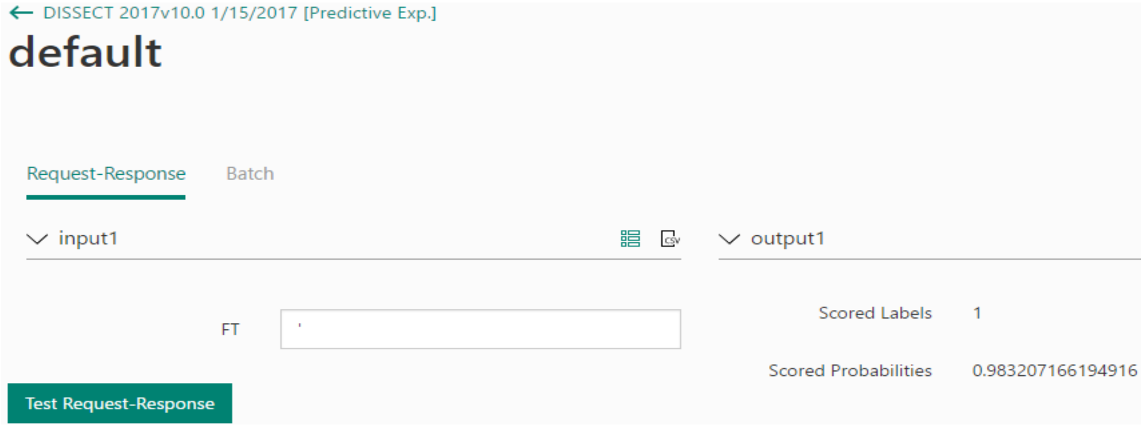
FT: bob

Scored Labels: 0

Scored Probabilities: 0.00272494880482554

Figure 5-13 A web test of valid input data: A figure of a web test of a valid input data using a string example of bob.

In the approach presented in this thesis, we intercept and analyse web requests' input data for SQL keywords and token presence at the hotspot predicate's expression to detect and prevent SQLIA. Figure 5-14 is a true positive test of a single quote (') when present at the predicate's expression location, it demonstrates SQLIA was correctly predicted. We also tested SQL keyword (select) presence at this predicate's expression of SQL language element as shown in Figure 5-15 below confirming a correct prediction of a true positive.


← DISSECT 2017v10.0 1/15/2017 [Predictive Exp.]

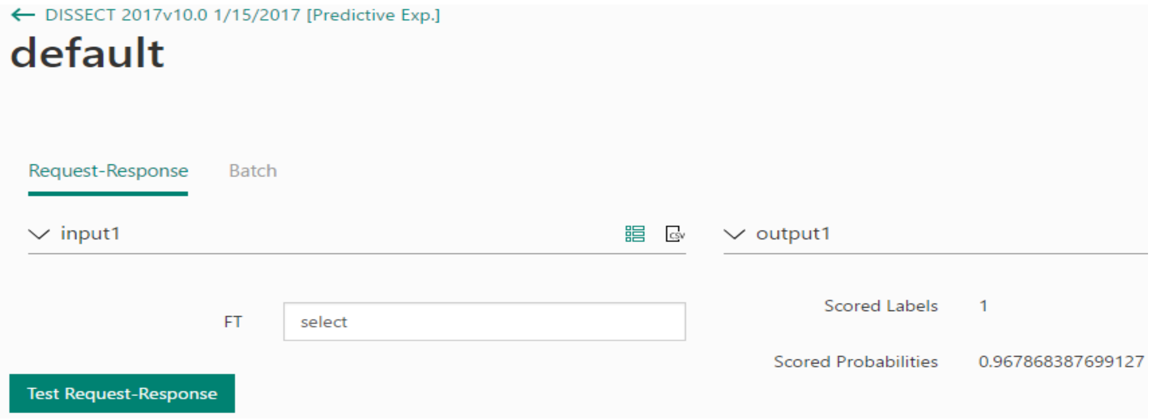
default

FT: '

Scored Labels: 1

Scored Probabilities: 0.983207166194916

Figure 5-14 A test using single quote SQL symbol: A figure of a tautological SQLIA type testing using a single quote SQL symbol that mostly precedes SQLIA.



← DISSECT 2017v10.0 1/15/2017 [Predictive Exp.]

default

FT: select

Scored Labels: 1

Scored Probabilities: 0.967868387699127

Figure 5-15 A malicious input data: A figure of malicious input data, e.g. SQL keywords like SELECT when present at the hotspot in an intercepted web request can infer SQLIA.

5.6 Discussion

Through literature review, we identified a gap in the existing literature as to present in this chapter a proposal of multi-layers approach to SQLIA mitigation that applies ML using a pattern-driven data set in predicting SQLIA in an intercepted web request with a proxy API deployed at cloud SDN. The web service generated from the ML trained model is consumed in the Proxy API. Also, the web client form interface also receives the web service for input validation. The proposal presented in this thesis chapter is a multi-layered approach to intercepting and analysing web request to predict SQLIA in providing SQLIA mitigation. This chapter discussion is the subject of an IEEE paper titled Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention [63]. This paper is used in the comparison table between this proposal presented and existing approaches. The below paragraphs has selected papers on ML related to this discussion, but a much more scope has been examined in Chapter 3.

There are existing proposals to detect and prevent SQLIA; a few mitigate SQLIA by applying SVM machine learning. The few that do employ SVM [71], [72], [74], [128]–[132], [201] are lacking in data engineering (text pre-processing) as their approach is geared towards source code scanning for vulnerabilities in web application source code which is restricted in cloud-hosted applications. There is an element of query string or SQL queries comparison of some sort in these existing approaches. Also, to date, none has discussed using ML to predicting SQLIA in a context of big data, focusing on patterns and text pre-processing on MAML studio. ML cloud-based software and AI platform as services provide large-scale supercomputer employing AI techniques to deliver ready to use solutions lacking in a single server deployment in the intranet or extranet as seen in existing work.

Applying ML requires a data set vector matrix to train a classifier implementing an SVM algorithm [202], [203] to predict SQLIA accurately. There are issues of standardised data set, and the few data set [65], [66] that exist are obsolete, researchers have presented various approaches for extracting data sets with most proposals suffering from limited data engineering. Bockermann et al. [75] proposed tree kernels for analysing SQL statement combined with feature vectorisation of data input to an SVM classifier, but drawbacks in the tree-kernels computational overhead. Choi et al. [72] train an SVM classifier that explores feature vectorisation using N-Grams for query comparison that drawback in accuracy as small data set with few features was used. Kar et al. [137] proposes using a SQL query graph of tokens and centrality of nodes to train an SVM classifier but suffers from complexities. Some of these approaches also fall short in the implementations beyond the CM and ROC curve evaluations and validations.

In focusing on the contributions of this chapter to SQLIA mitigation, we list the following that was recognised in the course of this thesis in the context of emerging computing and big data.

- A pattern-driven data set inferred from a web application type being protected will break the status quo lexical or syntactic query comparison for SQLIA mitigation as seen in existing approaches as data set can now be inferred on the fly.
- Cloud is increasingly used to host web applications and thus there is an astronomical increase in web requests to cloud SDN as to result in big data.
- The recent years have seen an astronomical growth of web requests that drives internet traffic to present the challenge of big data; this is beyond traditional lookup using programming loop constructs to search for strings or SQLI signatures.
- There is restriction to web application source code for vulnerabilities scanning and legacy application sanitisation as web servers are now hosted in the internet cloud that are accessed through cloud SDN endpoints, which also limits existing approaches [71], [72], [74], [128]–[133] that applies SVM learning algorithm [202], [203] in query comparison of some sort.

- There is a need for SQLIA mitigation towards real-world application beyond classical approaches that stop at ROC curve and CM, especially with the recent advancement of cloud-hosted ML platforms.

In comparing the proposal presented in this chapter with existing approaches; this chapter focused on web application type to provide a pattern-driven data set that is hashed to provide vector matrices. The vector matrices are used to train a supervised learning SVM classification algorithm as against existing approaches [71]–[75], [130], [133], [193] employing query comparison of some sort using a data set containing repeating strings of queries. The trained, supervised learning SVM algorithm model was evaluated through statistical measures, ROC curve and cross-validation. The result of the validation and cross-validation presents a model with low variance capable of classifying unknown data in the real-world. Table 5-6 highlights query comparisons, pattern-driven data set, code access and emerging computing cloud SDN as the focus of the SQLIA mitigation which is the subject of this chapter related academic paper [63] that set apart this chapter proposal from existing approaches that apply ML over the years.

Table 5-6 Related work applying ML comparison: A table comparing the existing ML approaches to SQLIA mitigation in the context of source code access, cloud SDN applicable, pattern-driven data set and query comparison with related publication to this chapter highlighted.

Authors	**Algorithms**	**Need access to source code**	**Intranet / intranet domain boundary**	**Emerging computing applicable: cloud SDN applicable**	**Pattern-driven data**	**Query comparison**
Bockermann et al. [75]	SVM	Yes	Yes	No	No	Yes
Komiya [71]	SVM	Yes	Yes	No	No	Yes
Choi et al. [72]	SVM	Yes	Yes	No	No	Yes
Wang and Li [73]	SVM	Yes	Yes		No	Yes
Pinzón et al. [74], [141], [193]	SVM and NN	Yes	Yes	No	No	Yes
Kim and Lee [138]	SVM	Yes	Yes	No	No	Yes
Uwagbole et al. [63]	SVM	No	Yes	Yes	Yes	No

5.7 Conclusion

The contributions discussed in this chapter provide a representative pattern-driven data set that undergo feature hashing to obtain vector matrices to train a supervised learning model implementing SVM algorithm that accurately predicts SQLIA thereby preventing malicious web requests from reaching the target back-end database. It offers a context of SQLIA detection and prevention on big data internet.

Also, this chapter presents a proof of concept of a working prototype using ML algorithms of TC SVM implemented in MAML [204] to predict SQLIA. The methodology presented in the chapter then forms the subject of the empirical statistical measures, ROC curve and cross-validation with a conclusion of a trained model with low biases (capable of classifying unknown data in the real-world).

The scheme presented in this thesis chapter can be used to generate a data set attribute values of any size. We trained a classifier with a data set of 725206 attribute values. The approach presented is driven by the cloud-based MAML studio platform which is capable of even a much larger analytics of data size input of 10 GB with the size of the vector of rows and columns depending on the .NET maximum integer value of 2,147,483,647 limitations.

We suggest in the chapter related publication [63] that patterns exist in any data input to the web application to enable the generation of massive learning data. Further, by applying text pre-processing to such learning data, we can improve the prediction accuracy of the resultant trained, supervised learning model. Based on this web application type, we derived a pattern-driven data set using R string API successfully. The pattern-driven data set is used to train a supervised learning model employing an SVM classification algorithm to predict SQLIA. The evaluated, trained SVM model is exposed as a web service which is then consumed in a web form for input validation, and at the proxy API at the cloud SDN for intercepting web request for analysis. The intercepted request is analysed to predict SQLIA as to accept the benign and reject malicious web request in ongoing SQLIA mitigation.

On account of the discussion in the preceding paragraphs, this chapter answers the research question if a web application type to be protected can produce the artefact for the extraction of a pattern-driven data set? This research question is answered in the following contributions of this chapter.

- We further numeric encoding of pattern-driven data set features presented in Chapter 4 with feature hashing of the pattern-driven data set to obtain vector matrices used to train the classifier with statistical measures evaluations and cross-validation to achieve a result of a trained model capable of classifying unknown data in the real-world.

- We implemented in a successful proof of concept as demonstrated in the chapter of a web application that expects dictionary words as valid input (SQLIA negative) while elements of SQL tokens and SQLIA type substitution at the SQLI hotspots are predicted as SQLIA positive.

We demonstrated in this thesis chapter an applied predictive analytics to SQLIA detection and prevention in a big data context with an excellent result that is empirically evaluated by presenting statistical measure performance metrics and cross-validation. In benchmarking this thesis against existing work, the methodology proposed here is functional in a big data context which is lacking in existing work before now on SQLIA to our knowledge.

6 Pattern-Driven Corpus to Mitigate SQLIA

Figure 6-1 presents the organisation chart layout of this chapter with the sections.

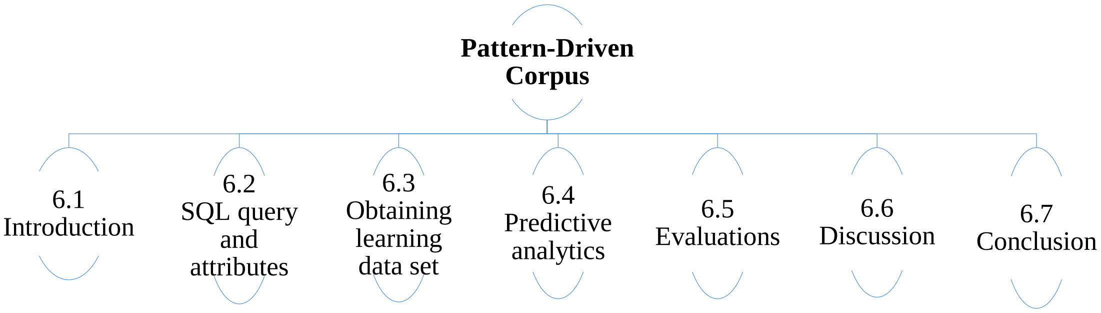


Figure 6-1 The chapter organisation chart: The chapter organisation chart presents the layout of the chapter's sections.

6.1 Introduction

Figure 6-2 presents this section organisation chart with the associated subsections.

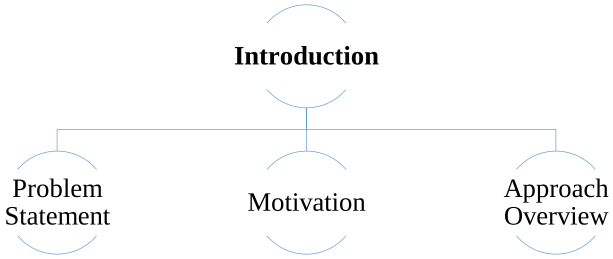


Figure 6-2 The section organisation chart: A figure of the section organisation chart that provides the layout of this section with the associated subsections.

This chapter provides a further work on a pattern-driven data extraction technique using SFA in answering the research question if the pattern-driven data set can be extracted from a web application type to be protected? Emerging computing relies heavily on secure back-end storage for the massive size of big data originating from the Internet of Things (IoT) smart devices to the cloud-hosted web applications. SQLIA remains an intruder's exploit of choice to steal confidential data from the back-end database with damaging data security ramifications. The existing approaches of string lookup techniques were aimed at on-premise web application domain boundary and so are not applicable to roaming cloud-hosted services' edge SDN to application endpoints with massive web request hits.

This chapter presents a further work in a web application type context as the source of the pattern-driven data set to train a supervised learning model. The model is trained with ML algorithms of TC LR and TC SVM that are implemented on MAML studio to mitigate SQLIA. This scheme presented in this thesis then forms the subject of the empirical evaluations.

This chapter starts with the introduction in sections 6.1 and ends with a conclusion in 6.7. Sections 6.3 discusses obtaining learning data and 6.4 details the predictive analytics experiment steps while 6.5 presents evaluations and performance metrics results. Section 6.6 discusses the issues with the existing work to highlight the contributions of this chapter.

6.1.1 Problem statement

In our research question, we ask if a pattern-driven data set can be extracted from a web application type being protected by ML-based SQLIA mitigation? This chapter provides a further work on a pattern-driven data set extraction technique using SFA in answering the research question if the pattern-driven data set can be extracted from a web application type to be protected.

6.1.2 Motivation

There is a continuous upward trend in big internet data with more individuals, governments and businesses adopting and hosting files and applications in the emerging computing cloud hosting environments and associated services.

Availability of historical learning data is a known issue in SQLIA research. The web application type context is so diverse to have a unified data set to train various AI models for SQLIA mitigation that cover every web application domain context.

We leverage the patterns that exist in every input data in both legacy and new web applications to generate related member strings that then forms the historical learning data or data set used to train a supervised learning model in the work presented in this thesis.

6.1.3 Approach overview

There is a need to build a prediction model trained from data set of the desired web application type context to predict SQLIA. We apply NFA implemented in RegEx to define the constraint patterns, and employing SFA with a constraint solver named

Satisfiability Modulo Theories (SMT-Z3) [153], [154], [205] to generate related member strings from the defined RegEx patterns.

In our labelling of the data set, the presence of known attack signature at injection points will contain patterns of SQL tokens which are deemed SQLIA positive. Conversely, the valid web requests (SQLIA negative) would take the form of generating all possible related member strings.

We trained a supervised learning model with this pattern-driven learning data in demonstrating a proof of concept by applying predictive analytics to a test web application expecting a dictionary word list as input data. The pattern-driven learning data are obtained by automata states walk to derive as many member strings in a given pattern of related strings.

An intruder would exploit SQLIA types to carry out the attack at the injection points in any combination. These SQLIA types exploit techniques are: Tautology; Union; Piggyback; Invalid/Logical queries; Time-based; Obfuscation encoding and Stored procedure. These SQLIA types are the source of SQLIA positive labelled feature values in the scheme presented in this thesis.

6.2 SQL hotspot and modelling data set attributes

Figure 6-3 presents this section organisation chart with the associated subsections.

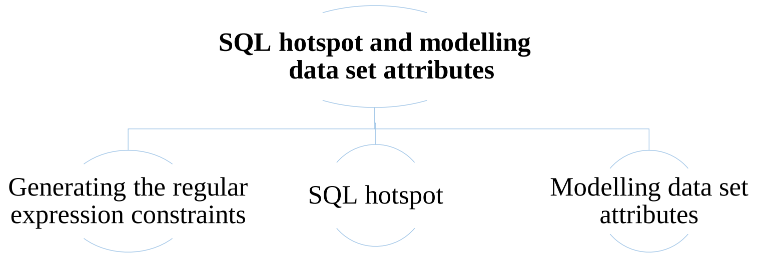

SQL hotspot and modelling data set attributes
Generating the regular expression constraints
SQL hotspot
Modelling data set attributes

Figure 6-3 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

6.2.1 SQL hotspot

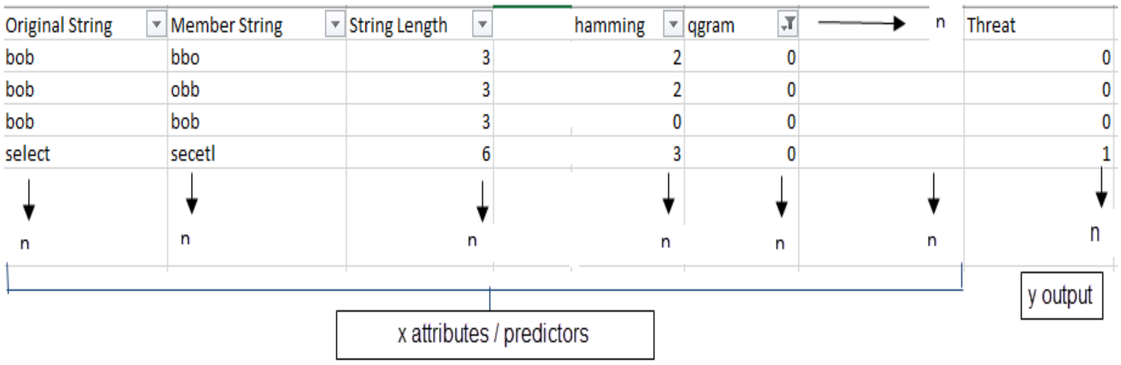
SQL element comprises of tokens which are labelled SQLIA positive in the data set discussed in this thesis. SQL tokens comprise of keywords that include identifiers, operators, literals and punctuation symbols. The SQL syntax language element has the following: SQL Clause (WHERE, SET, UPDATE, etc.); predicate (as in *LoginName = 'bob'*); and, expression (as in *'bob' OR 2=2*).

SQLIA is predicted when a web request is analysed to contain a SQL token and known SQLIA signature at the SQL query injection point. In a SQL query, the WHERE clause predicate and the expression used to control query results are the SQL injection spots. A malicious query string can be passed to a SQL expression in tautological SQLIA type (e.g., *'x'='x' OR 2=2*) to return results from a table far beyond the developer's intention. This SQLIA injection point location as illustrated in Figure 4-4 above has been explored in SQLIA detection and prevention research including SQLProb by Liu et al. [10].

6.2.2 Modelling data set attributes

The versatility of data set actively relies on the scope of the attributes whether it encompasses all elements or features that will be required to predict a labelled output accurately. We inferred various attributes from patterns in an expected web request, SQLIA types and SQL tokens in building a pattern-driven data set to predict whether a web request is a dictionary word list or SQLIA in the test application presented in this thesis. Also, to replicate the approach in any web application domain context would require the same attributes scoping. Figure 6-4 below illustrates attributes scoping deemed needed to accurately predict whether a string extracted from a web request is SQLIA or a valid safe request.

ML requires numeric values translated into vector matrices to train supervised learning algorithms or classifiers. We observed the related member string features and assigned scalar numeric values in building robust attribute *x* values to predict an outcome or output *y* as illustrated in Figure 6 2.



Original String	Member String	String Length	hamming	qgram	→ n	Threat
bob	bbo	3	2	0		0
bob	obb	3	2	0		0
bob	bob	3	0	0		0
select	secetl	6	3	0		1

Figure 6-4 Attributes scoping: A figure of attributes scoping of x to accurately predict labelled output y.

6.3 Obtaining the learning data (data set)

Figure 6-5 presents this section organisation chart with the associated subsections.

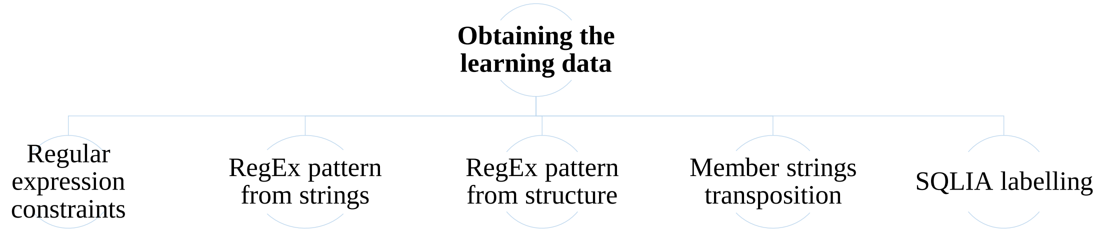

Obtaining the learning data
Regular expression constraints
RegEx pattern from strings
RegEx pattern from structure
Member strings transposition
SQLIA labelling

Figure 6-5 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

We explore automata states walk to generate a data set of patterns of expected input data where none exists to train a supervised learning model. Also, a pattern-driven approach prevents the security implications of making the learning data input to ML being a repository that lays bare the expected input data.

To simplify the derivation techniques of generating the related member strings, we use a string *"bob"* throughout this section. The process can be replicated as many feature values as desired in the intended data set. It also must be pointed out that the context of this thesis is big data scenario with a large volume of feature values of alphanumeric strings in nature meaning that heuristic string signatures lookup will not be scalable. The string *"bob"* is a minute representation for the simplification of the big picture in the significant learning data generation.

Figure 6-6 below illustrates the fundamentals of FSA [56], [151] states walk that forms the building block to our learning data extraction techniques. Automata is a self-propelled abstract computing device in workflow orchestration that automatically follows a predetermined sequence of operations. An automaton is typically represented by a set of five tuples (Q, ∑, q_o, F, ∂) as depicted in Figure 6-6 where the tuples are detailed below:

- Q is finite set of states *{1,2,3}*
- ∑ is finite symbol *{b, o}* termed alphabet
- q_o is the initial state *{1}*
- F is a set of final state *{3}*
- ∂ is the transition function as collated in Table 6-1 below

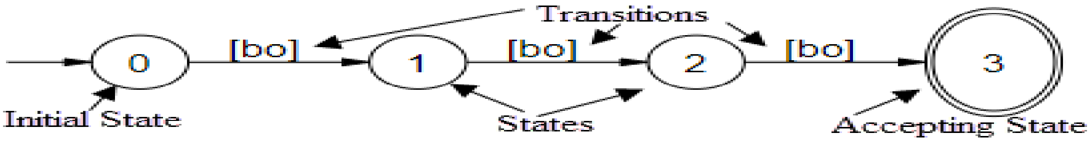


Figure 6-6 FSA states walk to generate member strings: A figure illustrating FSA states walk from the expected input string to generate member strings in creating massive learning data.

Table 6-1 Transitions function (∂) and states walk interpretation: A table illustrating transitions function (∂) and states walks interpretation to derive related member strings.

	Transitions			Accepted member strings
	0,1	***1,2***	***2,3***	
Alphabet *∑= {b, o}*	b	o	b	bob
	b	o	o	boo
	o	o	b	oob
	o	o	o	ooo
	o	b	o	obo
	b	b	b	bbb
	b	b	o	bbo
	o	b	b	obb

We explore automata states walk collation to generate all possible accepted member strings shown in Table 6-1. The manual collation of automata states walks is automated by employing Rex [80] utility by Veanes et al. [153], [205]. Rex is an implementation of SFA which uses Z3 [153] constraint solver [154] to generate as many related member strings as possible from a defined RegEx constraint patterns. In Rex, the RegEx implements an NFA with an epsilon move and a further conversion of NFA to Deterministic Finite Automata (DFA) for optimisation in determining states as against probing possible states paths as in NFA. Below describes the input file passed to the Rex command to generate related member strings.

The RegEx pattern input file to Rex command line is written as Rex /r: InputFile.csv */k: 8* where r and k are the source input file and the size (number of iterations) command options to generate member strings respectively. The input file would contain patterns around the original strings and size where these attributes exist, e.g. ^ *[bob]{3}$* which is interpreted as to derive related member strings of length *3*. Alternatively, where there are no precursor strings, RegEx patterns are inferred from the structure and scope of the expected input data, e.g. *(^[b](?:[a-z]{2})$)* which denote to derive a related member strings of length *3*, starting with letter *b* followed with two more letters from *a* to *z*. The data set procurement technique employing automata states walks to generate related member strings is described in the subsequent sections below.

6.3.1 Generating the regular expression constraints

We apply RegEx [206], [207] to produce constraints of patterns that exist in expected input data. These RegEx constraints of the patterns are for two purposes: use in related member string generation detailed below; and also, as a part of the feature pre-processing or data engineering.

Take an example of a web application presented in this thesis that expects a unigram (single word or string) of an English dictionary word list; we use RegEx to define pattern constraints and employing Rex utility tool to derive as many related member strings from the RegEx pattern constraints. We use a unigram (analysing strings word by word) of English dictionary word list as expected input data in a test web application to demonstrate a pattern-driven data set can be utilised in any web application domain context to train a supervised learning model to predict SQLIA accurately.

This thesis proposal provides the ability to generate a pattern-driven data set based on web application types. This proposal of ability to generate a pattern-driven data set on the fly, which removes the reliance on antiquated data set for the training of ML classifier. These obsolete purpose-built data sets [65], [66] do not account for the various data sets that are relevant to train ML model for the diverse web applications requiring SQLIA mitigation.

To illustrate in simplicity the internal workings of the pattern-driven data set generation, we take a simple RegEx pattern for a string to show how massive quantities of learning data can be generated as shown in Table 6-2. For example, we provided an input of RegEx pattern to a Rex utility tool as presented in Table 6-2. The Regex pattern constraint defines a string that starts with *b* followed by a combination of letters from a group of *a* to *z* but restricted to two letters which result in the derivation of member strings of $b \ldots b_n$ as shown in Table 6-3.

Figure 6-7 below illustrates the Rex SFA generated UML state machine, an indication of the internal workings of how the possible member's strings are generated. Rex command tool (rx.exe) provides an option to accept as many RegEx constraints as possible to generate as many member strings to form extensive learning data. Figure 6-8 shows the Rex utility command line with the RegEx pattern *(^[b](?:[a-z]{2})$)* and the following options: /e for character encoding and /k for the number of related member string to generate.

Table 6-2 A RegEx for a simple string constraint: A table containing an example of RegEx constraints that is passed to an SFA to derive related member strings.

RegEx Constraint
^[b](?:[a-z]{2})$

Table 6-3 An example of member strings from the defined pattern: A table containing possible string members from a defined RegEx pattern [b](?:[a-z]{2}) for ten instances using Rex utility.

Derived Member strings
"bhz"
"bwc"
"bdy"
"bsw"
"bsj"
"bwp"
"blm"
"bzc"
"bam"
"bby"

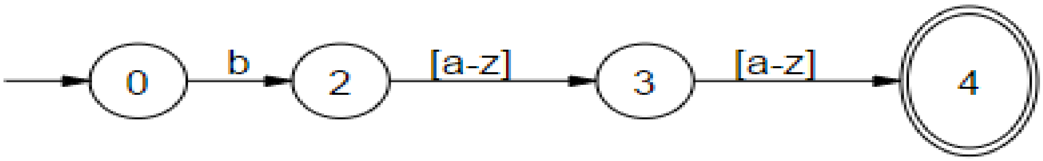
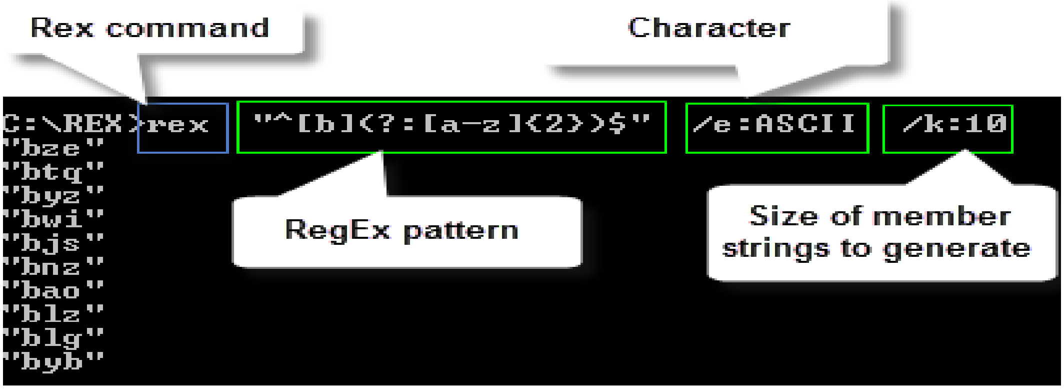


Figure 6-7 UML of generated constraint from Table 6-2: A figure of UML generated constraint from Table 6-2 using Rex tool.



```
C:\REX>rex "^[b](?:[a-z]{2})$" /e:ASCII /k:10
"bze"
"btq"
"byz"
"bwi"
"bjs"
"bnz"
"bao"
"blz"
"blg"
"byb"
```

Figure 6-8 Running Rex command utility against a constraint: A figure demonstrating running Rex command against a defined RegEx constraint to generate the possible related member strings as shown in Table 6-3.

6.3.2 RegEx pattern from strings and size of dictionary word list

A string *S* is a finite combination of symbols *{b, o}* that are extracted from the alphabet denoted by Σ. Then *S* = *“bob”* is expressed as alphabet Σ = *{b, o}* where *b* and *o* are the symbols. Therefore, S is accepted by an NFA or a DFA (Q, Σ, q_o, F, ∂) when $S = \partial^* (q_o, S) \in F$.

The Language L is a set of strings with a defined length denoted by |S|. In this example, the string *"bob"* has length three represented as $|S| = 3$. Applying Kleene Closure or Plus [208] denoted by Σ^+ then, $\Sigma^+ = \Sigma^* - \varepsilon = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3 \cup \ldots_n$, where $\Sigma^* - \varepsilon$ is the Kleene Star (Σ^*) minus epsilon or empty strings (ε), $\Sigma_{1 \ldots n}$ are finite sets of possible related member strings with length that can be generated. If alphabet $\Sigma = \{b,o\}$, $|S| = 3$, then L is accepted by NFA or DFA when $\{ S \mid S \in \Sigma^+$ and $\partial^* (q_o, S) \in F \}$, Therefore the related member strings derived is expressed as $\Sigma^+ = \{bob, boo, oob, ooo, obo, bbb, bbo, obb\}$ as shown in Figure 6-6 above collated in Table 6-1 above.

6.3.3 RegEx pattern from a given string structure and size

We apply RegEx to produce constraints of patterns that exist in expected input data passed to Rex utility to derive related member strings. The RegEx pattern *(^[b](?:[a-z]{2})$)* accepts strings that start with a symbol *b* with any combinations of symbols *[a - z]* where string length $|S| = 2$, then the total string length is $|S| = 3$. Therefore, $\Sigma^+ = \{bhz, bwc, bdy, bsj, blm, bzc, bam, bby\}$ are accepted member strings as shown in Figure 6-9 below illustrating automata states walk.

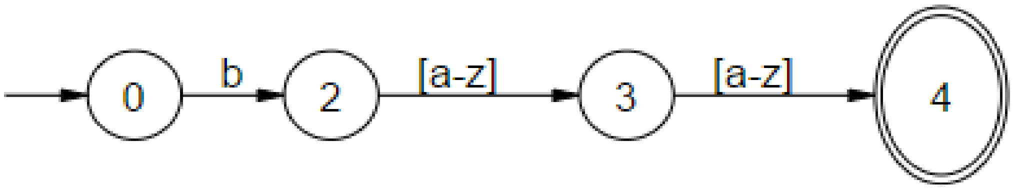
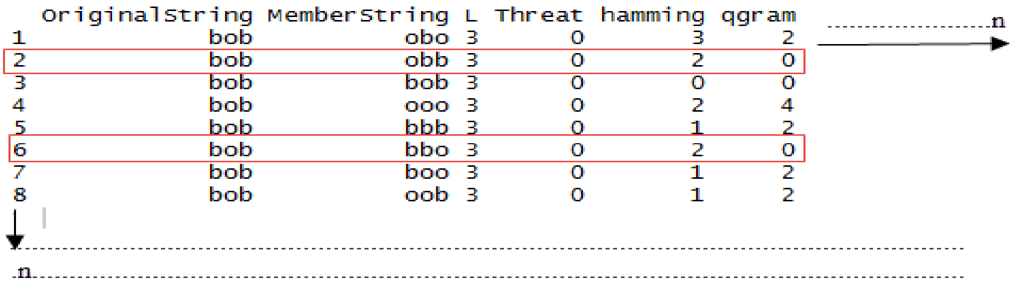


Figure 6-9 A UML of states walk from RegEx pattern: A UML of states walks from a given RegEx pattern to generate possible member related strings from a given string.

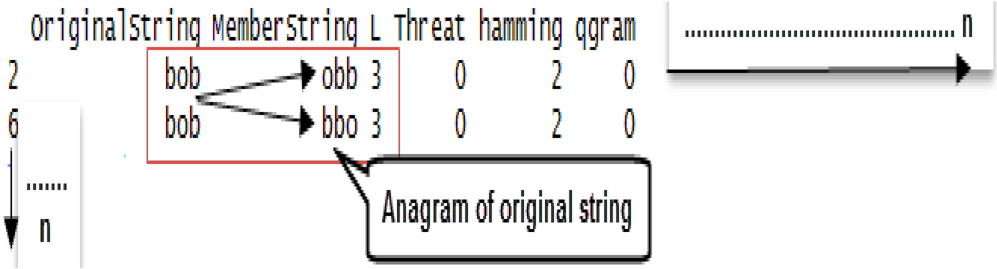
6.3.4 Member strings transposition

We further measure the string distance to compare the pattern of the original string with related derived member strings by string transposition as to filter the anagram patterns of the generated related member strings. We explore R stringdist (string matching package in R) [182] in the member string transposition to improve the performance of the data set in a binary classifier including an intruder attempt to circumvent the classifier by transposition (shuffled strings). Figure 6-10 below is a snippet of original string *"bob"* with the derived related member strings distance values. Figure 6-11 below illustrates string distance patterns measures by comparing Hamming *H* and qgrams *Q* to obtain the transposed (anagram) member strings.



```
  OriginalString MemberString L Threat hamming qgram
1            bob          obo 3      0       3     2
2            bob          obb 3      0       2     0
3            bob          bob 3      0       0     0
4            bob          ooo 3      0       2     4
5            bob          bbb 3      0       1     2
6            bob          bbo 3      0       2     0
7            bob          boo 3      0       1     2
8            bob          oob 3      0       1     2
```

Figure 6-10 String distance measures of member strings: A figure illustrating string distance measures between original and derived member strings.



```
  OriginalString MemberString L Threat hamming qgram
2            bob          obb 3      0       2     0
6            bob          bbo 3      0       2     0
```

Figure 6-11 Filtered values of original string transposition: A figure illustrating filtered feature values containing transposition of original string when $Q = 0$ and $H != 0$.

We explore FSA to derive as many features required in the data set, and explore R language string distance analysis to create a dimension of related member strings. The filtering of derived strings is to obtain a pattern-driven data set that contains only related member strings that include shuffled, transposed and original strings. For example, the string *bob* has similarity to *bbo*, *obb*, *etc.* The filtering is based on string distance measurement as illustrated in Figure 6-11 to ensure strings that have the nuance of related member strings are not wrongly labelled in the procedure illustrated in Figure 6-12. This string distance measure approach to remove noise or unrelated member strings is achieved through the use of R stringdist library [182]. We used the hamming distance method of R stringdist to determine the string positions shift or transposition for strings of equal length. The qgram method of R stringdist is used to determine if the derived member strings are shuffled or anagram outcome of the original string. The preceding string distance measures are to measure to establish a related member string in the derived pattern-driven data set used here to train TC LR and TC SVM classifiers.

This thesis proposal provides the ability to generate a pattern-driven data set used to train an ML classifier based on web application types. It is a paradigm shift from SQL queries syntax structure comparisons seen in existing work [71], [72], [134]–[141], [74], [100], [128]–[133] that require source code access to the scheme presented here of an approach of intercepting web request input destined to be substituted into SQL query element predicate's expression (SQLIA hotspot) as to predict SQLIA.

Choi et al. [72] train an SVM classifier using tokenisation by N-Grams of bigrams, trigrams and 4-grams to compare SQLIA query syntax structure between normal and malicious queries. The text classification approach as employed by these authors above by using more than one n-grams to compare the query structure has a limitation as it only detects queries with the exact count of the grams used in tokenisation. Padding of SQL query with more strings or words impact such queries classification outcomes. Whilst historical approaches of static and dynamic queries comparison seemed plausible in the past, the shifting of web applications on-premise domain boundary to the cloud SDN makes approaches that rely on static and dynamic source code scanning of the web application not applicable in emerging computing where access to source code is restricted. The scheme presented here is a paradigm shift from existing proposals to a scheme that infers a pattern-driven data set relevant to the intended web application type being provided ML SQLIA mitigation. We used unigram of N-grams in our tokenisation as we analyse intercepted input data word by word destined to be substituted in SQLIA hotspot. The proxy API used helps intercepts and decrypts the content of web request and as such does not require access to web application source code.

6.3.5 SQLIA labelling

The derived related member string as detailed above is labelled SQLIA negative (0) while the presence of SQL tokens and existing known SQLIA signatures during member strings pre-processing is labelled SQLIA positive (*1*). The binary class of *0* or *1* is to be predicted at SQL query injection points. The learning data labelling routine in R language is illustrated in Figure 6-12 Flowchart below.

Figure 6-12 starts with employing FSA to derive as many strings, patterns of expected input in web requests. The derived strings are filtered to obtain related member strings that are labelled in the data set as SQLIA positive or negative. The derived pattern-driven data set is imported into MAML studio where it undergoes a further text pre-processing. For example, depending on the web application type, certain words are relabelled; if expected input will contain the European Union, the European Union is relabelled SQLIA negative while Union SQL keyword remains labelled SQLIA positive. Making the European Union as stop words (excluded words) or relabelled to SQLIA negative inform the classifier during the training that a combination of the words European Union is SQLIA negative while a single SQL keyword Union without context is SQLIA positive in the classification.

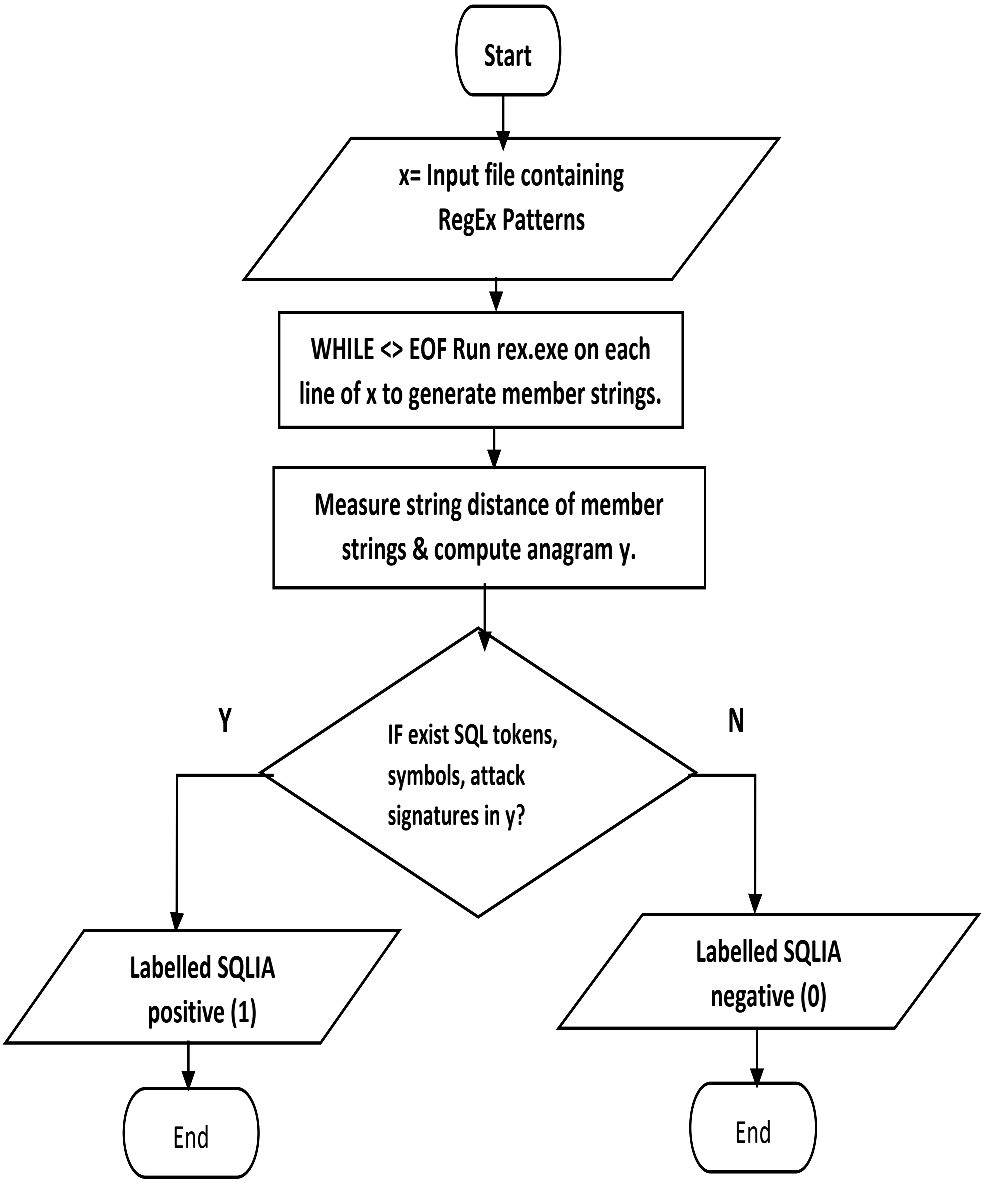

Start
x= Input file containing RegEx Patterns
WHILE <> EOF Run rex.exe on each line of x to generate member strings.
Measure string distance of member strings & compute anagram y.
Y
IF exist SQL tokens, symbols, attack signatures in y?
N
Labelled SQLIA positive (1)
Labelled SQLIA negative (0)
End
End

Figure 6-12 Data set feature values labelling flowchart: A flowchart illustrating data set feature values labelling used to train a supervised learning model.

6.4 Predictive analytics experiment and deployment

Figure 6-13 presents this section organisation chart with the associated subsections.

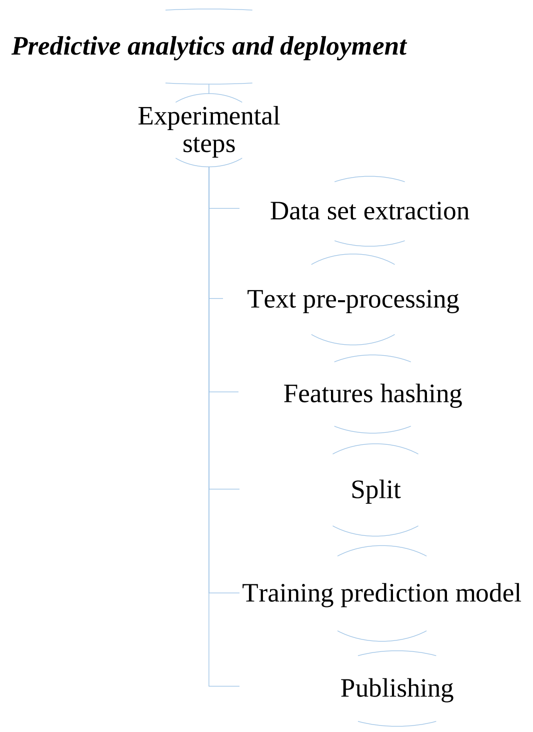


Figure 6-13 The section organisation chart: The section organisation chart provides the layout of this section with the associated subsections.

We apply predictive analytics in this thesis to mitigate SQLIA. The method is built on MAML studio, which is a cloud-based ML platform. The experimental steps are detailed below.

6.4.1 A high-level overview of the experimental steps:

6.4.1.1 Data set extraction

The learning data or feature values used here in the MAML studio to train a supervised learning model contains pattern-driven data set described in detail in section 6.3 above. We obtained *479,000* member strings with additional *862* unique SQL tokens extracted from Microsoft SQL reserved keywords [179].

6.4.1.2 Text pre-processing

This stage involves R Scripting that incorporates all the defined RegEx constraints detailed in 6.3 above to parsed learning data. The feature values are parsed for patterns, duplicates, normalised to lower cases and the removal of the missing attribute values which results in the pruning of the feature values to *362,603*. Also, depending on the web application type, certain words are relabelled, e.g. if expected input will contain the European Union, the European Union is relabelled to SQLIA negative while Union SQL keyword remains labelled SQLIA positive. The data set is sampled to provide an even distribution of the feature values. The imbalanced feature values of majority labelled SQLIA negatives over positives were corrected with SMOTE [187]. The entire feature values of *725206* are split equally. The SMOTE improves the accuracy and F1 score statistical measures in evenly distributed feature values.

6.4.1.3 Features hashing to the matrix

The hashing is to transform the data set feature values into binary vector matrices of 2^{15} *(32,768)* columns required for training a classifier in ML by setting the hashing bit size to 1 (unigrams of N-grams) where $N = 1$. The hashing procedure creates a dimensional input of the matrix with a faster lookup of feature weights by substituting the string comparison with the hash value comparison. We use a unigram hashing of strings into the binary matrix as the intention is to analyse the intercepted strings as a unit at the proxy. We observed analysing a string as a unigram offers a better prediction of TP and TN than examining a phrase of a group of strings together (*N-Grams* > *1*).

6.4.1.4 *Split of vector matrices between training and testing data*

We divide the matrix values of the hashed features into a ratio of *80:20* (training to testing data respectively) of which *80%* forms the training data input to the classifier while *20%* as test data for evaluations. We further optimised the classifier in the MAML studio with Tune Model Hyperparameters (TMH) module [209] to improve TP and TN predictions, but with low FN results of *162* achieved in TC SVM as shown in Table 6-4 below.

6.4.1.5 Training the prediction model

Both TC LR and TC SVM algorithms employ linear kernel which offers a binary prediction at the proxy of a linear separation between SQLIA positive and negative presence in a web request. This linearity of the classes which can be demarcated in a straight line makes algorithms (classifiers) using linear kernel a preferred choice in binary classification.

Also, the two classifiers show good accuracy and fast training times in performance metrics. TC SVM has the advantage of being scalable with significant features set, such as vectorisation (hashing), as used in this experiment to generate higher dimensionality of hashed columns. TC LR and TC SVM algorithms are trained with the training data of the partitioned matrix values. We achieved in the trained model AUC values of *0.984* and *0.986* for TC LR and TC SVM respectively in the ROC curve, but with TC SVM having a fewer FP and FN.

6.4.1.6 Publishing and consuming the prediction web service

The system requirements regarding RAM and the hard disk is minimal as the one-off workload of training the classifier including retraining is handled in the cloud by the MAML platform. The solution is scalable, and it is meant to detect and prevent SQLIA in web requests as illustrated in Figure 5-3 above.

6.5 Evaluation and performance metrics

6.5.1 Binary classification

The statistical measures, ROC curve, AUC and k-fold cross-validations are widely used by data scientists to measure the performance metrics in ML analytics. We extracted an evenly distributed data set of *725206* attribute values or row items by the pre-processing and equal balancing of feature values as described in 6.4.1.2 above. These feature values to be predicted contain an equal representation of labelled strings that are deemed valid web requests and SQLIA threat as described in data extraction labelling in 6.3 above. The string attribute values were hashed to obtain a matrix represented by X_{ij} that refers to the element in rows *i* and columns *j* of the input variables matrix *X*. The output variable to predict *Y* is a single vector representation by Y_i where *i* is the index or row count. Thus, the learning data *l* is represented by Equation 6-1 as a dimension of matrix X_{ij} to predict an output labelled vector Y_i where n is the top row count of index *i*. A function of *X* denoted as *f(X)* to predict a labelled output *Y*. Therefore, *f(X)=Y*, where x is input and y is the predicted output.

$$l = (\mathrm{x_j}^{(\mathrm{i})}, \mathrm{y}^{(\mathrm{i})} \ldots\ldots \mathrm{x_j}^{(\mathrm{i})}, \mathrm{y}^{(\mathrm{i})}) = \mathrm{X_{ij}}, \mathrm{Y_i} \ldots \mathrm{X_{nj}}, \mathrm{Y_n}$$

Equation 6-1

We split the hashed string features matrix and associated predictor variables into a ratio *80:20 %* of *725206* with *580164* matrix values as a training data set while the remaining *145042* as a testing data set. We observed a split ratio of *80:20* (training to testing data) resulted in better performance metrics compared with other split ratios that were tried. The training set is evaluated under various binary classification algorithms or classifiers to select a better performing classifier determined by the AUC performance value. The linear kernel-driven algorithms presented in this thesis are implemented in MAML (Azure ML) studio using the TC LR and TC SVM classifiers.

We observed linear kernel-driven classifiers are better performing in the binary classification of two classes in predicting the discrete value of *0/1* for SQLIA. The AUC provides an overall performance measure between the classifier algorithms as illustrated in Figure 6-14 for which the TC SVM with AUC value of 0.986 was observed to be better performing than the TC LR of AUC value *0.984*.

While the *80%* training sample is the part fed to the classifier to train the prediction model, the remaining *20%* is the testing data vectors, values unseen by the classifier which is the subject of this empirical evaluation presented in this thesis. The testing data input variables of *145042* rows are scored to generate score probabilities of a range $0 \leq x \leq 1$ where x is the input matrix to predict output y. The score probabilities provide a measure of observations that are correctly predicted as TP and TN including the two prediction errors of FP and FN within the range *{0.0,1.0}*. These prediction observations of TP, TN, FP and FN are presented in tables below which are calculated to determine how many of these observations score probability values fall within each score bins set of *{0,1}*.

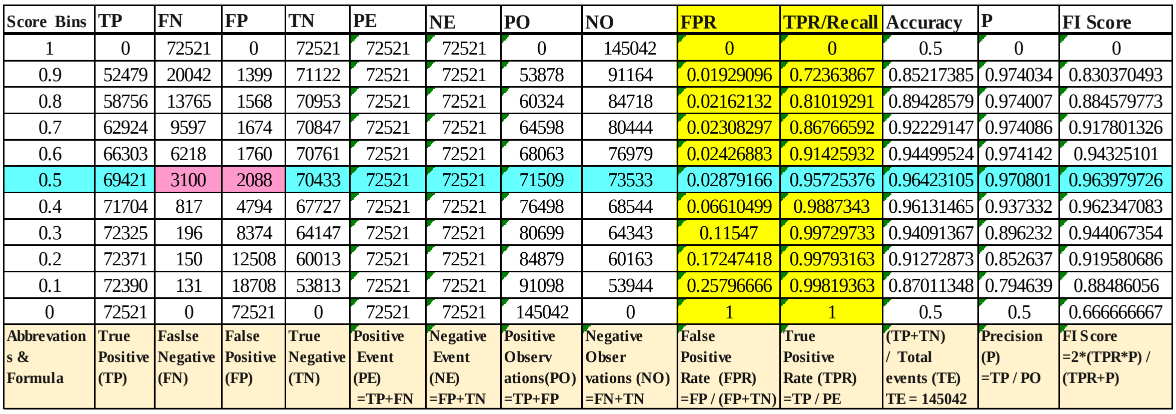
Therefore, the expected output is $y=0$ or SQLIA negative if the function of the predictor variables x is closer to *0* expressed as $f(x) \approx 0$. Conversely, the prediction output is $y = 1$ or SQLIA positive when the function of predictor variables x is closer to *1* denoted as $f(x) \approx 1$. Also, the prediction errors rate is used to gauge the performance of a classifier as illustrated in Table 6-4 with TC SVM achieving low FN (*162*). However, *1923* FP events were observed in TC SVM which indicates such web requests that are falsely alarmed will be referred to the system monitor for a further review.

Table 6-4 A snippet of prediction observations at default threshold: A table containing a snippet of calculated prediction observations at the default threshold of *0.5* that is repeated across *{0,1}*.

Events	Positive	Negative
Positive	TP TC SVM=72359, LR = 69421	FP TC SVM = 1923, LR = 2088
Negative	FN TC SVM= 162, LR = 3100	TN TC SVM=70598, LR = 70433

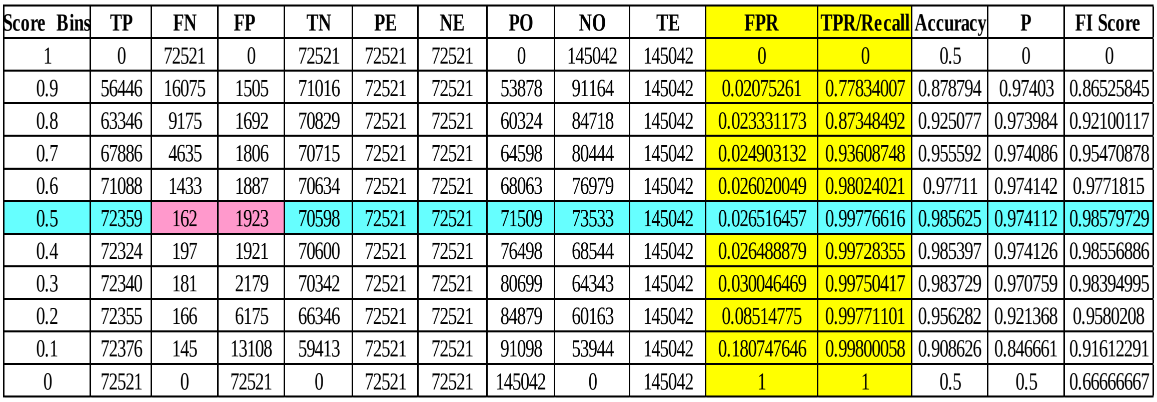
The MAML studio sets by default the cut-off threshold for the prediction of TP and TN including the errors of FP and FN to be *0.5*. This cut-off of *0.5* is a predetermined threshold employed by classification algorithms; it is a trade-off between the cost function of *x* to predict *y* against the performance metrics statistical measures. Therefore *f(x) < 0.5* score probability value is predicted as SQLIA negative *(0)* while *f(x) ≥ 0.5* is predicted as SQLIA positive *(1).* The score probabilities generated from the *145042* rows of testing data are partitioned into score bins of *0.1* increments of the set *{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}*. The calculated predicted observations are aggregated across these score bins using the score probability values as shown in Table 6-5 and Table 6-6.

Table 6-5 TC LR algorithm observations at various cut-off points: A table of observations at various cut-off points between *{0.0, 1.0}* of the TC LR trained model.



Score Bins	TP	FN	FP	TN	PE	NE	PO	NO	FPR	TPR/Recall	Accuracy	P	FI Score
1	0	72521	0	72521	72521	72521	0	145042	0	0	0.5	0	0
0.9	52479	20042	1399	71122	72521	72521	53878	91164	0.01929096	0.72363867	0.85217385	0.974034	0.830370493
0.8	58756	13765	1568	70953	72521	72521	60324	84718	0.02162132	0.81019291	0.89428579	0.974007	0.884579773
0.7	62924	9597	1674	70847	72521	72521	64598	80444	0.02308297	0.86766592	0.92229147	0.974086	0.917801326
0.6	66303	6218	1760	70761	72521	72521	68063	76979	0.02426883	0.91425932	0.94499524	0.974142	0.94325101
0.5	69421	3100	2088	70433	72521	72521	71509	73533	0.02879166	0.95725376	0.96423105	0.970801	0.963979726
0.4	71704	817	4794	67727	72521	72521	76498	68544	0.06610499	0.9887343	0.96131465	0.937332	0.962347083
0.3	72325	196	8374	64147	72521	72521	80699	64343	0.11547	0.99729733	0.94091367	0.896232	0.944067354
0.2	72371	150	12508	60013	72521	72521	84879	60163	0.17247418	0.99793163	0.91272873	0.852637	0.919580686
0.1	72390	131	18708	53813	72521	72521	91098	53944	0.25796666	0.99819363	0.87011348	0.794639	0.88486056
0	72521	0	72521	0	72521	72521	145042	0	1	1	0.5	0.5	0.666666667
Abbrevations & Formula	**True Positive (TP)**	**Faslse Negative (FN)**	**False Positive (FP)**	**True Negative (TN)**	**Positive Event (PE) =TP+FN**	**Negative Event (NE) =FP+TN**	**Positive Observations(PO) =TP+FP**	**Negative Observations (NO) =FN+TN**	**False Positive Rate (FPR) =FP / (FP+TN)**	**True Positive Rate (TPR) =TP / PE**	**(TP+TN) / Total events (TE) TE = 145042**	**Precision (P) =TP / PO**	**FI Score =2*(TPR*P) / (TPR+P)**

Table 6-6 TC SVM algorithm observations at various cut-off: A table of observations at various cut-off points between {0,1} of the TC SVM trained model.



Score Bins	TP	FN	FP	TN	PE	NE	PO	NO	TE	FPR	TPR/Recall	Accuracy	P	FI Score
1	0	72521	0	72521	72521	72521	0	145042	145042	0	0	0.5	0	0
0.9	56446	16075	1505	71016	72521	72521	53878	91164	145042	0.02075261	0.77834007	0.878794	0.97403	0.86525845
0.8	63346	9175	1692	70829	72521	72521	60324	84718	145042	0.023331173	0.87348492	0.925077	0.973984	0.92100117
0.7	67886	4635	1806	70715	72521	72521	64598	80444	145042	0.024903132	0.93608748	0.955592	0.974086	0.95470878
0.6	71088	1433	1887	70634	72521	72521	68063	76979	145042	0.026020049	0.98024021	0.97711	0.974142	0.9771815
0.5	72359	162	1923	70598	72521	72521	71509	73533	145042	0.026516457	0.99776616	0.985625	0.974112	0.98579729
0.4	72324	197	1921	70600	72521	72521	76498	68544	145042	0.026488879	0.99728355	0.985397	0.974126	0.98556886
0.3	72340	181	2179	70342	72521	72521	80699	64343	145042	0.030046469	0.99750417	0.983729	0.970759	0.98394995
0.2	72355	166	6175	66346	72521	72521	84879	60163	145042	0.08514775	0.99771101	0.956282	0.921368	0.9580208
0.1	72376	145	13108	59413	72521	72521	91098	53944	145042	0.180747646	0.99800058	0.908626	0.846661	0.91612291
0	72521	0	72521	0	72521	72521	145042	0	145042	1	1	0.5	0.5	0.66666667

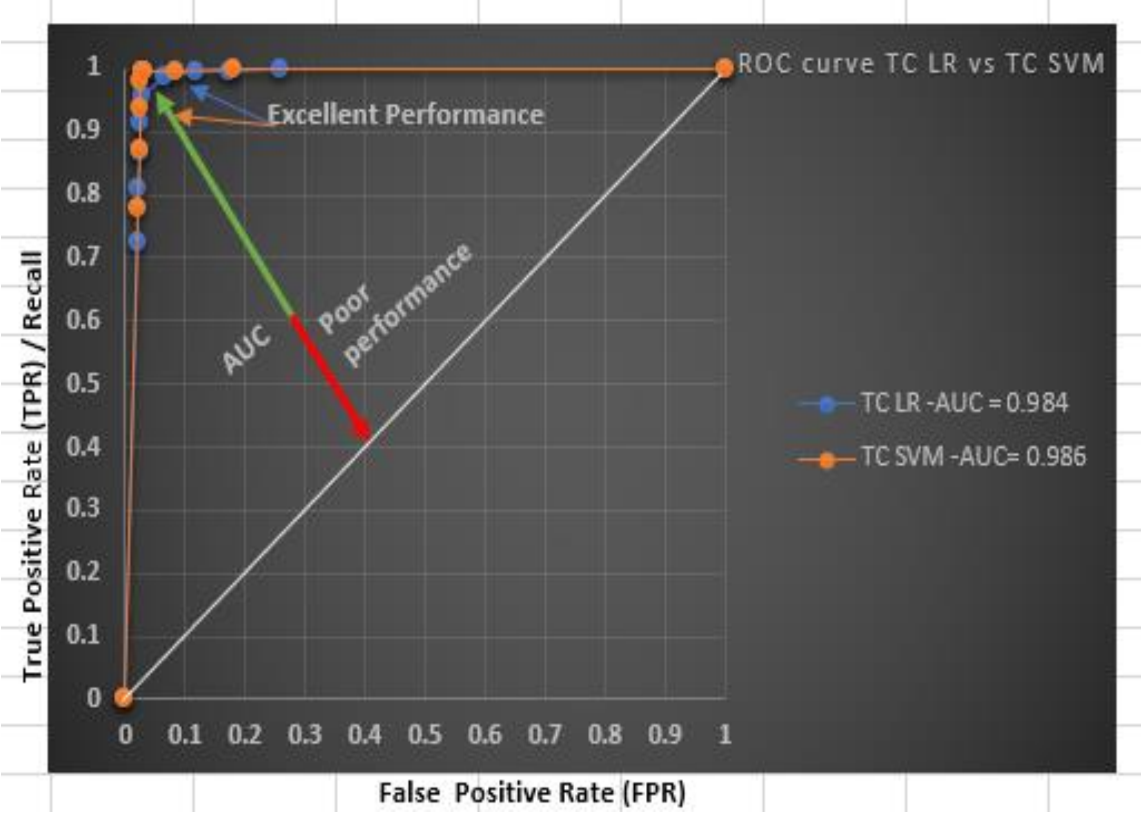


Figure 6-14 A ROC curve of AUC of TC LR vs TC SVM: A figure showing ROC curve comparing AUC of TC LR against TC SVM classifiers of the trained models comparing the performance metrics in AUC.

The statistical measures provide the performance metrics of a trained model. We calculated the statistical measures at the various thresholds *{0.0, 1.0}* as shown in Table 6-5 above where TC LR at default has the following: Accuracy = *0.964,* Precision = *0.971*, Recall = 0.957and F1 Score = 0.964. Table 6-6 is calculated as the preceding, where TC SVM has an improved performance metrics with Accuracy = *0.986,* Precision = *0.974,* Recall = *0.998*, F1 Score = *0.986* and AUC of *0.986* as shown in Figure 6-14 above.

6.5.2 Cross-validation

In discussing the cross-validation in relation to this chapter, we directly picked the best-trained model for cross-validation. For discussion on introduction to cross-validation refers to section 4.5.4. In the preceding sections, we derived a pattern-driven data set containing *725206* vectors to train a binary classifier with a labelled class of value of *1* for SQLIA positive while a value of *0* for SQLIA negative as presented in 6.3 above. The hashed data set vector matrices of *725206* observations are used for the cross-validation. We presented a TC SVM as the excellent trained model using training to testing split data ratio of *80%* to *20%* resulted overall in an excellent trained model as graded in Table 4-11 above (showing AUC grading). We achieved an AUC performance of value of *0.986* with low prediction errors which are the subject of the cross-validation presented below to gauge the performance metrics of the unknown data (akin to testing data in a real-world scenario) to the classifier with reduced biases.

Equation 6-2 presents the equation for calculating the mean μ, where μ is the mean, x_i are the accuracy values across the ten folds (*0.9851, 0.9854, 0.9844, 0.9851, 0.9852, 0.9738, 0.9853, 0.9852, 0.9858, 0.9770*), *n=10* and therefore μ = 9.8323/10 ≈ 0.9832. Equation 6-3 presents the equation for calculating the variance or standard deviation σ, where σ is the variance calculated to be *0.0042*. We achieved a lower variance of accuracy value of *0.0042* which indicates a less biased trained model capable of generalisation of unknown independent data in the real-world application. Table 6-7 presents a table containing evaluation results that are partitioned into ten folds with a low standard deviation (variance) of accuracy highlighted to indicate a trained model capable of generalisation of unknown independent data. The calculation is repeated across other performance metrics of Precision, Recall, F-Score and AUC with similar results, including a low average log loss value across the ten folds to indicate of a trained model with a low penalty for wrong score probabilities as presented in the table below.

$$\mu = \frac{\Sigma x_i}{n}$$ **Equation 6-2**

$$\sigma = \sqrt{\frac{\Sigma(x_i - \mu)^2}{n}}$$ **Equation 6-3**

Table 6-7 A table of evaluation results by folds: A table containing evaluation results of the data set matrices partitioned into ten folds with a low standard deviation of accuracy highlighted to indicate of a trained model capable of generalisation of unknown independent data.

Fold Number	Number of examples in fold	Model	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss
0	72520	SVM (Pegasos-Linear)	0.9851	0.9735	0.9973	0.9853	0.9854	0.092
1	72521	SVM (Pegasos-Linear)	0.9854	0.9747	0.9967	0.9856	0.9863	0.124
2	72521	SVM (Pegasos-Linear)	0.9844	0.973	0.9966	0.9846	0.9849	0.079
3	72521	SVM (Pegasos-Linear)	0.9851	0.9743	0.9966	0.9853	0.9858	0.073
4	72520	SVM (Pegasos-Linear)	0.9852	0.9738	0.9972	0.9853	0.9855	0.105
5	72520	SVM (Pegasos-Linear)	0.9738	0.9732	0.9742	0.9737	0.9851	0.158
6	72521	SVM (Pegasos-Linear)	0.9853	0.9742	0.9972	0.9855	0.9863	0.085
7	72521	SVM (Pegasos-Linear)	0.9852	0.9745	0.9966	0.9854	0.9858	0.084
8	72520	SVM (Pegasos-Linear)	0.9858	0.975	0.9972	0.986	0.9861	0.091
9	72521	SVM (Pegasos-Linear)	0.977	0.9735	0.9803	0.9769	0.9853	0.152
Mean	725206	SVM (Pegasos-Linear)	0.9832	0.974	0.993	0.9834	0.9856	0.104
Standard Deviation	725206	SVM (Pegasos-Linear)	0.0042	0.0007	0.0084	0.0043	0.0005	0.03

6.6 Discussion

SQLIA is still an ongoing issue [21], and WAF is still stealthily bypassed by web traffic containing SQLI [210]. We demonstrated and discussed in the preceding chapter and related published paper [63] that pattern-driven data set can be extracted from a web application type to be protected. We further the work presented in the Chapter 5 of using R string package to derive related member strings with SFA [153], [154], [205] in this chapter to derive related member strings from the defined RegEx patterns to procure a pattern-driven data set to train an SVM classifier in ongoing SQLIA detection and prevention. A pattern-driven data set inferred from a web application type being protected will break the status quo of lexical or syntactic query comparison for SQLIA mitigation as seen in existing work [72], [79], [117], [211], [212] which can be circumvented with more spaces or padding with extra strings.

There is a need for a data set relevant to SQLIA mitigation on real-world web application other than the purpose-built and challenge-driven data set as seen in ECML/PKDD 2007 [65] and HTTP CSIC 2010 [66]. These out of date data sets on SQLI were purpose-built for competition not idea to build mitigation against SQLIA in real-life scenario business application. A pattern-driven data set has a set of items of subsequence and substructures that frequently occur together [162]. We identified through this research that a web application type would have input data of a specific type, size, form, structure, domain type vocabulary and grammar which forms a pattern.

The existing proposals displayed a need for pattern-driven data set as against repeating strings of SQL queries and URL query string which also reflected in the procurement of data set as a test case that was used in these proposals over the years. Also, they were SQLIA mitigation approaches that existed before the emerging computing of cloud-hosted web services where the traditional web server no longer resides on the organisation’s intranet or extranet.

In focusing on the contributions of this chapter, we list the following that was recognised in the course of this thesis in the context of emerging computing and big data as to have achieved the proposal presented in this thesis.

- The web requests that drives internet traffic is now big data; this is beyond traditional lookup using classic programming loop constructs to search for SQLI signatures which are not known to be scalable when compared with ML platforms in emerging computing big data.

- The dependence on static code access scanning for SQLIA mitigation in most existing work as against the need for web requests to cloud-hosted services to be intercepted for analysis.

Table 6-8 Related work applying AI comparison: A table comparing the existing ML approaches to SQLIA mitigation in the context of source code access, cloud SDN applicable, pattern-driven data set and query comparison with related publication to this chapter highlighted.

Authors	**Algorithms**	**Need access to source code**	**Intranet / intranet domain boundary**	**Emerging computing applicable: cloud SDN applicable**	**Pattern-driven data**	**Query comparison**
Bockermann et al. [75]	SVM	Yes	Yes	No	No	Yes
Komiya [71]	SVM	Yes	Yes	No	No	Yes
Choi et al. [72]	SVM	Yes	Yes	No	No	Yes
Wang and Li [73]	SVM	Yes	Yes		No	Yes
Pinzón et al. [74], [141], [193]	SVM and NN	Yes	Yes	No	No	Yes
Kim and Lee [138]	SVM	Yes	Yes	No	No	Yes
Uwagbole et al. [64]	TC LR and TC SVM	No	Yes	Yes	Yes	No

In comparing the proposal presented in this chapter with existing research work over the years; this chapter focused on web application type to derive a pattern-driven data set by employing SFA technique. The derived pattern-driven data set attribute values are hashed to provide vector matrices to train a supervised learning TC LR and TC SVM classification algorithms for comparison with a better performing model deployed as a web service used in ongoing SQLIA mitigation. The trained, supervised learning models are evaluated through statistical measures in the ROC curve and cross-validated employing k-fold data set partitioning. Table 6-8 above highlights query comparison, pattern-driven data set and emerging computing cloud SDN focused SQLIA mitigation which is the subject of a paper [64] that set apart the proposal in this chapter from existing approaches that apply AI over the years.

6.7 Conclusion

We demonstrated in this chapter a pattern-driven data set generated using SFA in the absence of a pre-existing data set to apply predictive analytics to SQLIA detection and prevention in a big data context. We empirically evaluated our results in the ROC curve with overall better performance metrics that is cross-validated. Also, we observed TC SVM having a better performance metrics with AUC value of *0.986* when compared with the TC LR AUC value of *0.984* to have deployed TC SVM trained model as a web service.

On account of the discussion in the preceding paragraphs, this chapter further the thesis statement that web application type can yield the pattern of web requests that are deemed valid combined with historical studies of different SQLIA types including SQL tokens provide artefact to derive a pattern-driven data set to train an ML model. This research goal is demonstrated in the following contributions of this chapter.

- We further the implementation presented in Chapter 5 with SFA to derive related member strings in crafting the pattern-driven data set. The referred web application expects dictionary words as a valid input while elements of SQL tokens and SQLIA type substitution predicted at the SQLI hotspots are predicted as SQLIA positive.
- We observed using the SFA technique to derive related member strings that we could generate a pattern-driven data set with features of related member strings of any size to train a classifier for SQLIA mitigation to a real-world application that is cross-validated.
- The trained models are evaluated in the ROC curve with the TC SVM has the best performance metrics on AUC value of *0.986* deployed as a web service. The web service is consumed in a web form for input validation, and at the Fiddler proxy API [155] for the ongoing SQLIA prediction as to reject intercepted requests that are SQLIA positive.

We demonstrated in this thesis applied predictive analytics to SQLIA detection and prevention in a big data context with an excellent result that is empirically evaluated and cross-validated as presented above. In benchmarking this thesis against existing work, the methodology proposed here is functional in a big data context which is lacking in existing work before now on SQLIA to our knowledge.

7 Conclusion

Figure 7-1 presents the organisation chart layout of this chapter with the sections.

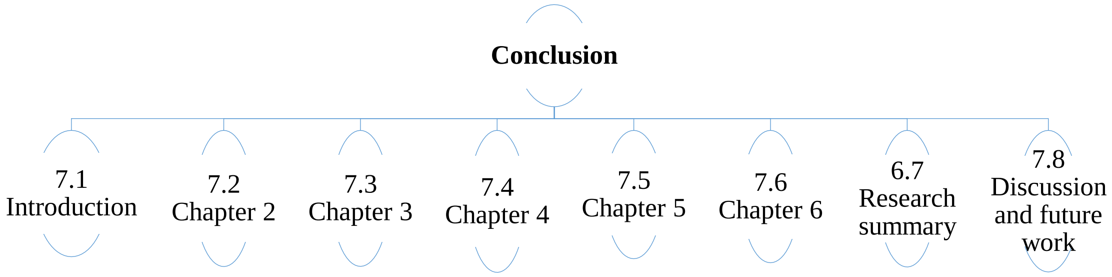


Figure 7-1 The chapter organisation chart: The chapter organisation chart presents the layout of the chapter's sections.

7.1 Introduction

Web applications with SQLIV and instances of SQLIA are still topical [21], [159], [210]. We looked at the issues of non-availability of the ready-made data set to train an ML model in providing SQLIA mitigation, with the few data sets that is in the public domain being outdated [1], [2], [67], [160]. We demonstrated in this thesis and the related published papers [61]–[64] that a web application type being protected can yield a pattern-driven data set to train a supervised learning model in predicting and preventing SQLIA. The trained model is exposed and consumed as a web service at web form and proxy API in ongoing SQLIA prediction and prevention.

There are various SQLIA detection and prevention approaches [4], [5], [95]–[104], [6], [105]–[109], [119]–[123], [7], [124]–[127], [174], [194], [195], [8], [10], [82], [84], [94] proposed by researchers that at the time were a plausible direction towards mitigating SQLIA in the on-premise hosted web servers for web applications. However, with the recent advancement in emerging computing with web application and services hosted in the internet cloud (which defines a new application domain boundary), there is a need for an ML approach to mining big data internet traffic to detect and prevent SQLIA in web requests. Also, past approaches [71], [72], [193], [74], [100], [130], [131], [133]–[135], [141] that relied on the source code for scanning for SQLIA detection and prevention will be hindered when adapted to web-driven cloud-hosted applications where source code access is restricted.

The continuous upward trend in big data has rejuvenated the ML technique being increasingly used to tackle massive data mining of vulnerabilities in web applications as it provides predictive analytics for big data including astronomical growth in web traffic emanating from web requests to cloud-hosted applications. In mitigating SQLIA, the traditional approach of programming looping constructs for string lookup or matching of valid and malicious queries are not functional in big data analytics where AI approaches [61]–[64], [143], [159], [213] provides a dimensional input matrix that makes a lookup of feature weights faster by augmenting the string comparison with the hash value comparison which ML techniques provide.

AI (bio-inspired) approach has the ability to evolve as an excellently trained model will predict SQLIA and continue to self-automate and retrain itself to continuously offer protection which traditional programming constructs of a string lookup or queries matching [8], [82], [102]–[109], [94]–[101] does not offer. The issue that stands out in some existing approaches [74], [125], [129], [133], [193] is computing resource expensive, which by having ML cloud-based robust computing infrastructure provides much needed scalable infrastructure to deliver big data mining of large volume of web requests in delivery ML SQLIA mitigation solution.

However, just as the KDD Cup 1999 [161] has become antiquated in the IDS domain [1], [2], [67], [160], so also has ECML/PKDD 2007 [65] and HTTP CSIC 2010 [66] has become obsolete. These test case data sets served the competition-driven purpose and textbook evaluations at the time, not a benchmark data set to build and evaluate SQLIA mitigation being provided to the real-world application. Thus, this begs the research question and goal of this thesis if the pattern that exists in web requests that are deemed valid, and historical studies of different SQLIA types including SQL tokens can provide artefact to be used to derive a pattern-driven data set to train an ML model and be validated?

The ontology to extract encoded features that form a pattern-driven data set earmarked to train an ML model with evaluation for a web application type in SQLIA mitigation is a contribution of the thesis.

A pattern-driven data set has a set of items of subsequence and substructures that frequently occur together [162]. A pattern is engineered data set devoid of repeating strings or queries that have relevance to the classification problem. It is derived in the scope of the web application type that the ML-based SQLIA mitigation is being provided and can be scaled to any magnitude as to have a significant data set to train a classifier to the maximum as to effectively predict an outcome. A pattern-driven data set empowers the web developer or system expert to derive a data set based on the web application type. Most importantly, it does not age fast as it is tailored to a web application type. Conversely, non-pattern driven data set as seen in some existing work [72] and the few competitions-driven data sets [65], [66] are logs of repeating strings which gravitate the proposals to query comparison of some sort in SQLIA mitigation.

In focusing on the contributions of the thesis chapters we list the following that was recognised in the course of this thesis in the context of emerging computing of big data and cloud hosted services as to have achieved the proposal presented in this thesis.

- A pattern-driven data set inferred from a web application type being protected will break the status quo of lexical or syntactic query comparison for SQLIA mitigation as seen in existing approaches.
- The recent years have seen an astronomical growth of web requests that drives internet traffic to present the challenge of big data in cloud-hosted services; this is beyond the traditional programming lookup using programming looping constructs to search for strings or SQLI signatures.
- The restriction of web application source code for vulnerabilities scanning and legacy application sanitisation as web servers are now hosted in the internet cloud that is accessed through cloud SDN endpoints, which also limits existing approaches that applies SVM learning algorithm [202], [203] in query comparison of some sort.
- A requirement for SQLIA mitigation towards real-world application beyond classical statistical measure performance metrics that stop at ROC curve and CM.

During the research, we submitted and presented four conference papers applying ML predictive analytics to SQLIA prediction and prevention [61]–[64]. We detail in these papers; the novel data set generation and a further improvement of the trained model's prediction results on True Positives (TP) and True Negatives (TN), but with lower False Positives (FP) and False Negatives (FN). The methodology applied here has been empirically evaluated in the numeric encoding of both expected input and patterns of SQLIA types [61], [62]. The good results of the encoded pattern-driven data set to a supervised learning model in performance metrics motivated further works in using R string API and SFA in the derivation of related member strings to obtain a pattern-driven data set presented in IEEE papers [63], [64].

7.2 Chapter 2: Background & Theory

This chapter discussed the fundamentals of SQLIA in the following sections: SQLIA intent (an intruders intention and purpose for SQLIA); SQLIA mechanisms (the conduit for SQLIA); SQLIA types (techniques employed to carry out SQLIA); and a review of the techniques applied in this thesis.

The scheme presented in this thesis employs a web proxy API to intercept web requests of any intent and applies predictive analytics techniques to predict SQLIA at the SQL injection point (predicate and expression locations of the SQL statement structure).

Injection mechanisms to a vulnerable application can originate from web page forms, second-order injection, exploiting web-enabled server variables, query strings, and through cookies. An intruder could exploit injection points using the following SQLIA types in any combination: Tautology; Union; Piggyback; Invalid/Logical queries; Time-based; Obfuscation encoding; and Stored procedure. The SQLIA types and SQL tokens are a source of SQLIA positive labelled data items in the scheme presented in this thesis. Also discussed in this chapter are the fundamentals of the applied techniques in this thesis that includes the use of FSA to generate related member strings, Fiddler proxy to intercept web requests for AI prediction of SQLIA, and MAML being the implementation platform.

7.3 Chapter 3: Literature Review

This chapter examined and reviewed the existing research literature on SQLI detection and prevention techniques to enhance the security of web applications. The research area of SQLI detection and prevention has seen diverse methodologies proposed over the years by various researchers. In this chapter, we broadly discussed these approaches under three categories: (1) SQLIV testing and detection; (2) Approaches that apply defensive coding in web application code sanitisation for SQLI prevention; and (3) Dynamic runtime analysis, including taint-based and methods applying AI. Each section is followed by a discussion on existing literature as to establish the gap in the context of emerging computing, and the contributions this thesis makes to fill the gap. We critique the existing research work on the following:

- They were SQLIA mitigation that existed before the emerging computing.
- The approaches applying ML algorithms use data set that is not pattern-driven.
- The dependence on static code access scanning for SQLIA mitigation in most existing research work hinders intercepting web requests destined to cloud SDN for analysis as cloud-hosted services have restricted access to source code.
- The web requests that drives internet traffic is now big data, this is beyond traditional lookup using classic programming looping constructs to search for attack strings, or SQLI signatures which are not known to be scalable when compared with ML approaches driven by robust cloud-hosted infrastructure in the context of emerging computing.
- The need for a functional and scalable approach to SQLIA mitigation to use of robust cloud-hosted AI platforms such as Azure ML in emerging computing big data mining.
- A need for the historical learning data relevant to a real-world web application type in applying ML technique.

We presented in this thesis the multi-layers ML-based SQLIA mitigation that targets the web client form for input validation, and proxy server at SDN intercepted web requests prediction analytics for SQLIA. This thesis provides a runtime analysis technique of web requests to predict at the SQLIA hotspot (SQL predicate's expression location). Application of ML techniques provides a functional approach to SQLIA mitigation in emerging computing, where applications and services are hosted in the cloud with big data emanating from the web requests to these cloud-hosted applications.

The paradigm shift in this thesis to a pattern-driven data set to train a classifier removes the reliance on antiquated test case data sets [65], [66] and query comparison of some sort. The pattern-driven data set approach provides a technique to derive on the fly a data set relevant to a web application type context requiring ML mitigation to SQLIA. The existing approaches applying ML require full access to source code to require normal and malicious queries for comparison whether in the string lookup or SQL queries matching approaches. In the scheme presented in this thesis, the pattern-driven data set that contains attribute values of related member strings is the subject of prediction during capturing and analysis of web requests by looking at the substitution values destined to the SQL query structure predicate's expression (SQLIA hotspot) location. Understanding expected input data, existing SQLIA signatures, including SQL tokens allows to us concentrate on input data in transition analysis and prediction as against reconstructing the full queries needed in non-pattern driven data set approach.

7.4 Chapter 4: Numerical Encoding to Tame SQLIA

In this chapter, we looked at the issues of data set availability and proposed a pattern-driven data set. Availability of historical data or data set has advanced the use of AI by applying ML techniques. Unfortunately, as there is no unified pre-existing data set, researchers over the years have resorted to various approaches in generating sample data sets with most proposals lacking patterns to enhance learning and are fraught with complex computational overheads.

We investigated if the patterns that exist in any web application type context can be encoded to vector to train a supervised learning model. We conducted experiments and inferred patterns by numeric encoding of features to derive a data set vector of any magnitude to train a supervised learning model in testing the feasibility of mitigating SQLIA employing AI techniques. We presented the following contributions in this thesis chapter.

- The ontology for crafting a pattern-driven data set from the web application type with a technique to encode the data set into vectors required to train a supervised learning model.
- We trained a supervised learning classification algorithm with this pattern-driven data and validated the trained model under various classification algorithms with high performance metrics in the ROC curve, CM and cross-validation as presented in this chapter.
- The success of this conceptual approach of the web application type as the source of a pattern-driven data set to train a classifier has led to a further work in Chapters 5 and 6 by employing string hashing vectorisation in-place of manual numeric encoding in providing a proof of concept of how the proposal will be applied in a real-world application scenario.

We concluded from the experimental results with an empirical evaluation of low prediction error that a pattern-driven data set can be used to train a supervised learning model in towards ML SQLIA mitigation. The high performance metrics of the results of an AUC range of between *0.944* to *1.0* set a precedent for the suitability of applying feature hashing or vectorisation to the pattern-driven data set generated from related member strings in Chapters 5 and 6.

7.5 Chapter 5: Applied Predictive Analytics to SQLIA

This chapter answers the research question: if a web application type to be protected can produce the artefact for the extraction of a pattern-driven data set? We implemented a test web application expecting dictionary word list labelled as SQLIA negative, while SQL tokens and known SQLIA signatures were labelled as SQLIA positive. This labelled pattern-driven data set is vectorised to train a classifier. The vectorisation to obtain vector matrices removes the need for numerical encoding presented in Chapter 4. We evaluated and cross-validated the trained model with statistical measures. We also compared our approach to existing work proposed by other authors to identify the following contributions presented in this thesis chapter.

- We answered the research question that a web application type being protected could provide a pattern-driven data set to apply ML in SQLIA mitigation. We implemented a web application that expects dictionary words as valid input (SQLIA negative) while elements of SQL tokens and SQLIA type substitution at the SQLI hotspot is predicted as SQLIA positive.
- Based on this web application type, we derived a pattern-driven data set using R string API. The pattern-driven data set is vectorised to train a supervised learning model employing an SVM classification algorithm to predict SQLIA. The evaluated, trained SVM model is exposed as a web service which is then consumed in a web form for input validation, and also at proxy API deployed at cloud SDN for intercepting web requests for SQLIA analysis. The intercepted request is analysed to predict SQLIA as to accept the benign and reject malicious web request in ongoing SQLIA mitigation.

We demonstrated in this chapter applied predictive analytics to SQLIA detection and prevention in a big data context with an excellent result that is empirically evaluated. In benchmarking this thesis against existing work, the methodology proposed here is functional in a big data context which is lacking in existing work before now on SQLIA to our knowledge.

7.6 Chapter 6: Pattern-Driven Corpus to Mitigate SQLIA

In this chapter, we further the research question if a web application type to be protected can produce the artefact for the extraction of a pattern-driven data set? We derived a pattern-driven data set by employing FSA and SFA to generate all possible string members to obtain learning data in applying the ML technique to a test application domain context to predict and prevent SQLIA with the following contributions.

- We further the implementation and contributions presented in Chapter 5 with SFA to derive related member strings in crafting the pattern-driven data set. The referred web application expects dictionary words as a valid input while elements of SQL tokens and SQLIA type substitution predicted at the SQLI hotspots are predicted as SQLIA positive.
- Based on the web application type, we derived a pattern-driven data set using SFA as against R string API technique presented in Chapter 5 to derive related member strings. The pattern-driven data set is used to train a supervised learning model employing a TC LR and TC SVM classification algorithms with the better cross-validated classifier selected to predict SQLIA. The trained models are evaluated in the ROC curve with the TC SVM has a better performance metrics of an AUC value of *0.986*. The selected TC SVM trained model is exposed as a web service which is then consumed in a web form for input validation, and proxy API at the cloud SDN for intercepting web requests for SQLIA analysis. The intercepted request is analysed to predict SQLIA as to accept the benign and reject malicious web requests in ongoing SQLIA mitigation.
- We observed using the SFA technique to derive related member strings that we could generate a pattern-driven data set with features of related member strings of any size to train a classifier for SQLIA mitigation for real-world application.

We demonstrated in this chapter a pattern-driven data set generated using SFA in the absence of a pre-existing data set to apply predictive analytics to SQLIA detection and prevention in a big data context. This approach uses hashed pattern-driven data set from a web application type context as against relying on access to the web application's source code or queries comparison, mostly employed in existing static and dynamic signature comparisons of SQLIA artefact research approaches over the years.

7.7 Research summary

SQLIA is still an ongoing issue that affects all organisation types including private, governments and business hosted web applications across the world as intruders exploit vulnerable web applications to pilfer protected data from the database with damaging data security ramifications. Pilfered or leaked data can then be used in various forms of criminality including extortion. The emerging computing of big data and cloud-hosted services posed a more functional issue to SQLIA mitigation that existed before now that involves strings lookup of SQLIA signatures. ML which provides an alternative method to SQLIA string signature lookup lacks an existing robust data set with few that exist being obsolete to train a classifier in advancing SQLIA mitigation. In this thesis, we examined SQLIA artefact of valid web requests, including SQL tokens and SQLIA signatures to derive a pattern-driven data set to train a supervised learning model in applying AI techniques to mitigate SQLIA.

In applying ML to SQLIA problems, there is a need for a data set which prompted in our research goals the following research questions. If a pattern-driven data set can be extracted from both expected web requests and SQL tokens including existing SQLIA signatures be used to train a supervised learning model and be validated, including this pattern-driven data set be extracted from any web application type context for SQLIA mitigation? Below are the novel contributions across the thesis chapters which answer the research goal of the thesis of using a pattern-driven data set obtained from the very web application type to be protected in training an ML model to mitigate SQLIA.

- We reviewed the existing literature on SQLIA to establish the need for a change from on-premise mitigation that includes source code scanning and query comparison of some sort to a SQLIA mitigation that is amenable to emerging computing of increasing big data and cloud-hosted services that can predict SQLIA on web requests in-transition to the back-end database.
- We presented an ontology in Chapter 4 and related publications [61], [62] for crafting a pattern-driven data set using R string API from the web application type with a technique to encode the data set into vectors required to train a supervised learning model in MAML studio. We trained a supervised learning classification algorithm with this pattern-driven data set and validated the trained model under various algorithms. We observed high performance metrics statistical measures, including

cross-validation. The success of this conceptual approach in Chapter 4 has led to further work in Chapters 5 and 6 by employing string hashing vectorisation in-place of manual encoding. Also, we demonstrated a proof of concept of how the proposal will be applied in a real-world web application.

- We further in Chapter 5 and related publication [63] the numeric encoding of features presented in Chapter 4 with hashing vectorisation to obtain vector matrices to train the classification algorithms. In answering the research question if the intended web application type can produce the artefact for a pattern-driven data set, we implemented a web application that expects dictionary words as a valid input while elements of SQL tokens and SQLIA type signatures substitution at the SQLI hotspots is predicted as SQLIA positive. The pattern-driven data set is used to train a supervised learning model employing a TC LR and TC SVM classification algorithms with the better-evaluated and cross-validated classifier selected to predict SQLIA. The selected, trained TC SVM model is exposed as a web service which is then consumed in a web form for input validation, and proxy API at the cloud SDN for intercepting web requests in-transition to a back-end database for SQLIA analysis.
- We demonstrated in Chapter 6 and related publication [64] a more robust method of pattern-driven data set procurement based on the web application type; we derived a pattern-driven data set using FSA and SFA as against R string API technique presented in Chapter 5 to derive related member strings. The referred web application expects dictionary words as a valid input while elements of SQL tokens and SQLIA type substitution predicted at the SQLI hotspots are predicted as SQLIA positive. The pattern-driven data set is used to train a supervised learning model employing a TC LR and TC SVM classification algorithms with the better-evaluated and cross-validated classifier selected to predict SQLIA. The trained TC SVM model is exposed as a web service which is then consumed in a web form for input validation and a proxy API at the cloud SDN for intercepting web request for SQLIA analysis. We observed using the SFA technique to derive related member strings that we could generate a pattern-driven data set with features of related member strings of any size to train a classifier for SQLIA mitigation for real-world application.

We conclude in this thesis that a pattern-driven data set to train a classifier can be inferred from SQLIA types, SQL tokens and expected valid web requests of a web application type. The recent advancement in AI platforms like Azure ML provides the

MAML studio functionality to build, train, validate and cross-validate an ML model as presented in the thesis which is then exposed as a web service that is consumed in multi-layer (client form and cloud SDN proxy) in ongoing SQLIA prediction and prevention.

7.8 Discussion and future work

7.8.1 Discussion

The research area of SQLIA has seen various methodologies proposed over the years [4], [5], [95]–[104], [6], [105]–[109], [119]–[123], [7], [124]–[127], [174], [194], [195], [8], [10], [82], [84], [94]. These proposals are categorised in this thesis into: SQLIV testing and detection [8], [82], [102]–[110], [94]–[101]; defensive coding [111]–[118]; and dynamic runtime analysis [4], [5], [124], [125], [10], [84], [119]–[123] which includes taint-based [7], [126], [127] and ML approaches [71], [72], [134]–[141], [74], [100], [128]–[133]. Though there has been approaches that applied ML in static and dynamic runtime query analysis to mitigate SQLIA, but apart from the defensive coding techniques [111]–[118], the common techniques in most proposals mirror a classical query comparisons.

The classical query comparison SQLIA mitigation approach which includes examining the syntactic structure and taint (trusted sources) reflects the old paradigm of organisation’s web server placement is internet facing from intranet or extranet location and there is a need to mitigate SQLIA at a layer of web application tiers. The processing for malicious input at tier level has seen SQLIA mitigation proposals targeted at different layers: web client; WAF (intercepting HTTP request for outliers); web application (defensive coding and runtime analysis); database firewall (profiling database and encoded or tokenised queries); Multi-layers approach (intercepting and analysing web requests in more than one layer). These existing approaches have consistently followed this paradigm of on-premise located web server for a web application to be protected. Also, these existing approaches gravitate to SQLIA mitigation proposals of query comparison of some sort mostly driven by the method of data set procurement through using web crawler of URLs and other automated unit testing tools. These test case data sets that are logged valid SQL queries for comparison between valid and malicious queries during runtime does not have patterns for predictive analytics as presented in this thesis.

Thus, that begs the question if the pattern that exists in web requests that are deemed valid, and historical studies of different SQLIA types including SQL tokens can provide artefact to be used to derive a pattern-driven data set to train an ML algorithm to predict SQLIA? This thesis demonstrated that a web application type could provide the artefact to derive a pattern-driven data set for ML-based real-world SQLIA mitigation.

7.8.2 Future work

We focused solely on SQLIA analysis and prediction by applying ML techniques to demonstrate that artefact for a pattern-driven data set can be extracted from attack features and valid expected web requests. The same idea proposed in this thesis can be extended to mitigate other similar web vulnerability types. As SQLIA is not the only web application vulnerability that exists, we propose to extend the SQLIA mitigation techniques presented in this thesis to other forms of web application vulnerability attacks.

As automated AI techniques may sound promising to the end-user, but the back-end procuring of data set and training of the classifiers still needs expert knowledge in any domain that applies ML techniques. This thesis requires expert knowledge of the R programming language [214]–[216] and regular expression [36], [153], [206], [217], [218] in crafting the pattern-driven data set. However, this thesis offers a new direction in the pattern-driven data set generation where none exists. Web applications are so diverse as they are built for different purposes, this makes it difficult to have a unified data set to train a classifier. Inferring the expected input data from these diverse web applications still relies on a human system expert to generate the pattern-driven data set that is labelled to train a classifier for prediction accuracy. There is a manual process between obtaining the data set and training a classifier that still needs human intervention, but exploring an ML unsupervised learning (clustering of related member strings) approach holds a solution to be explored in future work.

In training a classifier, there is the need to have a relevant data set for various web applications as web applications are designed for different requirements. As such, there could be no single data set to train a classifier to mitigate SQLIA in all vulnerable web application types (private, commerce, governments, etc.). We emphasise the importance of a pattern-driven data set relevant to the web application in the context of applying AI to SQLIA mitigation as proposed in this thesis.

We hope to take forward the findings in this thesis in creating an off-the-shelf tool to generate this pattern-driven data set in various web application types. This tool will assist developers, including first-hand training of a classifier to predict SQLIA in the legacy web application deployed to the cloud.

The proposals in this thesis apply and leverage the well established AI platforms of MAML studio with the out-of-the-box kernel optimisation. Discussion on kernel types and optimisation is intentionally omitted as that forms the AI core research as against a proposal applying AI techniques. We leverage the out-of-the-box MAML platform, embedded kernels or dot products in the various classification algorithms in the work presented in this thesis.

This thesis proposes intercepting of web requests destined for back-end database for ML predictive analytics to predict SQLIA which implies the web request must be readable or decrypted. To have addressed the decryption of web requests, we use application level proxy API named Fiddler [40]. Thus, all intercepted web requests must be readable for the methods proposed in this thesis to predict SQLIA types accurately. However, the proposal presented in this thesis has a failover to interpret a web request that cannot be decrypted by proxy as obfuscated SQLIA type and deemed it true SQLIA positive.

Finally, we also envisage exploring core AI principles to look at various kernels optimisations in SQLIA mitigation across the different SQLIA types including other forms of web application vulnerabilities. Also, as encryption technology evolves, we hope to continually explore ways to decrypt obfuscated web requests at the application level to make it readable to the SQLIA analysis and mitigation presented in this thesis.

References

[1] A. I. Abubakar, H. Chiroma, S. A. Muaz, and L. B. Ila, “A review of the advances in cyber security benchmark datasets for evaluating data-driven based intrusion detection systems,” in *Procedia Computer Science*, 2015, vol. 62, no. 62, pp. 221–227.

[2] M. Małowidzki, P. Berezi, and M. Mazur, “Network Intrusion Detection : Half a Kingdom for a Good Dataset,” *ECCWS 2017 16th Eur. Conf. Cyber Warf. Secur.*, pp. 1–6, 2017.

[3] K. Bache and M. Lichman, “UCI Machine Learning Repository,” *University of California Irvine School of Information*, vol. 2008, no. 14/8. 2013.

[4] G. T. Buehrer, B. W. Weide, and P. A. G. Sivilotti, “Using Parse Tree Validation to Prevent SQL Injection Attacks,” in *Proceedings of the 5th international workshop on Software engineering and middleware SEM 05*, 2005, no. September, p. 106.

[5] W. G. J. Halfond and A. Orso, “AMNESIA: Analysis and Monitoring for NEutralizing SQL-injection Attacks,” *Proc. 20th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, pp. 174–183, 2005.

[6] S. M. S. Sajjadi and B. Tajalli Pour, “Study of SQL Injection Attacks and Countermeasures,” *Int. J. Comput. Commun. Eng.*, vol. 2, no. 5, pp. 539–542, 2013.

[7] W. G. J. Halfond, A. Orso, and P. Manolios, “Using Positive Tainting and Syntax-aware Evaluation to Counter SQL Injection Attacks,” in *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2006, pp. 175–185.

[8] W. G. J. Halfond, A. Orso, and P. Manolios, “WASP: Protecting web applications using positive tainting and syntax-aware evaluation,” *IEEE Trans. Softw. Eng.*, vol. 34, no. 1, pp. 65–81, 2008.

[9] X. Fu and K. Qian, “SAFELI: SQL Injection Scanner Using Symbolic Execution,” *Proc. 2008 Work. Testing, Anal. Verif. Web Serv. Appl.*, pp. 34–39, 2008.

[10] A. Liu, Y. Yuan, D. Wijesekera, and A. Stavrou, “SQLProb : A Proxy-based Architecture towards Preventing SQL Injection Attacks,” *Conf. Proc. 2009 ACM Symp. Appl. Comput.*, pp. 1–8, 2009.

[11] P. Bisht, P. Madhusudan, and V. N. Venkatakrishnan, “CANDID: Dynamic candidate evaluations for automatic prevention of SQL injection attacks,” *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 2, p. 14:1--14:39, 2010.

[12] P. Pal and S. Bisht, “Improving Web Security by Automated Extraction of Web Application Intent,” 2011.

[13] Imperva, “SQL Injection: By The Numbers | Imperva Cyber Security Blog,” *Imperva*, 2011. [Online]. Available: http://blog.imperva.com/2011/09/sql-injection-by-the-numbers.html. [Accessed: 11-May-2015].

[14] H. Shahriar and M. Zulkernine, “Information-theoretic detection of SQL injection attacks,” in *Proceedings of IEEE International Symposium on High Assurance Systems Engineering*, 2012, pp. 40–47.

[15] Portswigger, “Using Burp to Exploit Blind SQL Injection Bugs | Burp Suite Support Center,” *Portswigger Web Security*. [Online]. Available: https://support.portswigger.net/customer /portal/articles/2163785-Methodology_Blind SQL Injection Exploitation.html. [Accessed: 19-Jun-2017].

[16] B. Damele and M. Stampar, “sqlmap: automatic SQL injection and database takeover tool,” *SQLMap*, 2016. [Online]. Available: http://sqlmap.org/. [Accessed: 19-Jun-2017].

[17] Portcullis Labs, “BSQL Hacker | Portcullis Labs.” [Online]. Available: https://labs.portcullis .co.uk/tools/bsql-hacker/. [Accessed: 19-Jun-2017].

[18] SQLNINJA, “sqlninja - a SQL Server injection & takeover tool,” *sqlninja*. [Online]. Available: http://sqlninja.sourceforge.net/. [Accessed: 19-Jun-2017].

[19] Rapid7, “Penetration Testing Software | Metasploit,” *Metasploit*, 2004. [Online]. Available: https://www.metasploit.com/. [Accessed: 19-Jun-2017].

[20] OWASP, “OWASP Top 10 - 2013,” *OWASP Top 10*, p. 22, 2013.

[21] CodeCurmudgeon, “SQLi Hall-of-Shame,” *The Code Curmudgeon*, 2016. [Online]. Available: http://codecurmudgeon.com/wp/sql-injection-hall-of-shame/. [Accessed: 12-Aug-2016].

[22] D. Pauli, “Researcher arrested after reporting pwnage hole in elections site • The Register,” *The Register*,2016.[Online].Available:http://www.theregister.co.uk/2016/05/09/researcher_arrested_after_reporting_pwnage_hole_in_elections_site/. [Accessed: 08-Aug-2016].

[23] S. Khandelwal, “Fourth, a 16-year-old Hacker, Arrested over TalkTalk Hack,” *The Hacker News*, 2015. [Online]. Available: http://thehackernews.com/2015/11/talktalk-hacker.html. [Accessed: 23-Jan-2016].

[24] K. Fiveash, “TalkTalk claims 157,000 customers were victims of security breach • The Register,” *theregister.co.uk*, 2015. [Online]. Available: http://www.theregister.co.uk/2015/11/06/talktalk_claims_157000_customers_data_stolen/. [Accessed: 05-Jun-2016].

[25] L. Franceschi-Bicchierai, “One of the Largest Hacks Yet Exposes Data on Hundreds of Thousands of Kids | Motherboard,” *Motherboard*, 2015. [Online]. Available: http://motherboard.vice.com/ read/one-of-the-largest-hacks-yet-exposes-data-on-hundreds-of-thousands-of-kids. [Accessed: 23-Jan-2016].

[26] C. Cimpanu, “New Joomla SQL Injection Flaw Is Ridiculously Simple to Exploit,” *BLEEPINGCOMPUTER*, 2017. [Online]. Available: https://www.bleepingcomputer.com/news /security/new-joomla-sql-injection-flaw-is-ridiculously-simple-to-exploit/. [Accessed: 08-Jun-2017].

[27] L. Tung, “Dangerous hole found in McAfee ePO antivirus central management suit - CSO | The Resource for Data Security Executives,” *CSO*, 2017. [Online]. Available: https://www.cso.com.au /article/613679/dangerous-hole-found-mcafee-epo-antivirus-central-management-suit/. [Accessed: 08-Jun-2017].

[28] L. Franceschi-Bicchierai, “Equifax Was Warned - Motherboard,” *Motherboard*, 2017. [Online]. Available: https://motherboard.vice.com/en_us/article/ne3bv7/equifax-breach-social-security-numbers-researcher-warning. [Accessed: 20-Dec-2017].

[29] D. Raywood, “Symantec Calls Vulnerability a Routine Advisory - Infosecurity Magazine,” *Info Security*, 2016. [Online]. Available: https://www.infosecurity-magazine.com/news/symantec-vulnerability-warning/. [Accessed: 18-Jan-2018].

[30] J. S. Davis, “Researcher finds backdoor that accessed Facebook employee passwords,” *SC Media*, 2016. [Online]. Available: https://www.scmagazine.com/researcher-finds-backdoor-that-accessed-facebook-employee-passwords/article/528524/. [Accessed: 18-Jan-2018].

[31] I. Thomson, “If your websites use WordPress, put down that coffee and upgrade to 4.8.3. Thank us

later- SQL-injection security hole needs patching ASAP," *The Register*, 2017. [Online]. Available: https://www.theregister.co.uk/2017/10/31/wordpress_security_fix_4_8_3/.[Accessed:18-Jan-2018].

[32] Z. Whittaker, "This bug let a researcher bypass GoDaddy's site security tool," *ZDNet*, 2017. [Online]. Available: http://www.zdnet.com/article/security-bug-let-hacker-bypass-godaddy-site-firewall-tool/. [Accessed: 18-Jan-2018].

[33] P. Paganini, "Joomla 3.8.4 release addresses three XSS and SQL Injection vulnerabilitiesSecurity Affairs," *Security Affairs*, 2018. [Online]. Available: http://securityaffairs.co/wordpress/68806 /breaking-news/joomla-3-8-4-xss-sqlinjection.html. [Accessed: 13-Mar-2018].

[34] P. Nair, "Security researcher hacks BSNL intranet; leaks details of 47,000 employees | Computerworld India," *COMPUTERWORLD*. [Online]. Available: http://www.computerworld.in /news/security-researcher-hacks-bsnl-intranet-leaks-details-47000-employees. [Accessed: 13-Mar -2018].

[35] INDIAN EXPRESS, "French researchers highlight security flaws in Indian Railways portals- The New Indian Express," *INDIAN EXPRESS*, 2018. [Online]. Available: http://www.newindian express.com/cities/hyderabad/2018/may/09/french-researchers-highlight-security-flaws-in-indian-railways-portals-1812279.html. [Accessed: 04-Jul-2018].

[36] K. Thompson, "Programming Techniques: Regular expression search algorithm," *Commun. ACM*, vol. 11, no. 6, pp. 419–422, Jun. 1968.

[37] Microsoft, "Regular Expression Language," *MSDN*, 2016. [Online]. Available: https://msdn. microsoft.com/en-us/library/az24scfc.aspx. [Accessed: 08-Apr-2017].

[38] M. Gagolewski, "Package 'stringi': Character String Processing Facilities," *CRAN*, 2016.

[39] IBM, "IBM Knowledge Center - Tokens," *IBM*. [Online]. Available: https://www.ibm.com /support/knowledgecenter/en/SSCRJT_5.0.3/com.ibm.swg.im.bigsql.commsql.doc/doc/r0000719. html. [Accessed: 02-Jul-2016].

[40] E. Lawrence, "Fiddler free web debugging proxy," *Telerik*. [Online]. Available: http://www .telerik.com/fiddler. [Accessed: 11-Feb-2015].

[41] Microsoft Azure, "Microsoft Azure Machine Learning Studio," *Microsoft Azure Machine Learning*. [Online]. Available: https://studio.azureml.net/. [Accessed: 25-Jan-2015].

[42] Microsoft, "Microsoft Azure Machine Learning," *studio.azureml.net*, 2016. [Online]. Available: https://studio.azureml.net/. [Accessed: 11-Aug-2016].

[43] A. Rajpurohit, "Why Azure ML is the Next Big Thing for Machine Learning?," *Kdnuggets*, 2015.

[44] P. A. Today, "Top 15 Artificial Intelligence Platforms," *Predictive-Analytics-Today*, 2017. [Online]. Available: https://www.predictiveanalyticstoday.com/artificial-intelligence-platforms/. [Accessed: 09-Jun-2018].

[45] Azure ML, "Sample 3: Cross Validation for Binary Classification: Adult Dataset | Azure AI Gallery," *Microsoft*, 2015. [Online]. Available: https://gallery.azure.ai/Experiment/d9d3c1 1b360343be9c9cbdc6a3981350. [Accessed: 10-Jul-2016].

[46] A. de Ruiter, "Using ROC plots and the AUC measure in Azure ML | Andreas De Ruiter's BI blog," *MSDN*, 2015. [Online]. Available: https://blogs.msdn.microsoft.com/andreasderuiter/2015/02/09/ using-roc-plots-and-the-auc-measure-in-azure-ml/. [Accessed: 15-Mar-2016].

[47] Microsoft Azure, “Cross-Validate Model,” *Micorsoft Azure*, 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/cross-validate-model. [Accessed: 09-Jun-2018].

[48] R. Kohavi, “A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection,” in *Appears in the International Joint Conference on Articial Intelligence (IJCAI)*, 1995.

[49] T. Fushiki, “Estimation of prediction error by using K-fold cross-validation,” *Stat. Comput.*, vol. 21, no. 2, pp. 137–146, Apr. 2011.

[50] T. G. Tape, “Using the Receiver Operating Characteristic (ROC) curve to analyze a classification model,” *Univ. Nebraska Med. Cent.*, pp. 1–3, 2000.

[51] T. Tape, “The Area Under an ROC Curve,” *Interpreting Diagnostic Tests*. [Online]. Available: http://gim.unmc.edu/dxtests/ROC3.htm. [Accessed: 16-Apr-2017].

[52] Y. Bengio and Y. Grandvalet, “No unbiased estimator of the variancee of k-fold cross-valudation,” *J Mach Learn Res*, 2004.

[53] T. G. Tape, “Interpreting Diagnostic Tests,” *University of Nebraska Medical Center*. [Online]. Available: http://gim.unmc.edu/dxtests/Default.htm. [Accessed: 17-Sep-2017].

[54] K. Markham, “Simple guide to confusion matrix terminology,” *Data Sch.*, pp. 1–9, 2014.

[55] K. H. Zou, A. J. O’Malley, and L. Mauri, “Receiver-operating characteristic analysis for evaluating diagnostic tests and predictive models,” *Circulation*, vol. 115, no. 5, pp. 654–657, Feb. 2007.

[56] M. O. Rabin and D. Scott, “Finite Automata and Their Decision Problems,” *IBM J. Res. Dev.*, vol. 3, no. 2, pp. 114–125, 1959.

[57] FiniteStateMachine, “Finite-state machines,” *J. Franklin Inst.*, vol. 275, no. 3, p. 250, 1963.

[58] Wikipedia, “Finite-state machine,” *Wikipedia*. .

[59] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A. Smola, “Feature Hashing for Large Scale Multitask Learning,” *Int. Conf. Artif. Intell. Stat.*, pp. 496–503, 2009.

[60] Wikipedia, “Feature hashing,” *Wikipedia*. [Online]. Available: Feature hashing. [Accessed: 09-Sep-2016].

[61] S. O. Uwagbole, W. Buchanan, and L. Fan, “Applied web traffic analysis for numerical encoding of SQL injection attack features,” in *European Conference on Information Warfare and Security, ECCWS*, 2016, vol. 2016–Janua.

[62] S. O. Uwagbole, W. J. Buchanan, and L. Fan, “Numerical encoding to Tame SQL injection attacks,” in *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, vol. 2016–Janua, pp. 1253–1256.

[63] S. O. Uwagbole, W. J. Buchanan, and L. Fan, “Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention,” in *3rd IEEE/IFIP Workshop on Security for Emerging Distributed Network Technologies (DISSECT)*, 2017.

[64] S. O. Uwagbole, W. J. Buchanan, and L. Fan, “An applied pattern-driven corpus to predictive analytics in mitigating SQL injection attack,” in *2017 Seventh International Conference on Emerging Security Technologies (EST)*, 2017, pp. 12–17.

[65] ECML/PKDD 2007, “Attack Challenge -ECML/PKDD Workshop,” *ECML/PKDD 2007*. [Online]. Available: http://www.lirmm.fr/pkdd2007-challenge/. [Accessed: 23-Jun-2015].

[66] G. Á. M. Carmen Torrano Giménez, Alejandro Pérez Villegas, “Http Dataset Csic 2010,” *Information Security Institute of CSIC (Spanish Research National Council)*, 2010. [Online]. Available: http://www.isi.csic.es/dataset/.

[67] M. V. Mahoney and P. K. Chan, “An Analysis of the 1999 DARPA/Lincoln Laboratory Evaluation Data for Network Anomaly Detection,” Springer, Berlin, Heidelberg, 2003, pp. 220–237.

[68] E. Vasilomanolakis, C. G. Cordero, N. Milanov, and M. Mühlhäuser, “Towards the creation of synthetic, yet realistic, intrusion detection datasets,” in *Proceedings of the NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 1209–1214.

[69] 1999 KDD Cup, “http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html,” *The UCI KDD Archive*, 1999. .

[70] E. D. Halsey, “With AI and Data, it’s ‘Junk in, Junk Out’ – Source Institute – Medium,” *Medium Corporation*, 2017. [Online]. Available: https://medium.com/source-institute/with-ai-and-data-its-junk-in-junk-out-18b33e8f391e. [Accessed: 03-Jan-2018].

[71] R. Komiya, I. Paik, and M. Hisada, “Classification of malicious web code by machine learning,” in *Proceedings of 2011 3rd International Conference on Awareness Science and Technology, iCAST 2011*, 2011, pp. 406–411.

[72] J. Choi, C. Choi, H. Kim, and P. Kim, “Efficient malicious code detection using N-gram analysis and SVM,” in *Proceedings - 2011 International Conference on Network-Based Information Systems, NBiS 2011*, 2011, vol. 11, no. 2, pp. 618–621.

[73] Y. Wang and Z. Li, “SQL Injection Detection via Program Tracing and Machine Learning,” *LNCS*, vol. 7646, pp. 264–274, 2012.

[74] C. I. Pinzón, J. F. De Paz, Á. Herrero, E. Corchado, J. Bajo, and J. M. Corchado, “IdMAS-SQL: Intrusion Detection Based on MAS to Detect and Block SQL injection through data mining,” *Inf. Sci. (Ny).*, vol. 231, pp. 15–31, May 2013.

[75] C. Bockermann, M. Apel, and M. Meier, “Learning SQL for database intrusion detection using context-sensitive modelling (extended abstract),” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, vol. 5587 LNCS, pp. 196–205.

[76] H. T. Nguyen, C. Torrano-Gimenez, G. Aalvarez, K. Franke, and S. Petrović, “Enhancing the effectiveness of web application firewalls by generic feature selection,” *Log. J. IGPL*, vol. 21, no. 4, pp. 560–570, Aug. 2013.

[77] S.-H. H. Ahn, N.-U. U. Kim, and T.-M. M. Chung, “Big data analysis system concept for detecting unknown attacks,” in *International Conference on Advanced Communication Technology, ICACT*, 2014, pp. 269–272.

[78] Y. Wang and Z. Li, “SQL injection detection with composite kernel in support vector machine,” *Int. J. Secur. its Appl.*, vol. 6, no. 2, pp. 191–196, 2012.

[79] Y. Wang, D. Wang, W. Zhao, and Y. Liu, “Detecting SQL Vulnerability Attack Based on the Dynamic and Static Analysis Technology,” in *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, 2015, vol. 3, pp. 604–607.

[80] M. Veanes, “Rex @ rise4fun from Microsoft,” *Microsoft Research*. [Online]. Available:

http://rise4fun.com/rex.

[81] R. B. McGhee, “Some finite state aspects of legged locomotion,” *MBiosci*, vol. 2, no. 1, pp. 67–84, 1968.

[82] W. G. J. Halfond, J. Viegas, and A. Orso, “A Classification of SQL Injection Attacks and Countermeasures,” *Prev. Sql Code Inject. By Comb. Static Runtime Anal.*, p. 53, 2008.

[83] D. Ariu and G. Giacinto, “HMMPayl: an application of HMM to the analysis of the HTTP Payload,” *Advances*, vol. 11, pp. 81–87, Sep. 2010.

[84] S. W. Boyd and A. D. Keromytis, “SQLrand: Preventing SQL injection attacks,” in *Applied Cryptography and Network Security*, 2004, pp. 292–302.

[85] W. G. J. Halfond and A. Orso, “Combining static analysis and runtime monitoring to counter SQL-injection attacks,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 30, no. 4, pp. 1--7, 2005.

[86] A. Liu, Y. Yuan, D. Wijesekera, and A. Stavrou, “SQLProb,” *Proc. 2009 ACM Symp. Appl. Comput. - SAC ’09*, p. 2054, 2009.

[87] D. Appelt, N. Alshahwan, and L. Briand, “Assessing the Impact of Firewalls and Database Proxies on SQL Injection Testing,” Springer, Cham, 2014, pp. 32–47.

[88] D. Appelt, A. Panichella, and L. Briand, “Automatically Repairing Web Application Firewalls Based on Successful SQL Injection Attacks,” in *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, 2017, pp. 339–350.

[89] A. Joshi and V. Geetha, “SQL Injection detection using machine learning,” *Control. Instrumentation, Commun. Comput. Technol. (ICCICCT), 2014 Int. Conf.*, no. 2, pp. 1111–1115, 2014.

[90] E. H. Cheon, Z. Huang, and Y. S. Lee, “Preventing SQL Injection Attack Based on Machine Learning,” *Int. J. Adv. Comput. Technol.*, vol. 5, no. 9, pp. 967–974, 2013.

[91] B. Zhou, R. Karuparthi, and B. Zhou, “Enhanced Approach to Detection of SQL Injection Attack,” *2016 15th IEEE Int. Conf. Mach. Learn. Appl.*, pp. 1–4, Dec. 2016.

[92] D. Appelt, C. D. Nguyen, and L. Briand, “Behind an application firewall, are we safe from SQL injection attacks?,” in *2015 IEEE 8th International Conference on Software Testing, Verification and Validation, ICST 2015 - Proceedings*, 2015.

[93] P. Hernandez, “Microsoft Preps for Azure Machine Learning Launch.,” *eWeek*, p. 1, 2014.

[94] C. Gould, Z. Su, and P. Devanbu, “JDBC checker: a static analysis tool for SQL/JDBC applications,” *Softw. Eng. 2004. ICSE 2004. Proceedings. 26th Int. Conf.*, pp. 697–698, 2004.

[95] C. Gould, Z. Su, P. Devanbu, G. Wassermann, C. Gould, Z. Su, P. Devanbu, G. Wassermann, C. Gould, Z. Su, and P. Devanbu, “Static checking of dynamically generated queries in database applications,” *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 4, pp. 645–654, Sep. 2004.

[96] G. Wassermann and Z. Su, “An analysis framework for security in Web applications,” *SAVCBS 2004 Specif. Verif. Component-Based Syst.*, p. 70, 2004.

[97] X. Fu, X. Lu, B. Peltsverger, S. Chen, K. Qian, and L. Tao, “A static analysis framework for detecting SQL injection vulnerabilities,” in *Proceedings - International Computer Software and Applications Conference*, 2007, vol. 1, pp. 87–94.

[98] E. Al-Khashab, F. S. Al-Anzi, and A. A. Salman, “PSIAQOP: Preventing SQL Injection Attacks Based on Query Optimization Process,” in *Proceedings of the Second Kuwait Conference on e-Services and e-Systems*, 2011, p. 10:1----10:8.

[99] L. K. Shar and H. B. K. Tan, “Predicting common web application vulnerabilities from input validation and sanitization code patterns,” *Proc. 27th IEEE/ACM Int. Conf. Autom. Softw. Eng. - ASE 2012*, p. 310, 2012.

[100] I. Medeiros, N. Neves, and M. Correia, “Detecting and Removing Web Application Vulnerabilities with Static Analysis and Data Mining,” *IEEE Trans. Reliab.*, vol. 65, no. 1, pp. 54–69, Mar. 2016.

[101] Y.-W. Huang, S.-K. Huang, T.-P. Lin, and C.-H. Tsai, “Web application security assessment by fault injection and behavior monitoring,” *Proc. World Wide Web Conf.*, p. 148, 2003.

[102] L. Liu, J. Xu, H. Yang, C. Guo, J. Kang, S. Xu, B. Zhang, and G. Si, “An Effective Penetration Test Approach Based on Feature Matrix for Exposing SQL Injection Vulnerability,” in *Proceedings - International Computer Software and Applications Conference*, 2016, vol. 1, pp. 123–132.

[103] Y. Shin, L. Williams, and T. Xie, “SQLUnitGen: Test Case Generation for SQL Injection Detection,” 2006, pp. 2006–21.

[104] A. Sharma, “Exploiting undefined behaviors for efficient symbolic execution,” in *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, 2014, pp. 727–729.

[105] J.-C. Lin, J.-M. Chen, and H.-K. Wong, “An Automatic Meta-revised Mechanism for Anti-malicious Injection,” in *Network-Based Information Systems*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 98–107.

[106] H. Shahriar and M. Zulkernine, “MUSIC: Mutation-based SQL injection vulnerability checking,” in *Proceedings - International Conference on Quality Software*, 2008, pp. 77–86.

[107] A. Ciampa, C. A. Visaggio, and M. Di Penta, “A heuristic-based approach for detecting SQL-injection vulnerabilities in Web applications,” in *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems*, 2010, pp. 1–7.

[108] D. Appelt, C. D. Nguyen, L. C. Briand, and N. Alshahwan, “Automated testing for SQL injection vulnerabilities: an input mutation approach,” in *Proceedings of the International Symposium on Software Testing and Analysis*, 2014, pp. 259–269.

[109] M. Ceccato, C. D. Nguyen, D. Appelt, and L. C. Briand, “SOFIA: An automated security oracle for black-box testing of SQL-injection vulnerabilities,” in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, pp. 167–177.

[110] D. Appelt, “Automated Security Testing of Web-Based Systems Against SQL Injection Attacks,” 2016.

[111] R. A. McClure and I. H. Krüger, “Sql Dom,” in *Proceedings of the 27th international conference on Software engineering - ICSE ’05*, 2005, p. 88.

[112] OWASP INJECTION, “SQL Injection Prevention Cheat Sheet,” *2016*, 2016. [Online]. Available: https://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet. [Accessed: 18-May-2016].

[113] A. Owasp, “OWASP Top 10 Proactive Controls 2016,” 2016.

[114] J. D. Meier, A. Mackman, B. Wastell, P. Bansonde, and A. Wigley, “How To: Prevent Cross-Site

Scripting in ASP.NET," *Microsoft*, 2005. [Online]. Available: https://msdn.microsoft.com/en-us/library/ff648339.aspx. [Accessed: 19-Mar-2017].

[115] D. Wichers, "Input validation cheat sheet," 2013.

[116] S. Thomas, L. Williams, and T. Xie, "On automated prepared statement generation to remove SQL injection vulnerabilities," *Inf. Softw. Technol.*, vol. 51, no. 3, pp. 589–598, 2009.

[117] K. Natarajan and S. Subramani, "Generation of Sql-injection Free Secure Algorithm to Detect and Prevent Sql-Injection Attacks," *Procedia Technol.*, vol. 4, pp. 790–796, 2012.

[118] V. Karakoidas, D. Mitropoulos, P. Louridas, and D. Spinellis, "A type-safe embedding of SQL into Java using the extensible compiler framework J%," *Comput. Lang. Syst. Struct.*, vol. 41, pp. 1–20, 2015.

[119] H. Fernando and J. Abawajy, "Securing RFID systems from SQLIA," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 7017 LNCS, no. PART 2, pp. 245–254.

[120] D. Kar, S. Panigrahi, and S. Sundararajan, "SQLiDDS: SQL Injection Detection Using Query Transformation and Document Similarity," *Distrib. Comput. Internet Technol. Icdcit 2015*, vol. 8956, pp. 377–390, 2015.

[121] F. Valeur, D. Mutz, and G. Vigna, "A learning-based approach to the detection of sql attacks," *Intrusion Malware Detect. Vulnerability Assess.*, pp. 123–140, 2005.

[122] Y. S. Jang and J. Y. Choi, "Detecting SQL injection attacks using query result size," *Comput. Secur.*, vol. 44, pp. 104–118, 2014.

[123] K. Wei, M. Muthuprasanna, and S. Kothari, "Preventing SQL injection attacks in stored procedures," in *Proceedings of the Australian Software Engineering Conference, ASWEC*, 2006, vol. 2006, pp. 191–198.

[124] D. Das, U. Sharma, and D. Bhattacharyya, "An Approach to Detection of SQL Injection Attack Based on Dynamic Query Matching," *Int. J. Comput. …*, vol. 1, no. 25, pp. 28–34, 2010.

[125] X. Fu and C.-C. Li, "A String Constraint Solver for Detecting Web Application Vulnerability," *Proc. 22nd Int. Conf. Softw. Eng. Knowl. Eng. - SEKE'2010*, pp. 535–542, 2010.

[126] G. Wassermann and Z. Su, "Sound and precise analysis of web applications for injection vulnerabilities," *ACM SIGPLAN Not.*, vol. 42, no. 6, p. 32, 2007.

[127] A. Kie, P. J. Guo, and M. D. Ernst, "Automatic Creation of SQL Injection and Cross-Site Scripting Attacks," 2009.

[128] I. Medeiros, N. F. Neves, and M. Correia, "Automatic detection and correction of web application vulnerabilities using data mining to predict false positives," in *Proceedings of the 23rd international conference on World wide web - WWW '14*, 2014, pp. 63–74.

[129] B. Smith and L. Williams, "Using SQL hotspots in a prioritization heuristic for detecting all types of web application vulnerabilities," in *Proceedings - 4th IEEE International Conference on Software Testing, Verification, and Validation, ICST 2011*, 2011, pp. 220–229.

[130] G. M. Howard, C. N. Gutierrez, F. A. Arshad, S. Bagchi, and Y. Qi, "PSigene: Webcrawling to generalize SQL injection signatures," in *Proceedings - 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2014*, 2014, pp. 45–56.

[131] R. Yan, X. Xiao, G. Hu, S. Peng, and Y. Jiang, “New deep learning method to detect code injection attacks on hybrid applications,” *J. Syst. Softw.*, vol. 137, pp. 67–77, Mar. 2018.

[132] Y. Yan, P. Guo, and L. Liu, “A novel hybridization of artificial neural networks and ARIMA models for forecasting resource consumption in an IIS web server,” in *Proceedings - IEEE 25th International Symposium on Software Reliability Engineering Workshops, ISSREW 2014*, 2014, pp. 437–442.

[133] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Network Anomaly Detection: Methods, Systems and Tools,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[134] D. Atienza, Á. Herrero, and E. Corchado, “Neural analysis of HTTP traffic for web attack detection,” in *Advances in Intelligent Systems and Computing*, 2015, vol. 369, pp. 201–212.

[135] T. Threepak and A. Watcharapupong, “Web attack detection using entropy-based analysis,” in *International Conference on Information Networking*, 2014, pp. 244–247.

[136] R. Awasthi and D. Mangal, “An Approach Based on SVM Classifier to Detect SQL Injection Attack,” vol. 3, no. 3, pp. 256–260, 2016.

[137] D. Kar, S. Panigrahi, and S. Sundararajan, “SQLiGoT: Detecting SQL Injection Attacks using Graph of Tokens and SVM,” *Comput. Secur.*, vol. 60, pp. 206–225, 2016.

[138] M. Y. Kim and D. H. Lee, “Data-mining based SQL injection attack detection using internal query trees,” *Expert Syst. Appl.*, vol. 41, no. 11, pp. 5416–5430, 2014.

[139] B. D. Priyaa and M. I. Devi, “Hybrid SQL injection detection system,” in *ICACCS 2016 - 3rd International Conference on Advanced Computing and Communication Systems: Bringing to the Table, Futuristic Technologies from Arround the Globe*, 2016, pp. 1–5.

[140] A. Giordani and A. Moschitti, “Translating Questions to {SQL} Queries with Generative Parsers Discriminatively Reranked,” *Coling2012*, no. December 2012, pp. 401–410, 2012.

[141] C. Pinzón, Á. Herrero, J. F. De Paz, E. Corchado, and J. Bajo, “CBRid4SQL: A CBR intrusion detector for SQL injection attacks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6077 LNAI, no. PART 2, pp. 510–519.

[142] Microsoft Azure, “Feature Hashing,” *Machine Learning*, 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/feature-hashing. [Accessed: 11-Jun-2018].

[143] N. M. Sheykhkanloo, “SQL-IDS: Evaluation of SQLi Attack Detection and Classification Based on Machine Learning Techniques,” in *Proceedings of the 8th International Conference on Security of Information and Networks*, 2015, pp. 258–266.

[144] R. M. Pandurang and D. C. Karia, “A mapping-based podel for preventing Cross site scripting and SQL injection attacks on web application and its impact analysis,” *Proc. 2015 1st Int. Conf. Next Gener. Comput. Technol. NGCT 2015*, pp. 414–418, Sep. 2016.

[145] N. Khan, J. Abdullah, and A. S. Khan, “Defending Malicious Script Attacks Using Machine Learning Classifiers,” *Wirel. Commun. Mob. Comput.*, vol. 2017, pp. 1–9, 2017.

[146] I. Jacobs, “URIs, Addressability, and the use of HTTP GET and POST,” *Tech. Archit. Gr. Find.*, 2004.

[147] Y. Aboukir, “SQL Injection through HTTP Headers - InfoSec Resources,” *Infosec*, 2012. [Online].

Available: http://resources.infosecinstitute.com/sql-injection-http-headers/.

[148] J. McCray, “Manual Web Application Testing Basics - Pastebin.com,” *Pastebin*, 2015. [Online]. Available: http://pastebin.com/ka5pvLp8. [Accessed: 21-Aug-2016].

[149] Crown, “Computer Misuse Act 1990,” *Comput. Misuse Act 1990*, p. 16, 2008.

[150] sqlinjection, “SQL Injection Using UNION,” *sqlinjection*, 2016. [Online]. Available: http://www.sqlinjection.net/union/. [Accessed: 03-Feb-2016].

[151] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[152] B. Brown, “Finite State Automata-youtube,” *Youtube video*, 2015. [Online]. Available: https://www.youtube.com/watch?v=SCqXMzt4pf0. [Accessed: 19-Apr-2017].

[153] M. Veanes, P. De Halleux, N. Tillmann, P. de Halleux, N. Tillmann, P. De Halleux, and N. Tillmann, “Rex: Symbolic regular expression explorer,” in *ICST 2010 - 3rd International Conference on Software Testing, Verification and Validation*, 2010, pp. 498–507.

[154] M. Veanes, N. Bjorner, and L. de Moura, “Symbolic Automata Constraint Solving,” vol. 6397. 2010.

[155] Telerik, “Fiddler - The Free Web Debugging Proxy by Telerik,” *Telerik*, 2012. [Online]. Available: http://www.telerik.com/fiddler. [Accessed: 25-Jun-2017].

[156] J. Barnes, *Azure Machine Learning Microsoft Azure Essentials*. 2015.

[157] OWASP, “SQL Injection - OWASP,” *OWASP*, 2016. [Online]. Available: https://www.owasp.org/index.php/SQL_Injection_Bypassing_WAF. [Accessed: 19-Jul-2017].

[158] M. Zwitter and M. Soklic, “UCI Machine Learning Repository: Breast Cancer Data Set,” *UCI*, 2012. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original).

[159] G. Betarte, E. Giménez, R. Martínez, and Á. Pardo, “Machine learning-assisted virtual patching of web applications,” 2018.

[160] J. McHugh, “Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory,” *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000.

[161] S. J. Stolfo, “KDD cup 1999 dataset,” *UCI KDD Repos. http//kdd.ics.uci.edu*, p. 0, 1999.

[162] M. Gupta and J. Han, “Approaches for Pattern Discovery Using Sequential Data Mining,” *Pattern Discov. Using Seq. Data …*, no. c, pp. 1–20, 2012.

[163] S. B. Kotsiantis, “Supervised Machine Learning : A Review of Classification Techniques,” *Informatica*, vol. 31, pp. 249–268, 2007.

[164] P. Dayan, *Unsupervised learning*. 2009.

[165] D. Mitropoulos and D. Spinellis, “SDriver: Location-specific signatures prevent SQL injection attacks,” *Comput. Secur.*, vol. 28, no. 3–4, pp. 121–129, 2009.

[166] T. Scholte, W. Robertson, D. Balzarotti, and E. Kirda, “Preventing input validation vulnerabilities

inweb applications through automated type analysis," in *Proceedings - International Computer Software and Applications Conference*, 2012, pp. 233–243.

[167] L. K. Shar and H. B. K. Tan, "Defeating SQL injection," *Computer (Long. Beach. Calif).*, vol. 46, no. 3, pp. 69–77, Mar. 2013.

[168] M. K. Gupta, M. C. Govil, and G. Singh, "Static analysis approaches to detect SQL injection and cross site scripting vulnerabilities in web applications: A survey," in *International Conference on Recent Advances and Innovations in Engineering, ICRAIE 2014*, 2014, pp. 1–5.

[169] D. Mitropoulos, P. Louridas, M. Polychronakis, and A. D. Keromytis, "Defending Against Web Application Attacks: Approaches, Challenges and Implications," *IEEE Trans. Dependable Secur. Comput.*, pp. 1–1, 2017.

[170] T. Pietraszek and C. Vanden Berghe, "Defending against injection attacks through context-sensitive string evaluation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006, vol. 3858 LNCS, pp. 124–145.

[171] G. Wassermann, C. Gould, Z. Su, and P. Devanbu, "Static checking of dynamically generated queries in database applications," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 4, p. 14–es, Sep. 2007.

[172] D. Li, Y. Lyu, M. Wan, and W. G. J. Halfond, "String analysis for Java and Android applications," *Proc. 2015 10th Jt. Meet. Found. Softw. Eng. - ESEC/FSE 2015*, pp. 661–672, 2015.

[173] X/Open Company., *Data management : SQL Call Level Interface (CLI)*. X/Open Company, 1995.

[174] A. Nguyen-Tuong, S. Guarnieri, D. Greene, J. Shirley, and D. Evans, "Automatically hardening web applications using precise tainting," in *IFIP Advances in Information and Communication Technology*, 2005, vol. 181, pp. 295–307.

[175] J. D. McCaffrey, "Creating Neural Networks using Azure Machine Learning Studio | James D. McCaffrey." [Online]. Available: https://jamesmccaffrey.wordpress.com/2014/09/24/creating-neural-networks-using-azure-machine-learning-studio/. [Accessed: 23-Jul-2017].

[176] Amazon, "Amazon Machine Learning - Predictive Analytics with AWS," *Amazon.com*, 2016. [Online]. Available: https://aws.amazon.com/machine-learning/. [Accessed: 11-Aug-2016].

[177] IBM, "IBM Watson Analytics," *IBM Watson Analytics*, 2016. .

[178] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 87, no. 23, pp. 9193–9196, Dec. 1990.

[179] Microsoft, "Reserved Keywords (Transact-SQL)," *MSDN*. [Online]. Available: https://msdn .microsoft.com/en-us/library/ms189822.aspx.

[180] E. Anderson, "The Species Problem in Iris," *Ann. Missouri Bot. Gard.*, vol. 23, no. 3, p. 457, Sep. 1936.

[181] R. A. Fisher, "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS," *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[182] M. van der Loo, "{stringdist}: an {R} Package for Approximate String Matching," *R J.*, vol. 6, no. 1, pp. 111–122, 2014.

[183] MSDN, “Matching Behavior,” *MSDN Library*. [Online]. Available: https://msdn.microsoft .com/en-us/library/0yzc2yb0(v=vs.100).aspx. [Accessed: 03-Feb-2016].

[184] Microsoft Azure, “Normalize Data - Azure Machine Learning Studio | Microsoft Docs.” [Online]. Available:https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/ normalize-data. [Accessed: 01-Jul-2018].

[185] Microsoft, “Cheat sheet: How to choose a MicrosoftML algorithm - Machine Learning Server | Microsoft Docs,” *Microsoft*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/machine-learning-server/r/how-to-choose-microsoftml-algorithms-cheatsheet. [Accessed: 25-Feb-2018].

[186] Microsoft Azure, “Clean and prepare data for Azure Machine Learning | Microsoft Docs,” *Microsoft Azure*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-data-science-prepare-data. [Accessed: 14-Apr-2017].

[187] Nitesh V. Chawla et. al, “SMOTE,” *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.

[188] D. Becker, “What is Log Loss? | Kaggle,” *Kaggle*, 2018. [Online]. Available: https://www. kaggle.com/dansbecker/what-is-log-loss. [Accessed: 26-Jul-2018].

[189] R. A. B. Collier, “Making Sense of Logarithmic Loss - datawookie,” *Datawokie*, 2015. [Online]. Available: https://datawookie.netlify.com/blog/2015/12/making-sense-of-logarithmic-loss/. [Accessed: 26-Jul-2018].

[190] Microsoft Azure, “Evaluate Model - Azure Machine Learning Studio | Microsoft Docs,” *Microsoft Azure*, 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/evaluate-model. [Accessed: 27-Jul-2018].

[191] J. D. McCaffrey, “Log Loss and Cross Entropy are Almost the Same | James D. McCaffrey,” *2016 DevIntersection Conference Pre-Event Interview*, 2016. [Online]. Available: https:// jamesmccaffrey.wordpress.com/2016/09/25/log-loss-and-cross-entropy-are-almost-the-same/. [Accessed: 26-Jul-2018].

[192] Deep Learning Course Wiki, “Log Loss - Deep Learning Course Wiki,” *Deep Learning Course Wiki*, 2017. [Online]. Available: http://wiki.fast.ai/index.php/Log_Loss. [Accessed: 26-Jan-2018].

[193] C. Pinzón, J. F. De Paz, J. Bajo, Á. Herrero, and E. Corchado, “AIIDA-SQL: An Adaptive Intelligent Intrusion Detector Agent for detecting SQL injection attacks,” in *2010 10th International Conference on Hybrid Intelligent Systems, HIS 2010*, 2010, pp. 73–78.

[194] T. Pietraszek and C. Vanden Berghe, “Defending Against Injection Attacks Through Context Sensitive String Evaluation,” *Recent Adv. Intrusion Detect.*, pp. 124–145, 2006.

[195] A. Naderi-Afooshteh, A. Nguyen-Tuong, M. Bagheri-Marzijarani, J. D. Hiser, and J. W. Davidson, “Joza: Hybrid Taint Inference for Defeating Web Application SQL Injection Attacks,” in *Proceedings of the International Conference on Dependable Systems and Networks*, 2015, vol. 2015–Septe, pp. 172–183.

[196] Microsoft, “Access SQL: basic concepts, vocabulary, and syntax - Access,” *MS Office*, 2007. [Online]. Available: https://support.office.com/en-gb/article/Access-SQL-basic-concepts-vocabulary-and-syntax-444d0303-cde1-424e-9a74-e8dc3e460671.

[197] Progress Telerik Fiddler, “Configure Fiddler to decrypt HTTPS traffic | Progress Telerik Fiddler Documentation,” 2018. [Online]. Available: https://docs.telerik.com/fiddler/Configure-Fiddler/ Tasks/DecryptHTTPS. [Accessed: 24-Jun-2018].

[198] MSDN, “Filter-Based Feature Selection,” *Microsoft Azure*. [Online]. Available: https://msdn.

microsoft.com/en-us/library/azure/dn905854.aspx. [Accessed: 18-Feb-2018].

[199] Andreas de Ruiter, “Setting the threshold of a binary learning model in Azure ML – Andreas De Ruiter’s BI blog,” *Microsoft*. [Online]. Available: https://blogs.msdn.microsoft.com /andreasderuiter/2015/02/10/setting-the-threshold-of-a-binary-learning-model-in-azure-ml/. [Accessed: 21-Aug-2016].

[200] Microsoft Azure, “Deploy a Machine Learning web service | Microsoft Docs,” *Microsoft Azure*, 2017. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-publish-a-machine-learning-web-service. [Accessed: 30-Apr-2017].

[201] H. Huang, L. Qian, and Y. Wang, “A SVM-based technique to detect phishing URLs,” *Inf. Technol. J.*, vol. 11, no. 7, pp. 921–925, 2012.

[202] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[203] V. N. Vapnik, “The Nature of Statistical Learning Theory,” *Springer*. 1995.

[204] Microsoft Azure, “Two-Class Support Vector Machine,” *MSDN Library*, 2016. [Online]. Available: https://msdn.microsoft.com/library/azure/12d8479b-74b4-4e67-b8de-d32867380e20/?f=255&MSPPError=-2147217396. [Accessed: 22-Jan-2016].

[205] M. Veanes, “Applications of symbolic finite automata,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7982 LNCS, pp. 16–23.

[206] MSDN, “Regular Expression Language - Quick Reference,” *Microsoft MSDN*. [Online]. Available: https://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx. [Accessed: 05-Mar-2016].

[207] MSDN, “Details of Regular Expression Behavior.” [Online]. Available: https://msdn.microsoft .com/en-us/library/e347654k.aspx. [Accessed: 10-Jul-2015].

[208] Wikipedia, “Kleene star,” *Wikipedia*, 2017. [Online]. Available: https://en.wikipedia.org/wiki /Kleene_star#Examples. [Accessed: 11-Nov-2017].

[209] G. Ericson and J. Takaki, “Tune Model Hyperparameters - Azure Machine Learning Studio | Microsoft Docs,” *Microsoft Azure ML*, 2018. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/tune-model-hyperparameters. [Accessed: 18-Feb-2018].

[210] D. Appelt, C. D. Nguyen, A. Panichella, and L. C. Briand, “A Machine Learning-Driven Evolutionary Approach for Testing Web Application Firewalls,” *IEEE Trans. Reliab.*, pp. 1–24, 2018.

[211] Atefeh Tajpour, Maslin Massrum, Mohammad Zaman Heydari, A. Tajpour, M. Massrum, and M. Z. Heydari, “Comparison of SQL injection detection and prevention techniques,” 2010, pp. V5-174-V5-179.

[212] M. Kaushik and G. Ojha, “Attack Penetration System for SQL Injection,” *Int. J. Adv. Comput. Res.*, vol. 4, no. 2, pp. 724–732, 2014.

[213] N. M. Sheykhkanloo, N.M. Sheykhkanloo, N. M. Sheykhkanloo, and N.M. Sheykhkanloo, “Employing Neural Networks for the Detection of SQL Injection Attack,” in *Proceedings of the 7th International Conference on Security of Information and Networks*, 2014, pp. 318–323.

[214] R. Analytics, “What is R?,” *R-project.org*, 2015.

[215] J. McCaffrey, “R programming succinctly,” 2017.

[216] J. Chambers and Bell Laboratories, “What is R?,” *r-project.org*, no. 1, 2000.

[217] R. Cox, “Regular Expression Matching Can Be Simple And Fast,” 2007. [Online]. Available: http://swtch.com/~rsc/regexp/regexp1.html. [Accessed: 05-Feb-2018].

[218] L. D’Antoni and M. Veanes, “Minimization of symbolic automata,” *Proc. 41st ACM SIGPLAN-SIGACT Symp. Princ. Program. Lang. - POPL ’14*, pp. 541–553, 2014.