



**SQL Injection Vulnerability Detection using Deep Learning:  
A Web Feature Based Approach**

By

**TONMOY GHOSH  
(151-35-932)**

A thesis submitted in partial fulfillment of the requirement for the degree  
of Bachelor of Science in Software Engineering

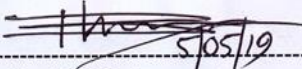
**Department of Software Engineering  
DAFFODIL INTERNATIONAL UNIVERSITY**

Spring – 2019


## APPROVAL

This Thesis titled “SQL Injection Vulnerability Detection using Deep Learning: A Web Feature Based Approach”, submitted by Tonmoy Ghosh, 151-35-932 to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc.in Software Engineering and approved as to its style and contents.


## BOARD OF EXAMINERS

  
-----  
**Dr. Touhid Bhuiyan**  
**Professor and Head**  
Department of Software Engineering  
Faculty of Science and Information Technology  
Daffodil International University


**Chairman**

  
-----  
**Md. Maruf Hassan**  
**Assistant Professor**  
Department of Software Engineering  
Faculty of Science and Information Technology  
Daffodil International University

**Internal Examiner 1**

  
-----  
**Asif Khan Shakir**  
**Lecturer**  
Department of Software Engineering  
Faculty of Science and Information Technology  
Daffodil International University

**Internal Examiner 2**

  
-----  
**Dr. Md. Nasim Akhtar**  
**Professor**  
Department of Computer Science and Engineering  
Faculty of Electrical and Electronic Engineering  
Dhaka University of Engineering & Technology, Gazipur

**External Examiner**

## DECLARATION

It hereby declare that this thesis has been done by me under the supervision of Mr. Md. Maruf Hassan, Assistant Professor, Department of Software Engineering, Daffodil International University. It also declare that neither this thesis nor any part of this has been submitted elsewhere for award of any degree.

*Tommay Ghosh*

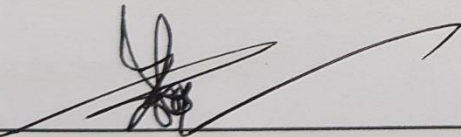
**Student Name:**

**Student ID:**

**Batch:**

Department of Software Engineering  
Faculty of Science & Information Technology  
Daffodil International University

**Certified by:**



**Name of the Supervisor:**

**Designation**

Department of Software Engineering  
Faculty of Science & Information Technology  
Daffodil International University

## **ACKNOWLEDGEMENT**

First of all, I am grateful to the Almighty God for giving me the ability to complete the final thesis.

I would like to express my gratitude to my supervisor Mr. Md. Maruf Hassan for the consistent help of my thesis and research work, through his understanding, inspiration, energy, and knowledge sharing. His direction helped me to finding the solutions of research work and reach to my final theory.

I would like to express my extreme sincere gratitude and appreciation to all of my teachers of Software Engineering department for their kind help, generous advice and support during the study.

I am also express my gratitude to all of my friend's, senior, junior who, directly or indirectly, have lent their helping hand in this venture.

Last but not the least, I would like to thank my family; my parents Mr. Tapon Ghosh and Mrs. Sabita Ghosh, for giving birth to me at the first place and supporting me spiritually throughout my life.

Tonmoy Ghosh.

# TABLE OF CONTENT

<b>APPROVAL</b> .....	Error! Bookmark not defined.
<b>DECLARATION</b> .....	<b>ii</b>
<b>ACKNOWLEDGEMENT</b> .....	<b>iv</b>
<b>TABLE OF CONTENT</b> .....	<b>v</b>
<b>LIST OF TABLES</b> .....	<b>vii</b>
<b>LIST OF FIGURES</b> .....	<b>viii</b>
<b>ABSTRACT</b> .....	<b>ix</b>
<b>CHAPTER 1:INTRODUCTION</b> .....	<b>1</b>
1.1 Background .....	1
1.2 Motivation of the Research .....	1
1.3 Problem Statement .....	1
1.4 Research Questions .....	2
1.5 Research Objectives .....	2
1.6 Research Scope .....	2
1.7 Thesis Organization .....	3
<b>CHAPTER 2:LITERATURE REVIEW</b> .....	<b>4</b>
<b>CHAPTER 3:RESEARCH METHODOLOGY</b> .....	<b>8</b>
3.1 Introduction.....	8
3.2 Proposed Method .....	8
3.3 Automated Tool .....	10
3.4 Tool’s Contribution.....	11
3.5 Data Collection .....	12
3.6. Data Preprocessing.....	14
3.6.1. Inconsistency Reduction .....	15
3.6.2. Null Value Reduction.....	16
3.6.3. Data Standardization .....	16
3.7. Feature Selection.....	16
3.7.1. Correlation .....	17
3.7.2. Chi-Square .....	18
3.5 Summary .....	19
<b>CHAPTER 4:RESULTS AND DISCUSSION</b> .....	<b>20</b>
4.1 Correlation Result .....	20
4.2. Chi-Square Result .....	21
4.3. Model Accuracy and Loss.....	23

<b>CHAPTER 5: CONCLUSIONS AND RECOMMENDATIONS</b> .....	<b>25</b>
5.1 Findings and Contributions .....	25
5.2 Recommendations for Future Works .....	25
<b>REFERENCES</b> .....	<b>26</b>
<b>Appendix – A</b> .....	<b>29</b>
<b>Parameters of Student Dropout Data Set</b> .....	<b>29</b>
<b>Appendix – B</b> .....	<b>30</b>
Pseudo code 1: Code sample of Data Standardization .....	30
Pseudo code 2: Code sample of Correlation .....	30
Pseudo code 3: Code sample of Chi-Square .....	31
Pseudo code 4: Code sample of Deep Learning Model .....	31

## LIST OF TABLES

Table 4.2.2: Significance of features based on Chi-Square Score .....	22
---	----

## LIST OF FIGURES

Figure 3.2.1: Proposed model of SQL injection vulnerability detection .....	9
Figure 3.3.1: Structure of SQL injection vulnerability finding tool .....	10
Figure 3.4.1: Combination of model and tool.....	11
Figure 3.5.1: Features from dataset.....	12
Figure 4.1.1: Correlation heatmap with reduced features .....	21
Figure 4.2.1: Features with Chi-Square score.....	22
Figure 4.3.1: Obtained accuracy of the model .....	23
Figure 4.3.2: Curve of loss of the model .....	24



## ABSTRACT

**Background:** Application of SQL injection poses a serious security threat to the database driven web applications. By this injection attack someone can easily steal the potential sensitive information and have access to the application's underlying database. A hacker can access the content of the database by specifically designed input. This can bring harm to personal or private wealth.

**Objective:** The objective is to make a dataset of SQL injection vulnerability with web applications various types of features. And make a deep learning model that can detect web applications SQL injection vulnerability. Also build a tool that can detect which web applications are SQL injection vulnerable and which are not.

**Results:** With the help of deep learning and my self-constructed web application feature-based dataset I've obtained an accuracy of 98.2%.

**Conclusions:** This paper discusses a technique for identifying SQL injection vulnerability using web application's feature-based method with the help of deep learning. Feature based SQL dataset is not available. I've discussed about it.

**Keywords:** Injection Attack, SQL Injection, Deep Learning, Web Application;

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Background**

The uses of web applications increasing day by day in an extensive rate. Nowadays people started using web application to do businesses, online transactions, make communications etc. Web applications need data from people to make these things happen. These data contain so many private data and these are stored in a database. When it is the matter of private data then the subject of security comes forward. Though security is very advance these days, even then some intelligent people (sometimes we call them hacker) somehow finds out vulnerability to the security system and exploits it with some techniques. Among those techniques SQL injection is one technique. It is very dangerous for web applications because it can make a chaos to the database.

### **1.2 Motivation of the Research**

The rapid rise in injection attack becomes a headache to web developers. According to the IBM X-Force Threat Intelligence Index, 79% of attacks are involved to the injection attack. And the rate of the injection attack has increased at 37% in 2017 compared to 2016. Also, the organization Open Web Application Security Project (OWASP) has listed the injection attack at the top position among other web application attacks. There is no tool has implemented focused on the web applications features.

### **1.3 Problem Statement**

In the field of information technology data is everything. Data about people's private data, organization's confidential data, government's confidential data etc. To store

these data database is required and data are stored in database in electronic form. And databases are connected with web application. People indirectly communicates with database by directly using web application. SQL injection also done through web application using a browser's search bar. Attackers adds different types of data fetching statements with the URL. And they use some logic with the URL so that database's security gets convince and provide desired data. Then the attacker gets what he wanted and chaos begins. So that is a big problem for web application. Since SQL injection attacks mostly performed through web application so that we tried to find out the web application's features by which attackers find out a web application's SQL injection vulnerability.

#### **1.4 Research Questions**

Question 1: Does this method counter web application's SQL injection vulnerability with feature-based technique using deep learning?

#### **1.5 Research Objectives**

- To propose a new solution to the SQL injection vulnerability problem.
- To build a deep learning model which would make good prediction of that vulnerability.
- To produce a tool which would detect SQL injection vulnerability automatically and compare that with existing SQL injection detection tools.

#### **1.6 Research Scope**

Deep learning techniques are used in lot of research areas such as classification problem, image processing, and many other detection problems. In this paper I've

gathered some features from web application and made a dataset with it. Using that dataset, I'll try to find out a web application's SQL injection vulnerability with the help of deep learning.

## **1.7 Thesis Organization**

This research document is in APA referencing format. It contains five different chapters which are described below:

Chapter 1: In this chapter I have discussed about the background of my thesis. I also have mentioned motivation behind the research, research objectives, research scope etc.

Chapter 2: Here I've discussed about previous existing methods about SQL injection detection from different researchers, their limitations, research types and key notes.

Chapter 3: Here I've described about my methodology and approaches that I've used in SQL injection vulnerability detection.

Chapter 4: Here I've described about the results of my methodology that I have used in chapter 3.

Chapter 5: This chapter concluded with the research study outcomes and focuses on the limitation of the research. It also described the direction of further work.

## CHAPTER 2

### LITERATURE REVIEW

Many researchers worked with SQL injection. Some of them worked with SQL injection detection, some of them worked with SQL injection detection, some of them worked with detection & prevention. Some researchers classified SQL injection in their paper. Some researchers used machine learning technique to detect and SQL injection and discussed about the prevention. But no one focused on web application's feature from which we can tell a web application SQL vulnerable. And no one used deep learning technology.

Randa Osman Morsi and Mona Farouk Ahmed (2019) proposed an algorithm which is based on the combination of two of the existing detection algorithms: pattern matching algorithm using Aho-Corasick (AC) and PT (Parsing Tree). And they've obtained 99.9% accuracy.

Ely Salwana Mat Surin, Nurhakimah Azwani Md Najib, Chan Wei Liang, Mohd Amin Mohd Yunus, Muhammad Zainulariff Brohan and Nazri Mohd Nawi (2019) studied on various SQL injection prevention technique from existing journal papers. They proposed a key generation and identification method for avoiding SQL Injection based on Blockchain concept.

S. Nanhay, D. Mohit, R.S. Raw, and K. Suresh (2016) defined SQL injection as the method of hacking that executes malicious SQL queries on the database server via a web-based application. They also discussed about the strategy on how to defend SQL injection in the journal and proposed solution to prevent it.

K.G. Vamshi, V. Trinadh, S. Soundabaya and A. Omar (2016) they've explained about how SQL injection works and the defensive mechanism against these threats. Their proposed mechanism for prevent are processing inputs, Sp\_executesql replaced with QUOTENAME and permission management.

K. Krit and S. Chitsutha (2016) they have explained about the methodology to prevent SQL Injection on Server-Side Scripting using Machine Learning approach. They used Support Vector Machine (SVM), Boosted Decision Tree, Artificial Neural Network, Decision Jungle to predict SQL injection. They got Decision Jungle as the best predictor.

P.K. Raja and Z. Bing (2016) proposed an enhanced approach to dynamic query matching technique by imposing a sanitizer for quick and easy detection of SQL injection attack. They developed a sanitizer to check runtime SQL queries. If the query passed the sanitizer then they compared the query with a master file which contains legal SQL queries to detect SQL injections.

Dubey, R., & Gupta, H. (2016) discussed about SQL injection techniques and proposed methodology on prevention of SQL injection attack. They filter queries that users want to execute by placing to the proxy server. Then they check user's authenticity to validate the query. They compare other their technique with other researchers technique and proved their technique best.

Dr. Ahmad Ghafarian (2017) proposed a hybrid method which consists database design, implementation and common gateway interface to detect and prevent SQL injection attack. He combined static and dynamic analysis to prevent SQL injection.

Debabrata Kar and Suvasini Panigrahi (2013) discussed about various types of SQL injection attacks and proposed a prevention technique based on query transform and hashing. They transform a query into its structural form instead of the parameterized form.

Li Qian, Zhenyuan Zhu, lun Hu and Shuying Liu (2015) introduces typical SQL injection attack and prevention technologies. Their detecting methods not only validates user input, but also use type-safe SQL parameters.

Romil Rawat and Shailendra Kumar Shrivastav (2012) they have used query tokenization system and performed SQL injection detection using Support Vector Machine (SVM) with an accuracy of 96.47%.

David Scott and Richard Sharp discussed on web application vulnerabilities and application level web security. They showed how SQL attacks are performed in the web application.

V.Shanmuganeethi, Ra. Yagna Pravin, C.EmilinShyni, S.Swamynathan (2011) used web services for detection of SQL injection vulnerability in web applications. They proposed a mechanism to intercept SQL statements without any modification of an application using Aspect Oriented Programming and to analyze the query for its legitimacy, and to customize the errors.

Ashish Kumar and Sumitra Binu (2018) discussed a technique for identifying and preventing SQL injection attack using tokenization concept. The paper discusses a function which verifies the user queries for the presence of various predefined tokens and thereby preventing the access to web pages in cases where the user query includes any of the defined tokens.

Chenyu M., & Fan G. (2016) used Intention Oriented Detection technique to defend against SQL injection attack. They developed a language named SQL Injection Description Language (SQLIDL) to detect the intention of SQL injection. The SQLIDL is used to transform SQL requests into string sets formalized by the Deterministic Finite Automaton (DFA).

Abirami J., Devakunchari R. and Valliyammai C. (2015) did a survey on existing SQL injection detection and prevention techniques which are done by various researchers. They discussed on those techniques working procedures.

Abhay K.Kolhe and Pratik Adhikari (2014) discussed on the injection technique, detection technique and prevention technique of SQL injection attack. They proposed IP tracking method for detection of SQL injection attack and MSQLi and MySQL\_real\_escape\_string() techniques for prevention SQL injection attack.

Voitovych O.P., Yuvkovetskyi O.S. and Kupershtein L.M. (2016) discussed on SQL injection types and proposed prevention technique against it. They validate all inputs by users and generates request signature. They used Secure Shell to filter output information to prevent SQL injection attack

Chen, Z., Guo, M., & zhou, L. (2018) used made a vocabulary of common words which are used in dataset using word2vector method and then used Support Vector Machine (SVM) algorithm to detect SQL injection attack.

Wahid Rajeh and Alshreef Abed (2017) proposed three-tier SQL injection detection method and mitigation for cloud environments. Their methodology involves dynamic, static and runtime prevention and detection mechanisms.



## **CHAPTER 3**

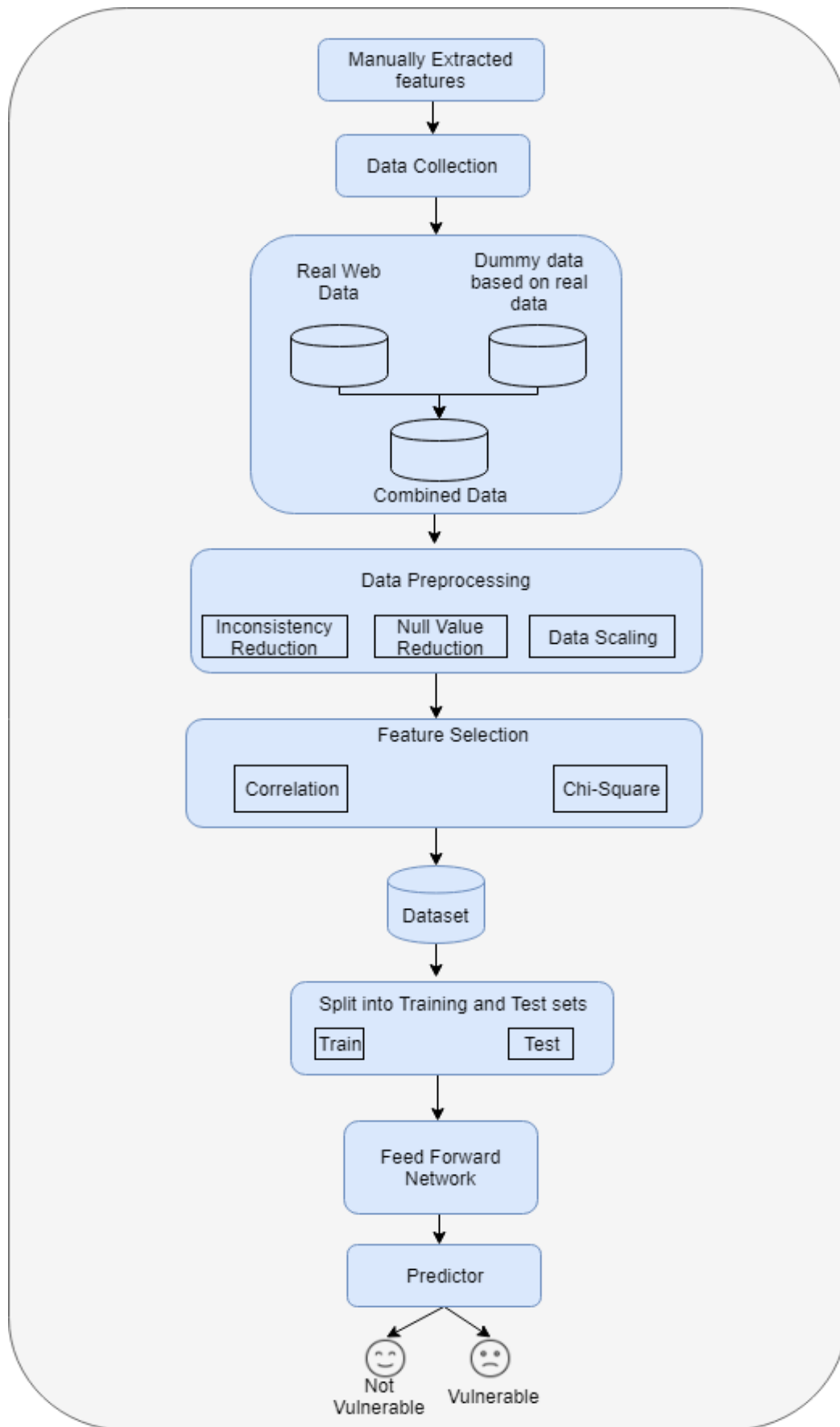
### **RESEARCH METHODOLOGY**

#### **3.1 Introduction**

Each issue solver takes after some preprocessing way to deal with take care of their issues. This research additionally takes after some preprocessing system on logical methodologies. The research methodology separated into a few sections for getting the outcome as much as literal such as data collection, data preprocessing, data analysis and Visualizing the outcome.

#### **3.2 Proposed Method**

A novel method has been proposed by us for the detection of SQL injection vulnerability of web application based on web application's various features. Due to unavailability of dataset to do this type of work I have constructed a dataset using these web application features and performed a classification operation with the help of deep learning. I've tried to increase my dataset's integrity using various data preprocessing, feature selection technique. Proposed model is shown in Figure 3.2.1.



**Figure 3.2.1:** Proposed model of SQL injection vulnerability detection.

### 3.3 Automated Tool

I have developed and automated tool which takes URL as input then tells us the web application of given URL is vulnerable to SQL injection or not. Structure of that tool is shown in Figure 3.3.1.

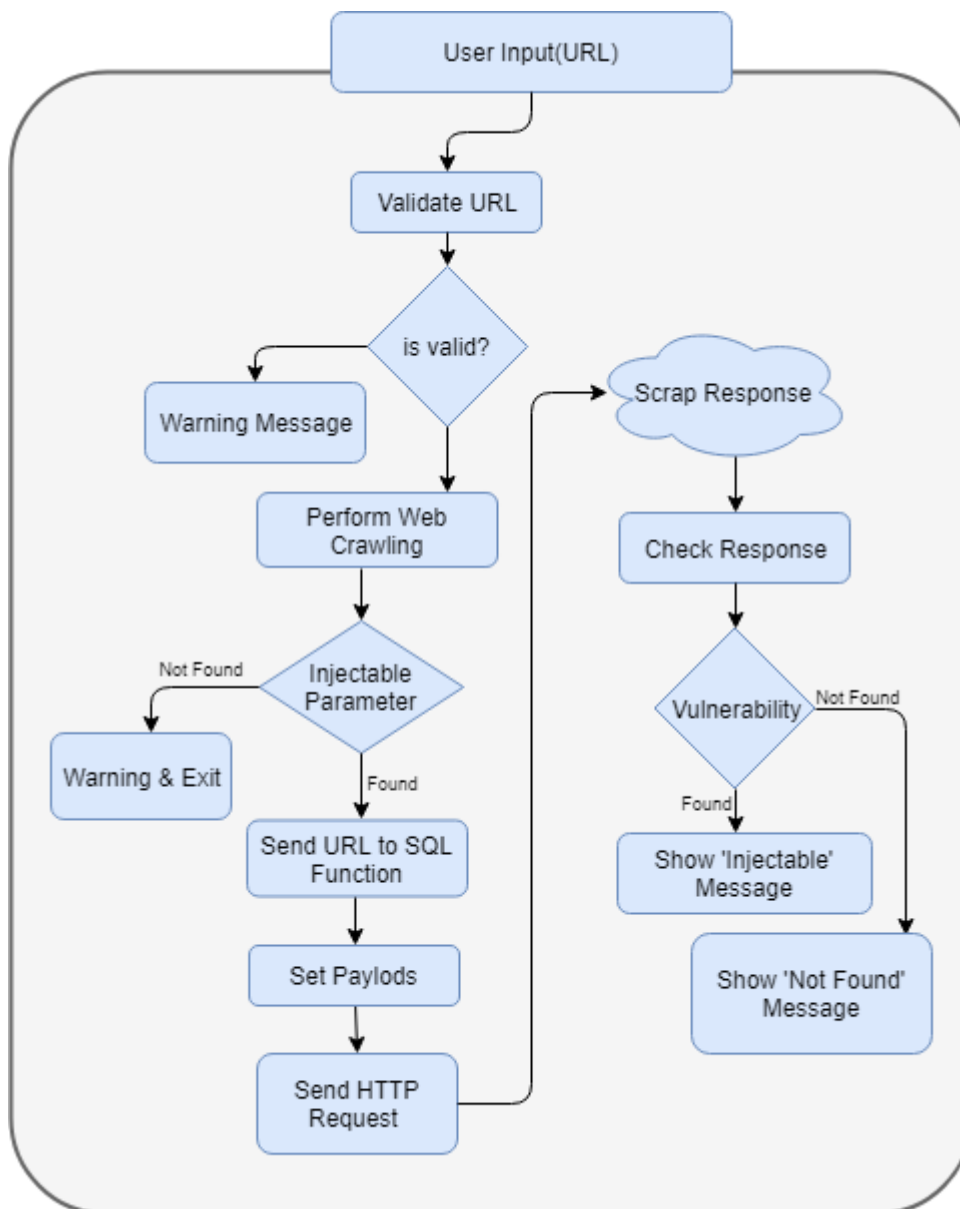


Figure 3.3.1: Structure of SQL injection vulnerability finding tool.

### 3.4 Tool's Contribution

Using my developed tool, I can make a simple test set using my extracted feature set that can be used to the predictor to predict the web application's SQL injection vulnerability. Proper visualization is shown in Figure 3.4.1.

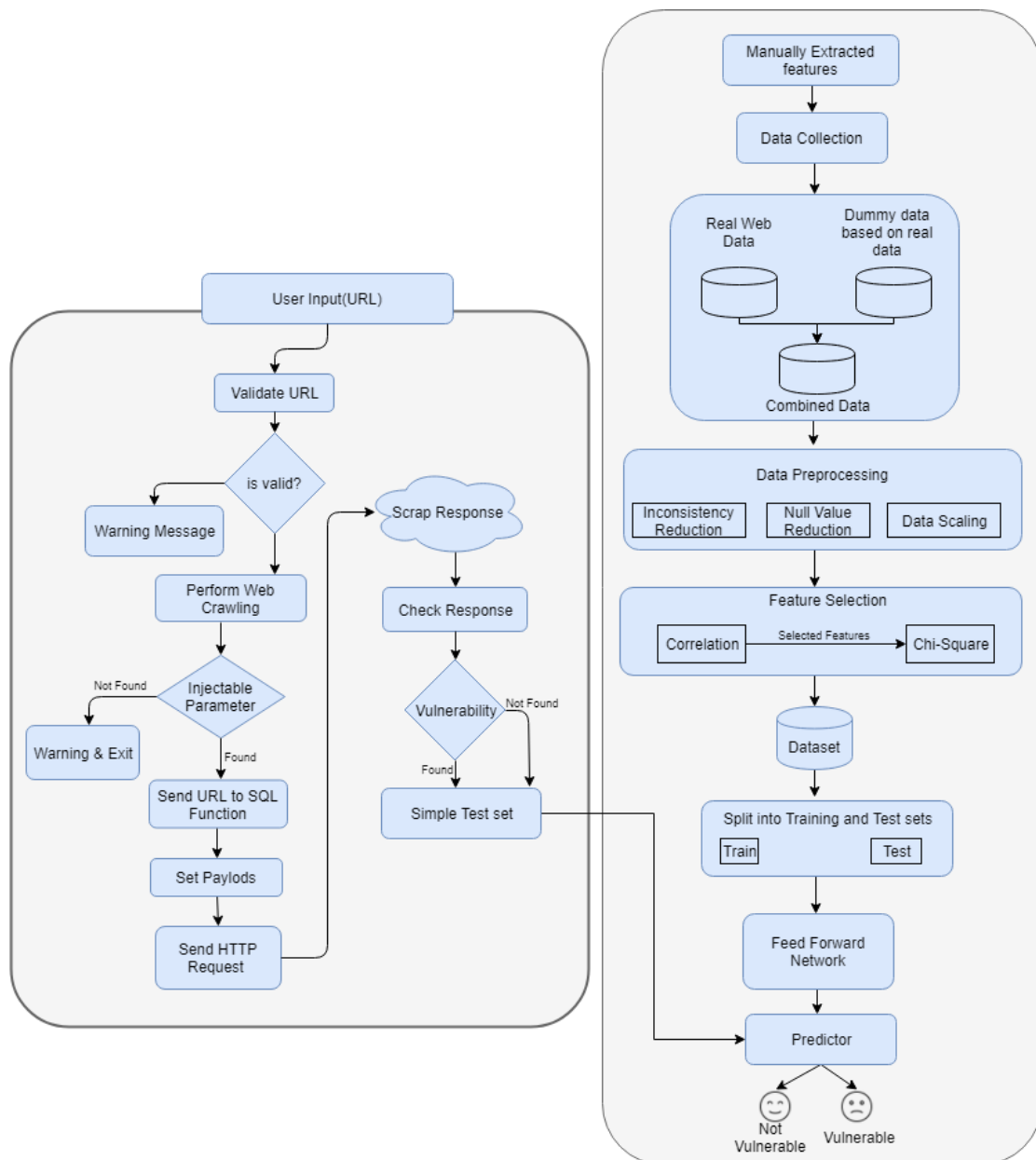


Figure 3.4.1: Combination of model and tool.

### 3.5 Data Collection

I have extracted features based on various types of behavior of web application. I've come to know about these behaviors of web application from various research paper and Cisco's documentation on SQL injection. SQL injection vulnerability can be disclosed when an attacker tries to execute different types of command through web application in a legal way which has no means to database and then database throws exception. I have identified different execution points of web application to execute malicious code which interacts with database.

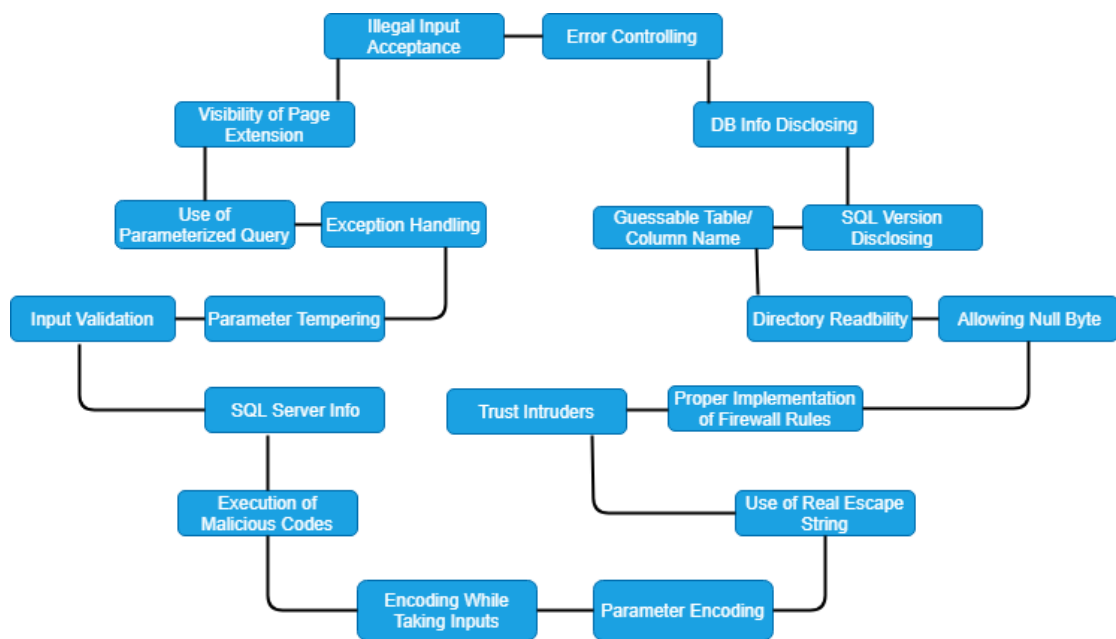


Figure 3.5.1: Features from dataset.

Demonstration of features are given below:

- Input validation: This is the most useful way to find SQL injection vulnerability of a web application. Attackers input a single quotation (') at the end of the URL

or use “1” at any input field of a web application to get the exception from database.

- Parameter tempering: Attacker uses parameter which does not exist for a particular web application and get exception from database and makes sure about low security of that particular web application.
- Exception Handling: That exception is thrown by database leads to the disclosure of SQL injection vulnerability.
- Use of parameterize queries: Parameterized queries increases the possibility of SQL injection vulnerability disclosure. More parameter more chances of finding vulnerability.
- Visibility of page extension: Page extension helps the attacker to guess about the possible web technologies used in a web application.
- Illegal input acceptance: Illegal input helps attacker to get exception from database.
- Error controlling: Sometimes errors from server could also lead to finding SQL injection vulnerability.
- DB info disclosing: Information about database helps the attacker to find the drawbacks of that particular database.
- SQL version disclosing: Disclosure of SQL version may lead the attacker to use any kind of bug of that SQL to perform vulnerability finding operation.
- Guessable table/column name: If the attacker somehow can make the right guess of table name or column name then it becomes easy for him/her to perform vulnerability finding operation.
- Directory readability: Showing all the directories can help attacker guess about the ways of inject SQL queries.

- Allowing null byte: There's an exploitation technique known as null byte injection which is used in web infrastructure to bypass sanity checking filters by adding URL encoded null byte characters to the user supplied data. By this process intended logic of the application can be altered. This also allows to get unauthorized access to the system files of a web application by malicious adversary.
- Proper implementation of firewall rules: If there is no proper implementation of firewall then it becomes easier to perform an attack.
- Trust intruder: If there is no intruder prevention method used for the security of web application then attempts of exploitation increase.
- Use of real escape string: Escape string prevents attacker to execute malicious code which contains unnecessary characters.
- Parameter encoding: Encoded parameter decreases the chances of being exploited.
- Encoding data while taking input: Encoding all the input data decreases the exploitation chances.
- Execution of malicious code: Ability of execution of malicious code can lead to the exploitation of web application.
- SQL server info: Disclosure of SQL server info helps attacker to determine how to find vulnerability.

### **3.6. Data Preprocessing**

Data preprocessing deals with the surety of increasing the integrity of data in dataset. In the initial phase, I have collected raw data then I have extracted features from it then I have performed feature selection. At last I have found an optimal dataset based on the selected features and I have fed it to the classification model for training purpose. Data

preprocessing is done between feature extraction and feature selection. I also can get dataset from third parties. These datasets may also need preprocessing and selection operation. Data preprocessing refers to process the data so that it can be applied to model and makes sure the model is not learning with wrong information. After data collection the data analysts try to understand the data. They use several statistical techniques; they try to visualize the data for better understanding. Because real world data are noisy, incomplete and inconsistent. Here,

- Noisy means there are errors or outliers into the dataset.
- Incomplete means there are missing values into the dataset.
- Inconsistent means there are values into dataset that shouldn't be there. That could be out of range data, that could be salary = -2000.

I've performed data cleansing to either fill up the missing values based on hypothesis or removed the row that contains the missing value. I also tried to remove the inconsistency of my dataset and errors and outliers as well. I have used data standardization to scale my data in order to remove column basis biasness from my dataset. I have also used inconsistency reduction technique to increase the integrity of my dataset. A proper processed dataset can help a model to have better understanding on dataset and results a better and reliable accuracy.

### **3.6.1. Inconsistency Reduction**

Anything that affects the data integrity results in data inconsistency. Data inconsistency problems can occur as a result of faulty input to dataset. These also known as outliers. That dataset is developed under a group. So, there is a highly possibility of having outliers in the database. Such as salary = -100, we all know that can't be possible. Inconsistent data may lead the model towards an inaccurate training accuracy. It



decreases dataset's integrity that's why I need to remove inconsistency from dataset. I have done that by going through all the data row by row in my dataset or I can visualize my data in order to find the existence of inconsistent value.

### **3.6.2. Null Value Reduction**

Null values are different from data values. Basically, a null value is an undefined value. Null values make queries, stored procedures, and views more complex. Null values can and often affect training accuracy and make model confuse. Null value could accidentally put in dataset by the contributors and that is a question of integrity, that's why I have performed null value removal operation. And this is easily can be done by Microsoft excel.

### **3.6.3. Data Standardization**

The data processing technique which converts the structure of an unbalanced datasets to a common data format known as data standardization. Data standardization transforms data from dataset after the data is pulled from the source and before it's loaded into model for training. Sometimes in dataset some features that influences the training process and makes accuracy biased. That is why I have used data standardization technique to scale all the features. That reduces the impact of particular features that influence training process.

### **3.7. Feature Selection**

Feature selection means selection of workable features from a set of features from a dataset. At the very first, when we intend to solve an issue, we collect data about that issue. That data is called raw data. Then we try to extract features from that raw data. During the feature extraction process we try to make sure that we are extracting all the

possible features that can be extracted from that dataset. Because we don't know which features are going to solve that particular issue very efficiently. If we extract features much more thinking about our issue then there is a high chance to skip any features that could have played a vital role to solve that issue. That's why we try to extract all possible features. But all the features may not need to solve our problem. And we find that using various types of mathematical algorithm. These feature selection algorithms can be seen as the combination of a search technique for proposing new feature subset. Feature selection also known as attribute selection or attribute selection. Using all the features may result overfitting. With feature selection we may acquire better accuracy from our model. Though feature selection reduces feature in a possible manner but it doesn't cause information loosing. Here in my case I have used Correlation and Chi-square to perform the selection of my features from my self-constructed dataset. I have combined correlation selected features and chi-square selected features to increase the integrity of my data and to obtain better accuracy from my model.

### **3.7.1. Correlation**

Correlation is a statistical method that measures and break down the level of connection between two variables. Correlation investigation manages the relationship between at least two variables. Correlation signifies the interdependency among the factors for corresponding two wonder, it is basic that the two marvels ought to have cause-impact relationship, & if such relationship does not exist then the two marvels cannot be associated. On the off chance that two factors shift so that development in one are joined by development in other, these factors are called circumstances and end results relationship. The correlation coefficient,  $r$ , is an outline measure that portrays the degree of the factual connection between two interim or proportion level variables. The

correlation coefficient is scaled with the goal that it is dependably between - 1 and +1. At the point when r is near 0 this implies there is little correlation between the factors and the more distant far from 0 r is, in either the positive. The degree of relationship between the variables under consideration is measure through the correlation analysis. If X is an attribute whereas Y is a target attribute, then the correlation of X & Y is

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{\left(\sum X^2 - \frac{(\sum X)^2}{N}\right)} \sqrt{\left(\sum Y^2 - \frac{(\sum Y)^2}{N}\right)}}$$

Where N is the total number of records. The correlation esteem is should between - 1 to +1. On the off chance that the connection is lower than 0 and close to - 1 then X and Y are negative related. In the event that connection esteem is close to positive 1 at that point there is a positive connection between X and Y. In the event that the correlation is 0, at that point there is no connection between X and Y.

### 3.7.2. Chi-Square

The Chi-Square test of freedom is utilized to decide whether there is a significant connection between two categorical variables. The recurrence of every classification for one ostensible variable is looked at over the classifications of the second ostensible variable. A possibility table can show the information where each line of the table speaks to a class for one factor and every segment speaks to a classification for the other variable. For instance, say a scientist needs to look at the connection between sex (male versus female) and compassion (high versus low). The chi-square trial of autonomy can be utilized to look at this relationship. The invalid speculation for this test is that there is no connection among sexual orientation and sympathy. The elective speculation is

that there is a connection among sex and compassion. If O is the observed value and E is the expected value then the chi-square is,

$$X^2 = \sum \frac{(O - E)^2}{E}$$

### **3.5 Summary**

At first, a brand-new dataset was constructed based on web application's features. Then data collection, data preprocessing and feature selection was performed then these are fed into a deep learning model. A tool has built to automatically find out SQL injection vulnerability.

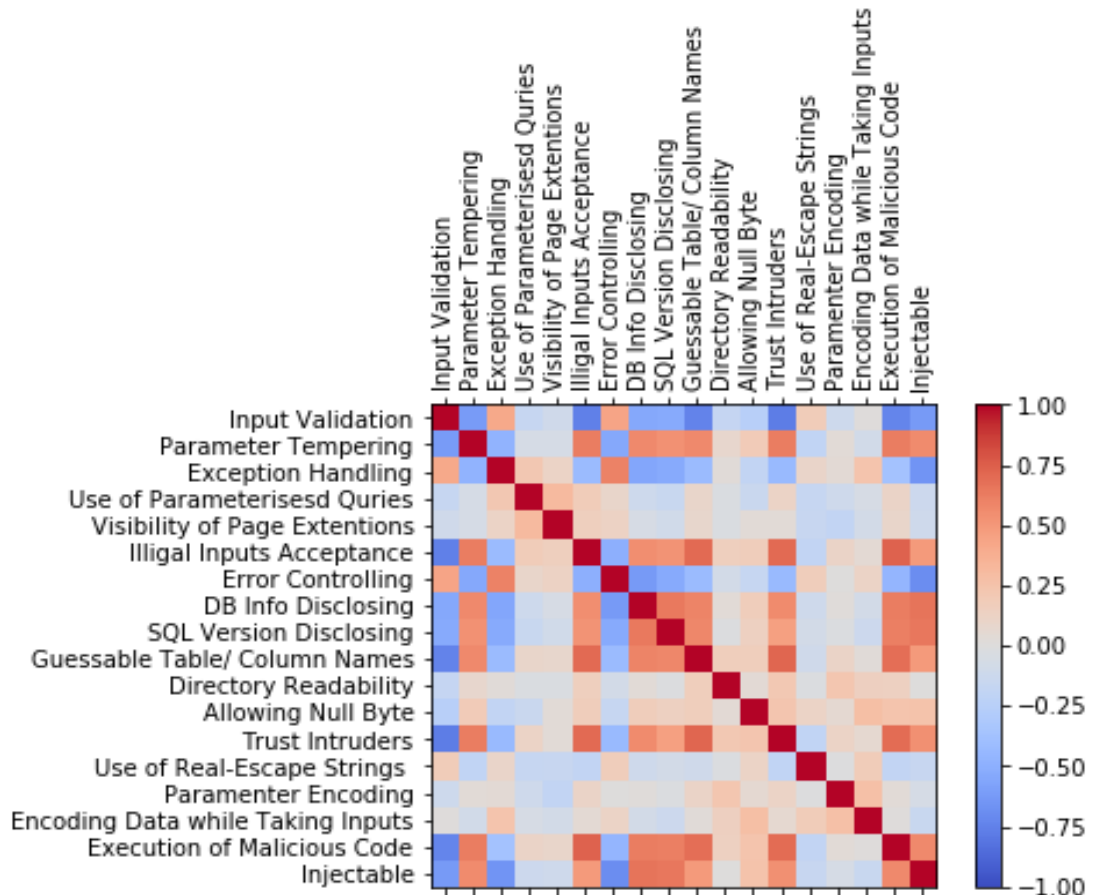
## **CHAPTER 4**

### **RESULTS AND DISCUSSION**

In this paper, there are several parts in result section such as Feature Selection (IG, Chi-Square, Correlation) result and combined result of these three features selection algorithms, Comparison decision tree and random forest algorithm for the processed dataset, elbow method result, k-means clustering result and visualization the result and finally discussion with the result.

#### **4.1 Correlation Result**

In Chapter 3, Section 3.7.1, discussed about the Correlation algorithm and show how to calculate correlation between attribute and target attribute. After applying correlation algorithm on the dataset, then found a result, which indicates the correlation among the features and reduce features which are not correlated. A correlation heatmap between features is given below:

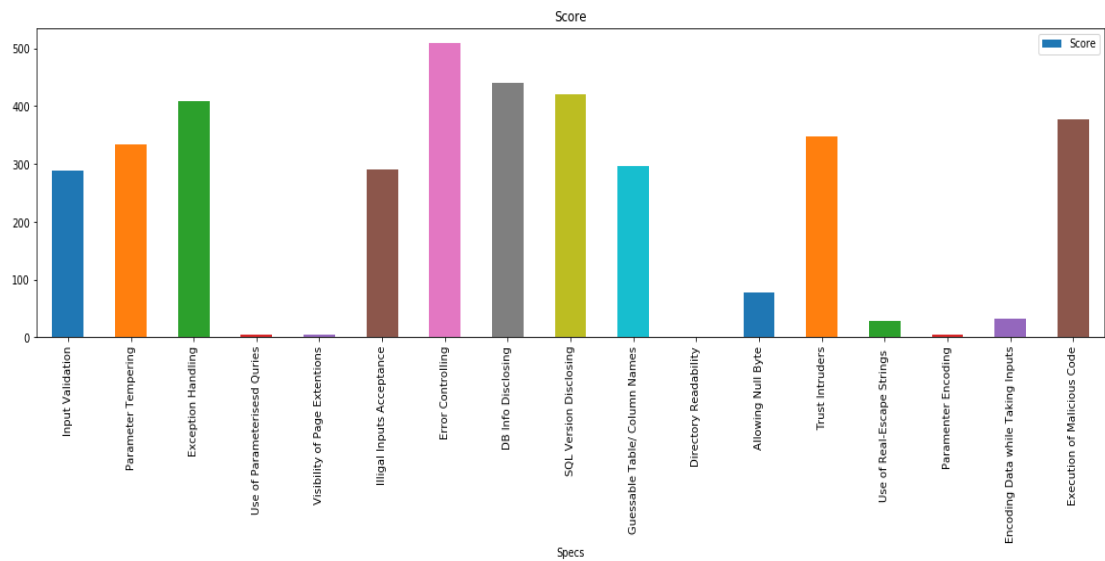


**Figure 4.1.1:** Correlation heatmap with reduced features.

That correlation heatmap shows us the correlation between reduced features. I have ran an iteration test from 0.90 to 0.55 to find out the right threshold value and better correlation. And that ended up at 0.74.

## 4.2. Chi-Square Result

In Chapter 3, Section 3.7.2, discuss about the Chi Square and how to calculate **Chi square** between target attribute. After applying **Chi Square** algorithm against the dataset, then get result list of Chi square value for each attribute. Chi Square result is shown below:



**Figure 4.2.1:** Features with Chi-Square score.

That operation was performed on the reduced features shown in Figure 4.2.1. From that we can easily understand the which can easily understand which features are significant and which features are not. I will reduce feature on the basis of this Chi-Square test. I have found out a hypothetical value by an iteration of my model from 0 to 100.

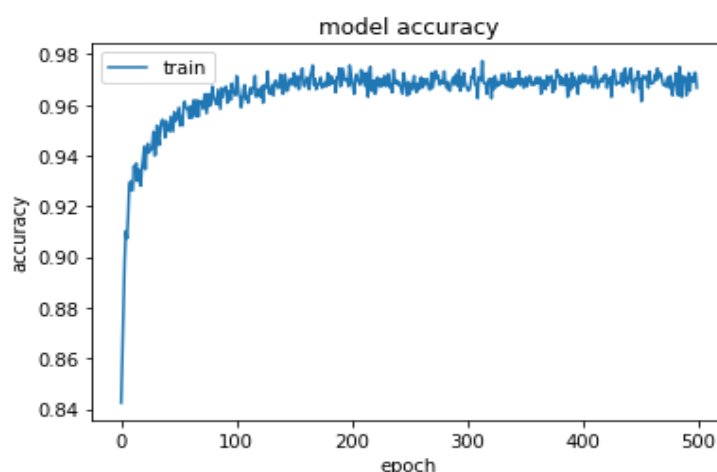
Features	Chi-Square Score	Significance
Input Validation	288.256	Significant
Parameter Tempering	334.047	Significant
Exception Handling	409.634	Significant
Use of Parameterised Queries	5.68944	Not Significant
Visibility of Page Extension	4.19805	Not Significant
Illegal Input Acceptances	290.574	Significant
Error Controlling	510.157	Significant
DB Info Disclosing	440.99	Significant
SQL Version Disclosing	420.94	Significant
Guessable Table/Column Name	297.471	Significant
Directory Readability	0.000650201	Not Significant
Allowing Null Byte	78.2139	Significant
Trust Intruders	348.714	Significant
Use of Real Escape String	28.6231	Not Significant
Parameter Encoding	5.63055	Not Significant
Encoding Data While Taking Input	33.3797	Significant
Execution of Malicious Code	377.44	Significant

**Table 4.2.2:** Significance of features based on Chi-Square Score.

I have gotten my hypothetical value 33. Features which has score less than 33 are not significant. And I have reduced them shown in Table 4.2.2.

### 4.3. Model Accuracy and Loss

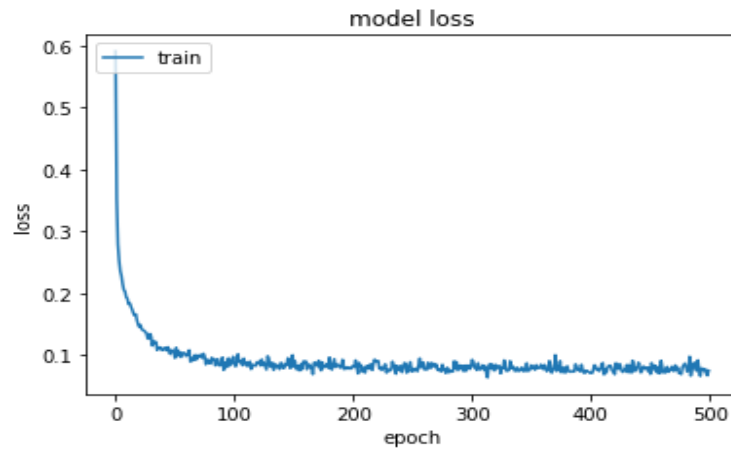
After the feature selection activities, I have fed my revised dataset to a Feed Forward Network. The prediction rate reached 98.2%. Though the web feature-based data is limited, it is very hard to check a single website, check its vulnerability and then collect information. if I could have collected more samples, the effect of training of the classification model of would be better.



**Figure 4.3.1:** Obtained accuracy of the model.

The obtained accuracy of my model is shown in Figure 4.3.1. I have split my dataset into 90% for training and the remaining 10% was held-out for the testing. Our feed forward network contains 500 epochs with batch size of 30. The network weights were randomly initialized using the default Keras initializer. Our neural network has gotten the best accuracy with one hidden layer.





**Figure 4.3.2:** Curve of loss of the model.

In Figure 4.3.2 I have shown the figure of loss of my neural network model. It tells us the distances between model's prediction from actual label. From that loss value I can determine how bad my model is. Here I have obtained a very low loss. So, I can say my model very near to the accurate.

## **CHAPTER 5**

### **CONCLUSIONS AND RECOMMENDATIONS**

#### **5.1 Findings and Contributions**

Prediction does not always define the accurate result but shows the assumption. I have developed a dataset based on web applications various types of features. Preprocessed it worked with its features in order to make it more authentic so that machine can learn more accurately from it. Data analysis is not a simple task. The real-world data is not organized. Here features are combined from two different types of feature selection algorithm. And I have obtained a good accuracy using that dataset. This result is not claimed 100% accurate, but based on statistics and data analysis that can be happened. I also have developed a tool which can tell either a web application is vulnerable or not.

#### **5.2 Recommendations for Future Works**

This analysis research is very long process. In this paper, within short time the researcher tries to cover the data analysis process as much as possible. Hence, the research topic is very importance, cause the SQL injection problem can make a huge loss to both personal and private sector. In future I will try dig into deeper of a web application to invent more efficient way to detect the attack and find an efficient solution to prevent that attack.

## REFERENCES

IBM X-Force Threat Intelligence Index 2018:  
<https://www.ibm.com/downloads/cas/MKJOL3DG>.

Open Web Application Security Project (OWASP):  
[https://www.owasp.org/index.php/Top\\_10-2017\\_Top\\_10](https://www.owasp.org/index.php/Top_10-2017_Top_10)

Randa Osman Morsi and Mona Farouk Ahmed (2019). A Two-Phase Pattern Matching Parse Tree Validation Approach for Efficient SQL Injection Attacks Detection. *Journal of Artificial Intelligence*.

Ely Salwana Mat Surin, Nurhakimah Azwani Md Najib, Chan Wei Liang, Mohd Amin Mohd Yunus, Muhammad Zainulariff Brohan and Nazri Mohd Nawawi (2019). Review of SQL Injection: Problems and Prevention. *INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION, VOL2 (2018) NO3 – 2*.

S. Nanhay, D. Mohit, R.S. Raw, and K. Suresh, “SQL Injection: Types, Methodology, Attack Queries and Prevention”, in *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016, p. 2872 – 2876.

K.G. Vamshi, V. Trinadh, S. Soundabaya, and A. Omar, “Advanced Automated SQL Injection Attacks and Defensive Mechanisms”, in *Annual Connecticut Conference on Industrial Electronics, Technology & Automation (CT-IETA)*, 2016, p. 1-6.

K. Krit and S. Chitsutha, “Machine Learning for SQL Injection Prevention on Server- Side Scripting”, in *International Computer Science and Engineering Conference (ICSEC)*, 2016, p. 1-6.

P.K. Raja and Z. Bing, “Enhanced Approach to Detection of SQL Injection Attack”, in 15th IEEE International Conference on Machine Learning and Applications (ICMLA), 2016, p. 466 – 469.

Dubey, R., & Gupta, H. (2016). SQL Filtering: An Effective Technique to Prevent SQL Injection Attack. 2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO).

Dr. Ahmad Ghafarian (2017). A Hybrid Method for Detection and Prevention of SQL Injection Attacks. 2017 Computing Conference.

Debabrata Kar and Suvasini Panigrahi (2013). Prevention of SQL Injection attack using query transformation and hashing. 2013 3rd IEEE International Advance Computing Conference (IACC).

Li Qian, Zhenyuan Zhu, lun Hu and Shuying Liu (2015). Research of SQL Injection Attack and Prevention Technology. 2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF 2015).

Romil Rawat and Shailendra Kumar Shrivastav (2012). SQL injection attack Detection using SVM. International Journal of Computer Applications (0975 – 8887). Volume 42– No.13, March 2012

D. Scott and R. Sharp, “Abstracting Application-level Web Security”, In Proceedings of the 11<sup>th</sup> International Conference on the World Wide Web (WWW 2002), Pages 396–407, 2002. Y. Huang, F. Yu, C. Hang, C. H. Tsai, D. T. Lee, and S. Y. Kuo.

V.Shanmuganeethi, Ra. Yagna Pravin, C.EmilinShyni, S.Swamynathan (2011). SQLIVD - AOP: Preventing SQL Injection Vulnerabilities using Aspect Oriented Programming. Communications in Computer and Information Science 169:327-337.

Ashish Kumar and Sumitra Binu (2018). Proposed Method for SQL Injection Detection and its Prevention. *International Journal of Engineering & Technology*, 7(2.6), 213.

Chenyu M. and Fan G., "Defending SQL injection attacks based on intention-oriented detection", 11th International Conference on Computer Science & Education (ICCSE), 2016.

Abirami J., Devakunchari R. and Valliyammai C. (2015). A top web security vulnerability SQL injection attack — Survey. 2015 Seventh International Conference on Advanced Computing (ICoAC).

Abhay K.Kolhe and Pratik Adhikari (2014). Injection, Detection, Prevention of SQL Injection Attacks. *International Journal of Computer Applications (0975 –8887)* Volume 87 –No.7, February 2014.

Voitovych O.P., Yuvkovetskyi O.S. and Kupershtein L.M. (2016). SQL injection prevention system. 2016 International Conference Radio Electronics & Info Communications (UkrMiCo).

Chen, Z., Guo, M., & zhou, L. (2018). Research on SQL injection detection technology based on SVM. Chen, Z., Guo, M., & zhou, L. (2018). Research on SQL injection detection technology based on SVM. *MATEC Web of Conferences*, 173, 01004.

Rajeh, W., & Abed, A. (2017). A novel three-tier SQLi detection and mitigation scheme for cloud environments. 2017 International Conference on Electrical Engineering and Computer Science (ICECOS).

## Appendix – A

### Parameters of Student Dropout Data Set

<b>Column Name</b>	<b>Type</b>	<b>Value</b>
Input Validation	Categorical	0/1
Parameter Tempering	Binary	0/1
Exception Handling	Binary	0/1
Use of Parameterized Queries	Binary	0/1
Visibility of Page Extension	Binary	0/1
Illegal Input Acceptance	Binary	0/1
Error Controlling	Binary	0/1
DB Info Disclosing	Binary	0/1
SQL Version Disclosing	Binary	0/1
Guessable Table/Column Name	Binary	0/1
Directory Readability	Binary	0/1
Allowing Null Byte	Binary	0/1
Proper Implementation of firewall Rules	Binary	0/1
Trust Intruders	Binary	0/1
Use of Real Escape String	Binary	0/1
Parameter Encoding	Binary	0/1
Encoding Data While Taking Input	Binary	0/1
Execution of Malicious Code	Binary	0/1
SQL Server Info	Binary	0/1
Injectable	Binary	0/1

## Appendix – B

### Pseudo code 1: Code sample of Data Standardization

```
27 #Feature scaling
28 from sklearn.preprocessing import StandardScaler
29 sc = StandardScaler()
30 x_train = sc.fit_transform(x_train)
31 x_test = sc.transform(x_test)
32
```

### Pseudo code 2: Code sample of Correlation

```
8 import numpy as np
9 import pandas as pd
10 import matplotlib.pyplot as plt
11 data = pd.read_csv('D:/SQLVD/Combined Dummy 3.csv')
12 |
13
14 def handle_correlation(dataset, threshold):
15     old_dataframe_column=len(dataset.columns)
16     new_dataset=dataset.drop('Injectable', axis=1).copy(deep=True)
17     col_corr = set() # Set of all the names of deleted columns
18     corr_matrix = new_dataset.corr()
19     print(corr_matrix)
20     for i in range(len(corr_matrix.columns)):
21         for j in range(i):
22             if (corr_matrix.iloc[i, j] >= threshold) and (corr_matrix.columns[j] not in col_corr):
23                 colname = corr_matrix.columns[i] # getting the name of column
24                 col_corr.add(colname)
25                 if colname in new_dataset.columns:
26                     del new_dataset[colname] # deleting the column from the dataset
27     new_dataframe_column=len(new_dataset.columns)
28     print(f""Old Data Frame Column Count:- {old_dataframe_column}\nNew Data Frame Column Count:- {new_dataframe_column}""")
29     return new_dataset
30
31
32 # How To apply this method
33 corr_free_data_frame = handle_correlation(dataset=data, threshold=0.75)
34 corr_free_data_frame = pd.concat([corr_free_data_frame, data.Injectable], axis=1)
35 new_data_frame = corr_free_data_frame
36
```

### Pseudo code 3: Code sample of Chi-Square

```
7 import pandas
8 import matplotlib.pyplot
9 from scipy.stats import chi2_contingency
10 from scipy.stats import chi2
11 from sklearn.feature_selection import SelectKBest
12 from sklearn.feature_selection import chi2 as sklearn_chi2
13
14 data_frame = pandas.read_csv("D:/SQLVD/corr_74.csv")
15 X = data_frame.iloc[:, :-1]
16 y = data_frame.iloc[:, -1:] #target column i.e price range
17 bestfeatures = SelectKBest(score_func=sklearn_chi2, k='all')
18 fit = bestfeatures.fit(X,y)
19
20 dfscores = pandas.DataFrame(fit.scores_)
21 dfcolumns = pandas.DataFrame(X.columns)
22 #concat two dataframes for better visualization
23 featureScores = pandas.concat([dfcolumns,dfscores],axis=1)
24 featureScores.columns = ['Specs', 'Score']
25
26 featureScores.plot(kind='bar', x='Specs', y='Score', subplots=True, figsize=(20,5))
27 matplotlib.pyplot.show()
28
29 feature_column_names=[]
30 predicted_class_name = list(data_frame.iloc[:, -1:].columns)
31 for item in featureScores[featureScores.Score>=78].iloc[:, [0]].values.tolist():
32     feature_column_names.append(item[0])
33
```

### Pseudo code 4: Code sample of Deep Learning Model

```
33 import keras
34 from keras.models import Sequential
35 from keras.layers import Dense
36 from keras.layers import Dropout
37
38 #Initializing the NN
39 classifier = Sequential()
40 #Adding input & first hidden layers to the NN.
41 classifier.add(Dense(7, kernel_initializer = 'uniform', activation = 'relu', input_dim = 13))
42 classifier.add(Dropout(rate = 0.4))
43 classifier.add(Dense(1, kernel_initializer = 'uniform', activation = 'sigmoid'))
44
45 #classifier.add(Dropout(rate = 0.1))
46 classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
47
48 history = classifier.fit(x_train, y_train, batch_size = 10, epochs = 500)
49
50 y_pred = classifier.predict_classes(x_test)
51
52
53 from sklearn.metrics import confusion_matrix
54 cm = confusion_matrix(y_test, y_pred)
55
56 score = classifier.evaluate(x_train, y_train)
57
58 print("\nTest score:", score[0])
59 print('Test accuracy:', score[1])
60
61 print("\n%s: %.2f%%" % (classifier.metrics_names[1], score[1]*100))
62
63
```