

DETECTION OF SQL INJECTION ATTACK USING ADAPTIVE DEEP FOREST

Submitted in partial fulfillment of the requirements
for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

Guntumadugu Nikhila
(Reg. No.38110182)

Kannekanti Vennela
(Reg. No.38110641)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119, TAMILNADU**

MAY – 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI– 600119

www.sathyabamauniversity.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Guntumadugu Nikhila (38110182)**, **Kannekanti Vennela (38110641)** who carried out the project entitled “**DETECTION OF SQL INJECTION ATTACK USING ADAPTIVE DEEP FOREST**” under my supervision from November 2021 to March 2022.

Internal Guide

M. S. Roobini, M.E., [Ph.D.,]

Head of the Department

Dr. L. Lakshmanan M.E., (Ph.D.)

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Guntumadugu Nikhila (Reg No: 38110182) and Kannekanti Vennela (Reg No: 38110641)** hereby declare that the Professional Training entitled “**DETECTION OF SQL INJECTION ATTACK USING ADAPTIVE DEEP FOREST**” done by me under the guidance of **M. S. Roobini, M.E., [Ph.D.,] Dept of CSE** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneshwari M.E., Ph.D., and Dr. L. Lakshmanan M.E., Ph.D.**, Heads of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **M. S. Roobini, M.E., [Ph.D.,]** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Web application attacks are incessantly increasing in number and in severity. The big data available on the internet motivates hackers to launch new kind of attacks. As announced by the OWASP organization, injection attack has been the first of the top 10 security vulnerabilities, and SQL injection attack is one of the most important types. This attack represents a serious threat to the web applications. SQL is a communication medium between Web application and back end database. So the mostly attackers use SQL for accessing a database. SQL injection can cause great harm to the network, resulting in data leakage and website paralysis.

Injection attack is one of the best 10 security dangers declared by OWASP. SQL infusion is one of the main types of attack. In light of their assorted and quick nature, SQL injection can detrimentally affect the line, prompting broken and public data on the site. Therefore, this article presents a profound woodland-based technique for recognizing complex SQL attacks. Research shows that the methodology we use resolves the issue of expanding and debasing the first condition of the woodland. We are currently presenting the AdaBoost profound timberland-based calculation, which utilizes a blunder level to refresh the heaviness of everything in the classification.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	vi
	LIST OF FIGURES	ix
1	INTRODUCTION:	1
	1.1 OVERVIEW OF THE PROJECT	1
	1.2 OBJECTIVE OF THE PROJECT	2
2	LITERATURE SURVEY:	3
	2.1 RELATED WORK	3
3	AIM AND SCOPE OF PRESENT IMPLEMENTATION:	6
	3.1 AIM AND SCOPE	6
	3.2 EXISTING SYSTEM	6
	3.2.1 DISADVANTAGES OF EXISTING SYSTEM	6
	3.3 PROPOSED SYSTEM	6
	3.3.1 ADVANTAGES OF PROPOSED SYSTEM	7

	3.4 ARCHITECTURE	7
	3.5 REQUIREMENTS	8
	3.5.1 HARDWARE REQUIREMENTS	8
	3.5.2 SOFTWARE REQUIREMENTS	8
	3.5.3 LIBRARIES REQUIRED	8
	3.6 SAMPLE INPUT	8
4	METHODOLOGY:	9
	4.1 SCEMATIC PROCESS	9
	4.2 MODEL IMPLEMENTATION	9
	4.3 ADAPTIVE DEEP FOREST	10
5	RESULT AND DISCUSSION PERFORMANCE ANALYSIS:	12
	5.1 RESULT ANALYSIS	12
6	SUMMARY AND CONCLUSIONS:	15
	6.1 CONCLUSION	15
	6.2 FUTURE WORK	15
	REFERENCES	16

APPENDIX	17
A. SOURCE CODE	17
B. SCREEN SHOTS	21

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
3.4	Architecture	7
3.6	Dataset	8
4.2	Model Implementation	10
B.1	Graphical Representation of Data	21
B.2	Frequency Distribution Graph	22
B.3	Accuracy score of Algorithms	22
B.4	AdaBoost Predictions	23
B.5	Categorical Graph	24
B.6	Hyper Parameters Tuning	24

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT:

With the rapid development of Web 2.0 technology, network applications have gradually become an indispensable part of our lives. At the same time, Web applications are confronted with more challenges. As announced by the OWASP organization, injection attack has been the first of the top 10 security vulnerabilities in 2013 and 2017, and SQL injection attack is one of the most important types

SQL (Structured Query Language) is a standard database programming language for accessing and manipulating data in database WEB programming languages (such as PHP, JAVA) provide various methods for constructing and executing SQL statements. Developers usually construct SQL statements by concatenating string which is submitted by users from web page. Due to the wide variety of SQL languages, there are too many encoding methods for constructing SQL statements, so there is a risk of being attacked anywhere via constructing SQL statements. In the process of SQL injection attack, the attacker inserts malicious code fragments into the request parameters, which causes the server to execute illegal queries, resulting in data leakage and database damage. For example, the attacker can obtain the username, password and other privacy data of website users by SQL injection, which seriously threatens the data security. As shown in, the attacker sends malicious SQL statements into the application through methods like user input injection, cookie injection, server variable injection and so on. Since SQL attacks only need to construct structured query statements without developing additional malwares, SQL injection has become one of the most frequently used attacks by attackers, and the attacking methods change frequently with the development of website technology. The structure of SQL language is changeable and supports different coding methods. Therefore, some traditional detection methods, such as blacklist mechanism and rule-based detection mechanism, cannot produce better defensive effect.

In recent years, many researchers have done a lot of work on SQL injection detection, but the detection scope is usually limited to some subsets of SQL injection.

1.2 OBJECTIVE OF THE PROJECT:

Attackers can use SQL injection vulnerabilities to bypass application security measures. They can go around authentication and authorization of a web page or web application and retrieve the content of the entire SQL database. They also use SQL injection to add, delete, and modify records in databases. The main objective of the project is to detect the SQL injection attacks.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK:

Konstantinos Kemalis et al, he acquainted another methodology with characterizing the utilization of SQL weaknesses. This new technique, from one perspective, utilizes specialized measurements that portray the idea of the SQL issue made and carried out by the Internet, then again, which controls the most common way of making an issue that is actually untrustworthy. They have fostered a SQL injection detection system (SQL-IDS) model that carries out the calculation. The framework screens Java-based applications and distinguishes SQL pressure attacks continuously. It reports the aftereffects of a progression of SQL attacks, which show that understanding the normal issues keeps the framework from playing out a top to bottom, incorporated examination that doesn't offer a positive or negative response [1]. V. Naga Manikanta et al, he examined how to appropriately shield the web application from SQL attacks utilizing a firewall strategy. That technique utilizes a hearty and vigorous investigation. In the inside and out examination, he depicted the URL, the structure, the infusion site, and the basic site highlights. Subsequently, we know about the genuine issues that might emerge in the materials. In a hearty examination, he tracked the issues experienced during activity utilizing the firewall archive and contrasted them with the white rundown of issues. The trial of utilizing a genuine website with two open-source apparatuses: Web Request and Web Control Framework (w3af) and GreenSQL. He utilized w3af for a rundown of all GreenSQL-agreeable issues as a vault fire. The outcomes showed that the execution strategy doesn't need changing any code on a normal site, however it can recognize a wide range of SQL attacks utilizing extra intermediary facilitating highlights [2]. Nabeel Salih et al proposed the SQLI Protocol (WASP) instrument for web apparatuses is being created for ongoing web applications to recognize SQL attacks in the way portrayed in this article. Planned materials will then, at that point, be explored and broken down for better execution. The procedure depends on constant diagnostics dependent on the information on foreign substances, effective and proficient dispersion, and linguistic structure in line with distinguishing invalid issues prior to arriving at the base utilizing Microsoft ASP.NET. The high-level apparatus can

distinguish and shut down all SQLI attacks progressively, bringing about no blunders, barely any benefits, and no base prerequisites for arrangement [3]. Hasanein D. Rjeib et al explained that web applications for large numbers of our online movement programs have been added to the web-based help. In any case, disregarding the protection and security prerequisites of organizations makes them more powerful as far as security. The security framework proposed in this article incorporates three methods for forestalling it: how to store put away things, explicit mistake messages, and how to store them on a Structured Query Language Injection Attacks (SQLIA). It likewise breaks down the arranged inoculation process. The normal assurance system is great since it can adequately forestall SQLIA information base attacks [4]. Abhishek Kumar Baranwal et al, in the research explained very much planned needle permits admittance to oppressive or unapproved clients and are principally accomplished through SQL infusion and Cross site prearranging. In that article, he did a point-by-point investigation of the various kinds of SQL and XSS attacks, how to recognize and forestall them, the future, and what we have seen and expounded on the most common way of battling these attacks. They also, we will make a similar examination of the various strategies used to battle these attacks. We will then, at that point, show what we have seen and expound on the future and arrange advancement against these attacks [5]. Geogiana Buja et al, explained in the research that SQL is extremely simple to ensure against attacks, and there are as yet countless frameworks on the Internet that can be powerless against this sort of attack, as there will be a couple of basic things that are not known. Hence, in this paper, we need to take an example to comprehend and recognize the shortcomings of the site; SQL input depends on a characterized, characterized definition. Likewise, search inquiries can report weaknesses in web applications. Subsequently, the arrangement model ought to have the option to limit the chance of SQL Injection attacks on the Internet [6]. Huafeng Zhang et al, in the research showed progress in SQL-based learning. Not all substances should be investigated and erased. You simply must be more segregating regarding how you help others. Joining these components into a pre-arranged format can make the continuous SQL idea clearer. That utilizes inside and out examination to distinguish SQL attacks in power. We select the objective ascribes dependent on the SQL attack and afterward take the

necessities in the URL or bundle as train data; Train chosen protests and amassed information utilizing deep belief networks (DBNs) and afterward figure out how to infuse SQL Injection. Then, track down a genuine instance of understanding SQL and quit wasting time progressively [7]. Vinaya Kumar et al, in the research explained that Intrusion Detection System (IDS) is developed widely by using the machine learning techniques for the detection and classification of attacks like cyber-attacks in the network and host levels in an instinctive definite manner. One of the type deep learning models, DEEP NEURAL NETWORK (DNN) is used for developing productive IDS for detection and classification of uncertain and incalculable cyber-attacks [8]. Dimitris Mitropoulos et al proposed research on different security mechanisms at odds with SQL injection attacks. He proposed a model which highlights the main key debility of the attacks. They categorized and then analyzed the previous proposed systems and methods based on the performance, availability and some other characteristics [9]. Yaohui Wang et al, in the research proposed a SQL injection detection method picking out PHP, which is the basis of injection analysis automation. This process is the combination of analysis technology, like behavior model and lexical objects comparison based SQL for the design and the establishment of a system for the detection of the SQL injections [10].

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 AIM AND SCOPE:

The aim of the project is to detect the SQL injection attacks. This method uses the SQL lexical and grammatical analysis technology to parse the SQL statement into the SQL grammar tree via stain analysis, and judges whether there is an SQL injection attack according to the stain state. While, dynamic analysis detection can only detect pre-defined vulnerabilities by application developers, and it has no detection capability.

When encountering attacks against unknown vulnerabilities. Parameter filtering is based on regular expressions and blacklists to filter invalid characters . This method needs predefined regular expressions, but due to the diversity of SQL grammar and user input, it cannot meet the detection requirements. Attackers can adjust the input to bypass detection. Appiah et al. proposed an improved pattern matching method for signature-based SQL injection attack detection framework , which integrates fingerprint identification and pattern matching to distinguish normal and malicious SQL queries.

3.2 EXISTING SYSTEM:

In the existing application, the malicious attacks are controlled by certain decisions and programming coding. The decision is based on a certain set of rules. These sets of rules are designed to detect attack patterns. Other Approaches are White-box testing, Black-box testing, Static analysis, Model based testing.

3.2.1 DISADVANTAGES OF EXISTING SYSTEM:

The drawbacks of the existing system are, for huge applications, the set of rules becomes more complex. Test cases are challenging to design without having clear functional specifications. It is more difficult to maintain the set of rules manually and test.

3.3 PROPOSED SYSTEM:

An appreciation model can highlight shortcomings in web applications. Accordingly, this model can lessen the possibilities that SQLIA will deal with your site. AI and the SVM calculation are utilized to ensure SQLIA control. The answer for this is to distinguish and keep SQLIA from closing down when you move the landing page of every application to the test page. Straight and straight areas were

made by sharing SVM segments similarly. SVM, however much as could be expected, depends on greatest level and line separation, yet doesn't address the connection between qualities. The fundamental thought of the SVM is to utilize this strategy, that is, to pick a scaffold plane and an enormous side.

The preparation is utilized to plan a site examination utilizing the preparation level. The attack signature has the SQLIA token organization and the SQLIA character perfectly positioned for the infusion, however the solicitation of the authority site appears as the data the program is sitting tight for. Learning vector data is characterized as a lattice or jargon of catchphrases (SQLIA negative) and SQL token (SQLIA positive). The design of the SQL language is changing. The SVM will show this negative articulation assuming that the client welcomes new strategies to match at any rate.

3.3.1 ADVANTAGES OF PROPOSED SYSTEM:

The advantages of the proposed system are, for huge applications, the set of rules becomes easier and simple. Test cases are simple to design with having clear functional specifications. It is more effective and easy to maintain the set of rules manually and test. The proposed system gives the best accuracy score.

3.4 ARCHITECTURE:

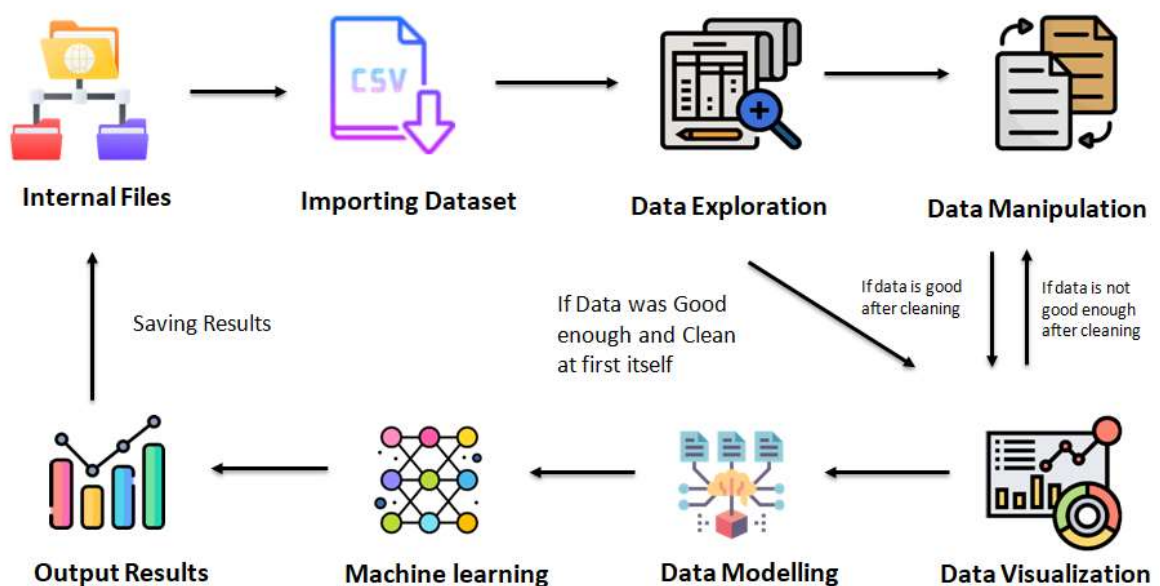


Fig.no.3.4: ARCHITECTURE

3.5 REQUIREMENTS:

3.5.1 HARDWARE REQUIREMENTS:

- 1TB hard disk
- 2Gb RAM

3.5.2 SOFTWARE REQUIREMENTS:

- Windows
- Jupyter Notebook
- Excel
- python-3.9.5

3.5.3 LIBRARIES REQUIRED:

- Exploring and Cleaning: Numpy, Pandas
- Visualization: Matplotlib, Seaborn,
- Model Building: Scikit-Learn

3.6 SAMPLE INPUT:

System takes an input from an existing arbitrary sqlinjection dataset and detects the SQL injection attacks using AdaBoost predictions. The SQLInjection dataset contains six dimensions with attributes such as, SQLError, UnionKeyword, EqualCondition, QueryWait, FeaturesEnabled and Label

df						
	SQLError	UnionKeyword	EqualCondition	QueryWait	FeaturesEnabled	Label
0	1	1	1	1	1	1
1	1	1	1	0	0	1
2	1	1	1	0	0	1
3	1	1	1	0	1	1
4	1	0	0	0	0	1
...	...	...	...	...	...	...
393	0	1	0	1	0	0
394	1	0	1	1	0	0
395	0	0	0	0	0	0
396	1	0	0	1	0	0
397	0	0	0	0	0	0

398 rows × 6 columns

Fig.no.3.6: DATASET

CHAPTER 4

METHODOLOGY

4.1 SCHEMATIC PROCESS:

To distinguish SQL attacks. In the first place, we work on the idea of profound timberland. The consideration of each stage is predictable with the normal of the outcomes, which can more readily resolve the issue of natural corruption brought about by deforestation. We are presently presenting the AdaBoost calculation and its capacity to adjust to profound woodland conditions, which can utilize the blunder level of every profound timberland level to work on the seriousness of the circumstance.

4.2 MODEL IMPLEMENTATION:

Data Collection

There are several techniques to collect the data, like web scraping, manual interventions and etc. An existing SQLInjection dataset is taken Kaggle for the detection of SQL Injection attacks

Data Importing

Import the Dataset and necessary libraries like numpy (for linear algebra), pandas (for data processing), matplotlib (for data visualization) and etc

Data Exploration

Data exploration is an approach similar to initial data analysis, used for visual exploration to understand what is in a dataset and the characteristics of the data

Data Manipulation

Data manipulation refers to the process of adjusting data to make it organized and easier to read or be easier to process. The python pandas library is an open source project that provides a variety of easy to use tools for data manipulation and analysis

Data Visualization

Data visualization is the graphical representation of information and data in a pictorial or graphical format.

Data Modelling

The process of modeling is training a machine learning algorithm to predict the

labels from the features, and validating it on holdout data. The output from modeling is a trained model that can be used for inference, making predictions on new data points

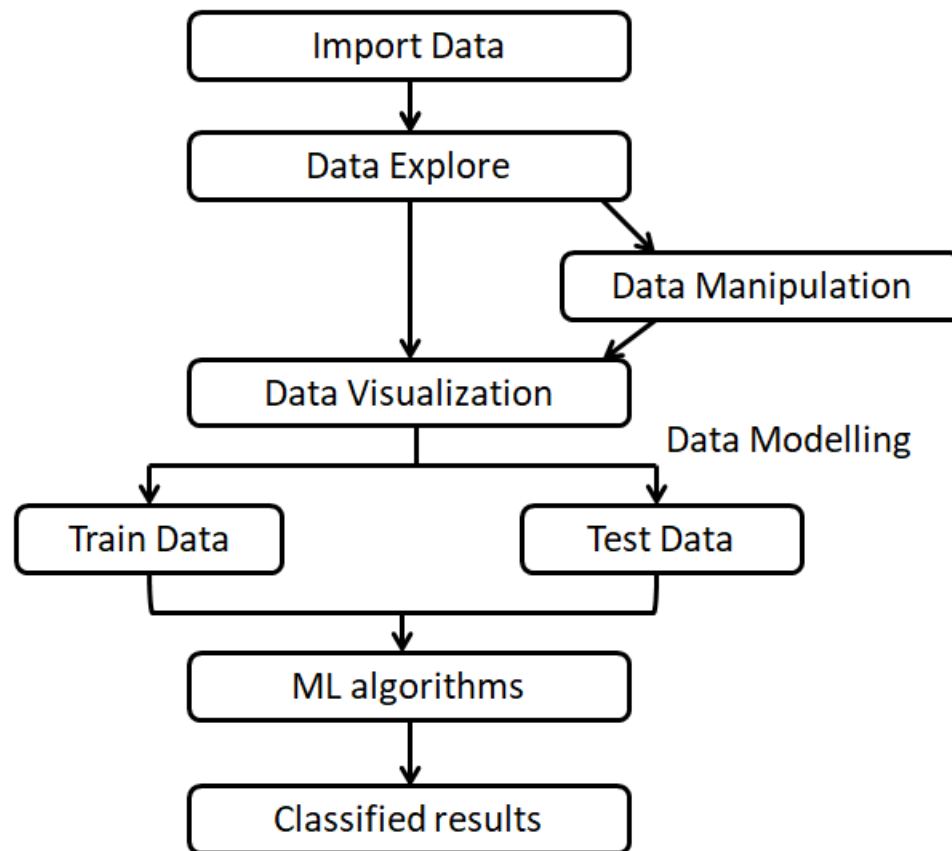


Fig.no.4.2: MODEL IMPLEMENTATION

4.3 ADAPTIVE DEEP FOREST:

Here, an Adaptive deep forest model (ADF) is proposed that is built on deep forest. In ADF, feature transformation enhances the feature performance based on the multi-grained scanning, and then by using cascade structure, layer by layer the characterization learning is performed. In hyper-parameter settings and performance, Adaptive deep forest has obvious advantages as compared to deep neural networks. Firstly, adaptive deep forest has much less hyperparameters compared to deep neural networks, so for hyper-parameter setting, it is more. And secondly, adaptive deep forest can achieve excellent performance in the default hyperparameter setting, it has higher efficiency when dealing with different data in different fields and the parallel computing.

Sliding windows are used to scan the raw features by Multi-grained. All feature vectors that are extracted from the positive/negative samples are always treated

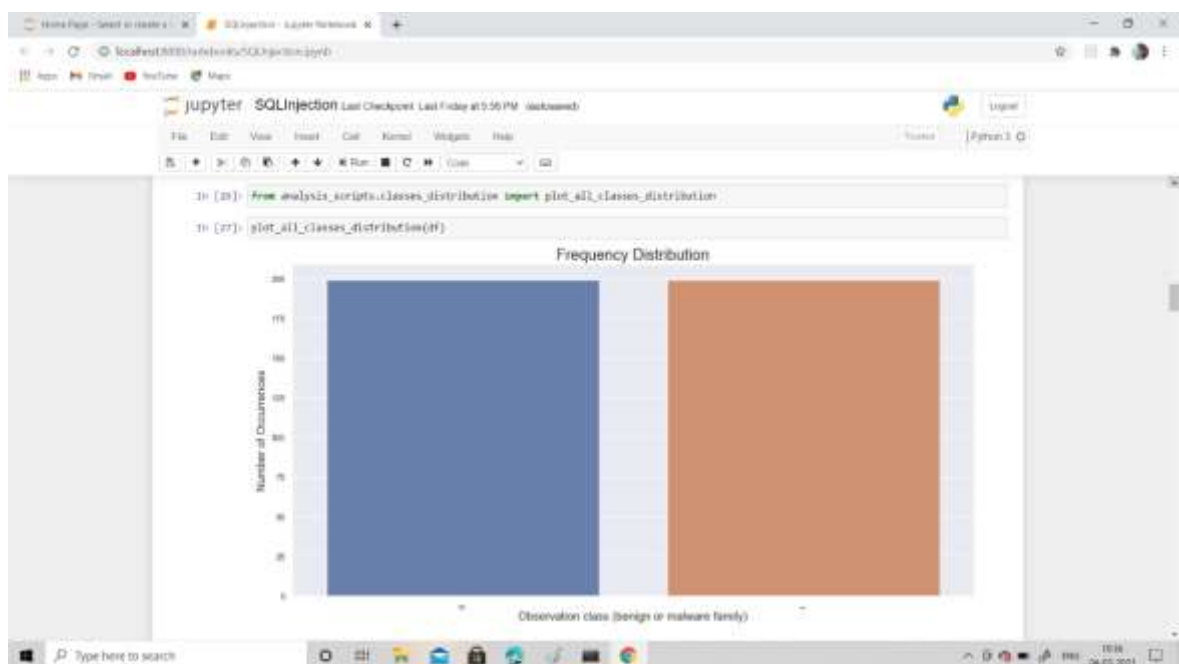
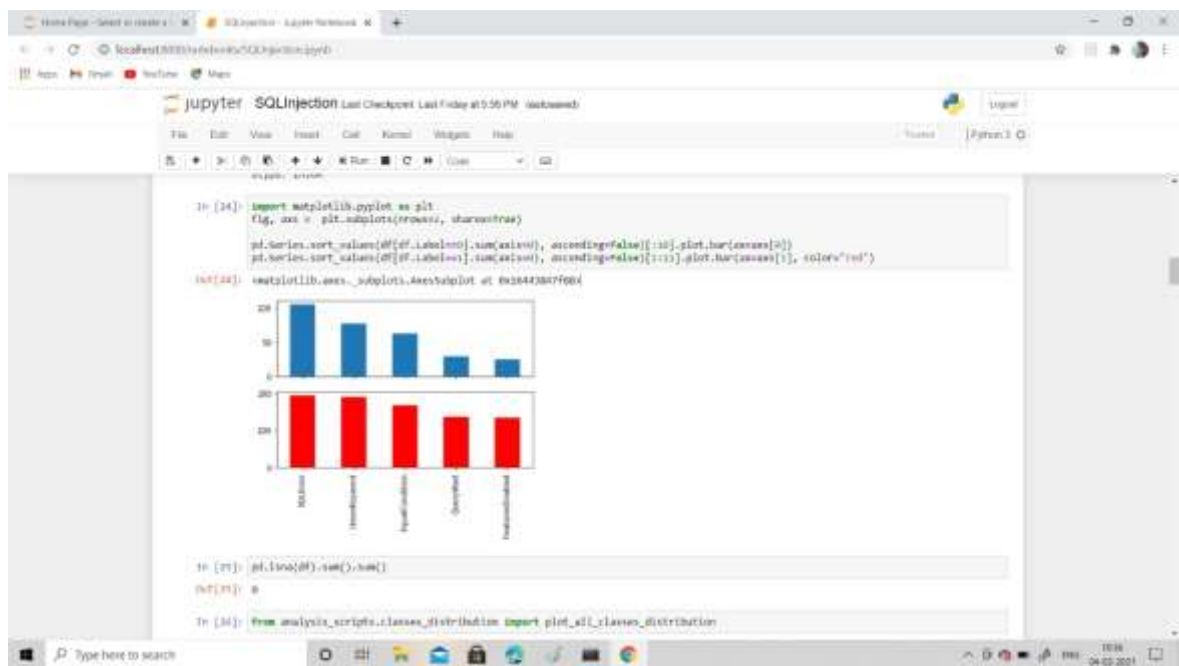
as positive/negative instances. To train a random forest and a completely random forest, the instances that are extracted from the same size window are always used. And then the vectors generated by these forests are aggregated into transformed feature vectors of the original input features. After adding a new layer, it should be noted that the performance of the entire cascade will be evaluated on the validation dataset, and the training process will be terminated, if there is no significant performance gain.

CHAPTER 5

RESULT AND DISCUSSION PERFORMANCE ANALYSIS

5.1 RESULT ANALYSIS:

The graphical representation of the features of the dataset when the label values are 1 and 0. The blue color bars represent when the label value is 0 and the red color bars represent when the label value is 1. The frequency distribution graph of the 0 and 1 value in the label feature. From the figure, the label consists of an equal number of 0 and 1 values.



Gaussian Naïve Bayes (GaussianNB) is used in implementing the Gaussian Naïve Bayes algorithm for the classification. From Fig, the accuracy of the GaussianNB classifier is 0.9 (90). It is an accuracy test of the KNeighbors classifier with an accuracy of 0.875 (87.5).

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
pred = gnb.predict(X_test)
accuracy = accuracy_score(pred, y_test)
print(accuracy)
print(classification_report(pred, y_test, labels=None))
```

```
0.9
```

	precision	recall	f1-score	support
0	0.86	0.91	0.88	33
1	0.93	0.89	0.91	47
accuracy			0.90	80
macro avg	0.90	0.90	0.90	80
weighted avg	0.90	0.90	0.90	80

```
i = 15
neigh = KNeighborsClassifier(n_neighbors=i)
neigh.fit(X_train, y_train)
pred = neigh.predict(X_test)
# accuracy
accuracy = accuracy_score(pred, y_test)
print("kneighbors {}".format(i))
print(accuracy)
print(classification_report(pred, y_test, labels=None))
print("")
```

```
kneighbors 15
0.875
```

	precision	recall	f1-score	support
0	0.80	0.90	0.85	31
1	0.93	0.86	0.89	49
accuracy			0.88	80
macro avg	0.87	0.88	0.87	80
weighted avg	0.88	0.88	0.88	80

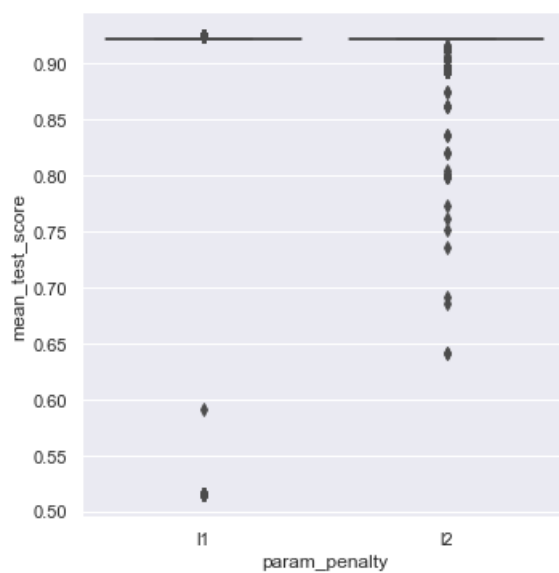
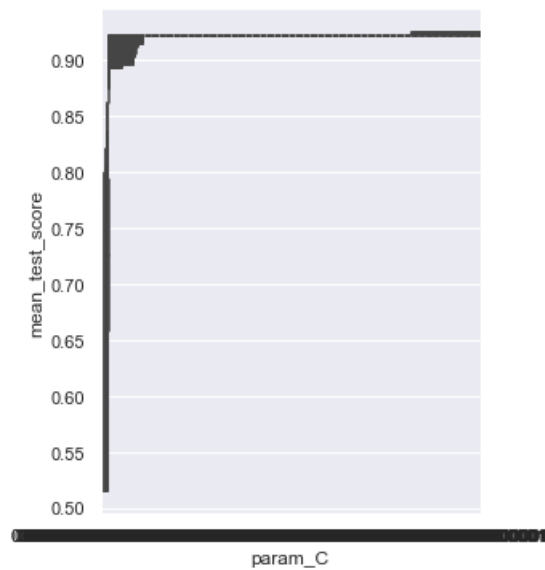
The AdaBoost classifier class implements the AdaBoost algorithm. It is a boosting classifier that combines the different types of multiple classifiers for increasing the accuracy of the classifiers.

```
ada=AdaBoostClassifier()
ada.fit(X_train, y_train)
adapred= ada.predict(X_test)
accuracy = accuracy_score(adapred, y_test)
print(ada)
print(accuracy)
print(classification_report(adapred, y_test, labels=None))
```

```
AdaBoostClassifier()
0.9
```

	precision	recall	f1-score	support
0	0.83	0.94	0.88	31
1	0.96	0.88	0.91	49
accuracy			0.90	80
macro avg	0.89	0.91	0.90	80
weighted avg	0.91	0.90	0.90	80

The categorical graph of the parameters of the data. Hyper parameters tuning is done with a Random forest model and the test set accuracy is 0.9 (90).



```
print('Best Model Hyper Parameters: {}'.format(cv.best_params_))
this_model = 'random forest'
model_comparison_dict[this_model] = cv.best_estimator_.score(X_test, y_test)
print('Test set accuracy = {}'.format(model_comparison_dict[this_model]))
```

Best Model Hyper Parameters: {'C': 0.8191, 'penalty': 'l1'}

Test set accuracy = 0.9

CHAPTER 6

SUMMARY AND CONCLUSIONS

6.1 CONCLUSION:

- Detection of SQL injection attack in complex network environment is a hotspot in the field of network security
- Methods based on Machine Learning have multiple hyper parameters and it's easy to cause over-fitting
- We proposed the SQL infusion strategy dependent on profound woods conditions. Utilizing ADF, the example size can be changed straightforwardly during the preparation.
- By using AdaBoost and other ensemble algorithms we had successfully predicted with best accuracy of 90%

6.2 FUTURE WORK:

There is still opportunity to get better in the WAPS-CIVS counteraction framework. SQL secures the WAPS-CIVS needle and doesn't resolve the needle through the storehouse. The saved methodology is the standard method for utilizing the base. Sees SQL proclamations made on the WAPSCIVS program/application web application. The XPath data set is like SQL injection, and has since been called XML, which involves information as a web application data set. WAPS-CIVS addresses just XPath-based examples in web applications. This doesn't resolve the issue of combining XQuery with XQuery Unity. XPath pin insurance can be improved by taking care of non-repetitive issues. The need to interface counterfeit records called CSRF or XSRF is notable, however it is like a Cross Site (XSS) attack. By coordinating the CSRF avoidance framework, the WAPS-CIVS counteraction framework documentation framework can be additionally evolved.

REFERENCES

- [1] Konstantinos Kemalis, Theodoros Tzouramanis, "SQL-IDS: A specification-based approach for SQL-injection detection", 2008
- [2] V. Naga Manikanta, Anjali sardana, "Protecting web applications from SQL injection attacks by using framework and database firewall", ICACCI '12: Proceedings of the international Conference on advances in Computing, Communications and Informatics august 2012 Pages 609-613 <https://doi.org/10.1145/2345396.2345495>
- [3] Nabeel Salih, A. Samad, "Protection Web Applications using Real-Time Technique to Detect Structured Query Language Injection Attacks", September 2016, International journal of computer applications 149(6):26-32
- [4] Hasanein D. Rjeib, Haider Alshargi, "Multi mechanism approach for preventing SQL injection attacks in stored procedure", January 2018
- [5] Abhishek Kumar Baranwal, "Approaches to detect SQL injection and XSS in web applications", EECE 571B, TERM SURVEY PAPER, APRIL 2012
- [6] Geogiana Buja, Kumarularifin Bin Abd Jalil, "Detection model for SQL injection attack: An approach for preventing a web application from the SQL injection attack", April 2014
- [7] Huafeng Zhang, Boa Zhao, "SQL Injection Detection Based on Deep Belief Network", CSAE 2019: Proceedings of the 3rd International Conference on Computer Science and Application Engineering October 2019 Article No.: 20Pages 1–6 <https://doi.org/10.1145/3331453.3361280>
- [8] Vinayakumar, R., Alazab, Mamoun, Soman, K. P., Poornachandran, Prabakaran, Al-Nemrat, A. and Venkatraman, Sitalakshmi 2019. Deep Learning Approach for Intelligent Intrusion Detection System. IEEE Access. 7, pp. 41525-41550. <https://doi.org/10.1109/ACCESS.2019.2895334>
- [9] Dimitris Mitropoulos, Panos Louridas, Michalis Polychronakis, "Defending Against Web Application Attacks: Approaches, Challenges and Implications", March 2017, Volume: 16 Issue: 2
- [10] Yaohui Wang, Wenbing Zhao, Yuan Liu, "Detecting SQL Vulnerability Attack based on the Dynamic and Static Analysis Technology", 2015 IEEE 39th Annual Computer Software and Applications Conference

APPENDIX

A. SOURCE CODE:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree
import pandas as pd
import numpy as np
import seaborn as sns
df = pd.read_csv("sqlinjection.csv")
df
df.head()
df.tail()
df.shape
df.size
df.isnull()
df.isnull().sum()
df = df.astype("int64")
df.Label.value_counts()
pd.Series.sort_values(df[df.Label==1].sum(axis=0), ascending=False)[1:11]
pd.Series.sort_values(df[df.Label==0].sum(axis=0), ascending=False)[:10]
import matplotlib.pyplot as plt
fig, axs = plt.subplots(nrows=2, sharex=True)
pd.Series.sort_values(df[df.Label==0].sum(axis=0),
ascending=False)[:10].plot.bar(ax=axs[0])
pd.Series.sort_values(df[df.Label==1].sum(axis=0),
ascending=False)[1:11].plot.bar(ax=axs[1], color="red")
from analysis_scripts.classes_distribution import plot_all_classes_distribution
plot_all_classes_distribution(df)
X=df.iloc[:, :-1]
```

```

y=df.iloc[:,-1]
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.20,random_state=0)
print('Training Observations: {}'.format(X_train.shape[0]))
print('Testing Observations: {}'.format(X_test.shape[0]), end='\n\n')
print('TRAINING OBSERVATIONS:')
print('{} Non-SQL Injection observations \n{} SQL Injection observations'
      .format(np.bincount(y_train)[0], np.bincount(y_train)[1]), end='\n\n')
print('TESTING OBSERVATIONS:')
print('{} Non-SQL Injection observations \n{} SQL Injection observations'
      .format(np.bincount(y_test)[0], np.bincount(y_test)[1]))

gnb = GaussianNB()
gnb.fit(X_train, y_train)
pred = gnb.predict(X_test)
accuracy = accuracy_score(pred, y_test)
print(accuracy)
print(classification_report(pred, y_test, labels=None))

i = 3
neigh = KNeighborsClassifier(n_neighbors=i)
neigh.fit(X_train, y_train)
pred = neigh.predict(X_test)
    # accuracy
accuracy = accuracy_score(pred, y_test)
print("kneighbors {}".format(i))
print(accuracy)
print(classification_report(pred, y_test, labels=None))
print("")

i = 6
neigh = KNeighborsClassifier(n_neighbors=i)
neigh.fit(X_train, y_train)
pred = neigh.predict(X_test)
    # accuracy
accuracy = accuracy_score(pred, y_test)

```

```

print("kneighbors {}".format(i))
print(accuracy)
print(classification_report(pred, y_test, labels=None))
print("")
i = 12
neigh = KNeighborsClassifier(n_neighbors=i)
neigh.fit(X_train, y_train)
pred = neigh.predict(X_test)
    # accuracy
accuracy = accuracy_score(pred, y_test)
print("kneighbors {}".format(i))
print(accuracy)
print(classification_report(pred, y_test, labels=None))
print("")
i = 15
neigh = KNeighborsClassifier(n_neighbors=i)
neigh.fit(X_train, y_train)
pred = neigh.predict(X_test)
    # accuracy
accuracy = accuracy_score(pred, y_test)
print("kneighbors {}".format(i))
print(accuracy)
print(classification_report(pred, y_test, labels=None))
print("")
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy = accuracy_score(pred, y_test)
print(clf)
print(accuracy)
print(classification_report(pred, y_test, labels=None))
ada=AdaBoostClassifier()
ada.fit(X_train, y_train)

```

```

adapred= ada.predict(X_test)
accuracy = accuracy_score(adapred, y_test)
print(ada)
print(accuracy)
print(classification_report(adapred, y_test, labels=None))
adaboost_pred = pd.DataFrame(adapred, columns = ['adaboost predictions'])
adaboost_pred['adaboost predictions'] = adaboost_pred['adaboost
predictions'].apply({1:'yes',0:'no'}.get)
adaboost_pred
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, train_test_split
import datetime
model_comparison_dict = {}
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import make_scorer, accuracy_score
from sklearn.model_selection import GridSearchCV
clf = RandomForestClassifier()
parameters = {'n_estimators': [4, 6, 9],
              'max_features': ['log2', 'sqrt','auto'],
              'criterion': ['entropy', 'gini'],
              'max_depth': [2, 3, 5, 10],
              'min_samples_split': [2, 3, 5],
              'min_samples_leaf': [1,5,8]
}
acc_scorer = make_scorer(accuracy_score)
s = datetime.datetime.now()
cls = LogisticRegression(random_state=123, solver='liblinear')

params = {'C': np.arange(.0001, 1, .001),
          'penalty': ['l1', 'l2']}

cv = GridSearchCV(cls, params,cv=5,refit=True,return_train_score=True)

```

```

cv = cv.fit(X_train, y_train)
print('train time {}'.format(datetime.datetime.now() - s))
s = pd.DataFrame.from_dict(cv.cv_results_)
s.columns
for p in params.keys():
    sns.catplot('param_' + p, 'mean_test_score', data=s, kind='box')
print('Best Model Hyper Parameters: {}'.format(cv.best_params_))
this_model = 'random forest'
model_comparison_dict[this_model] = cv.best_estimator_.score(X_test, y_test)
print('Test set accuracy = {}'.format(model_comparison_dict[this_model]))

```

B.SCREEN SHOTS:

SOURCE CODE AND OUTPUTS:

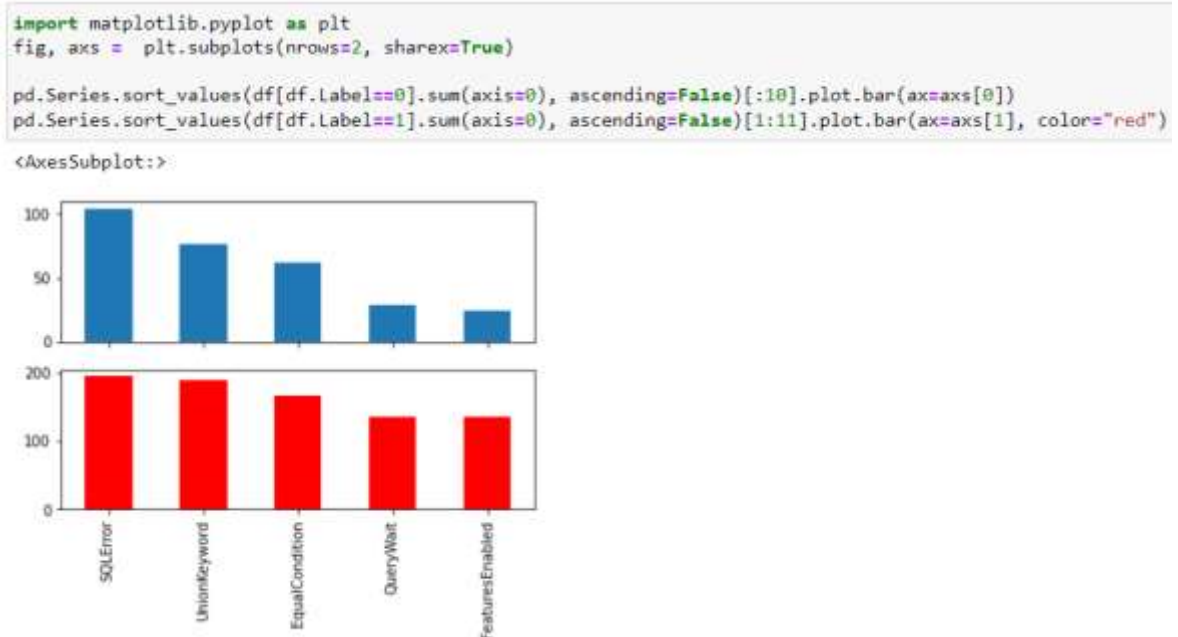


Fig.no.B.1: GRAPHICAL REPRESENTATION OF DATA

```
from analysis_scripts.classes_distribution import plot_all_classes_distribution
plot_all_classes_distribution(df)

C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword args:
x, y. From version 0.12, the only valid positional argument will be 'data', and passing other arguments without an explicit key
word will result in an error or misinterpretation.
warnings.warn(
```

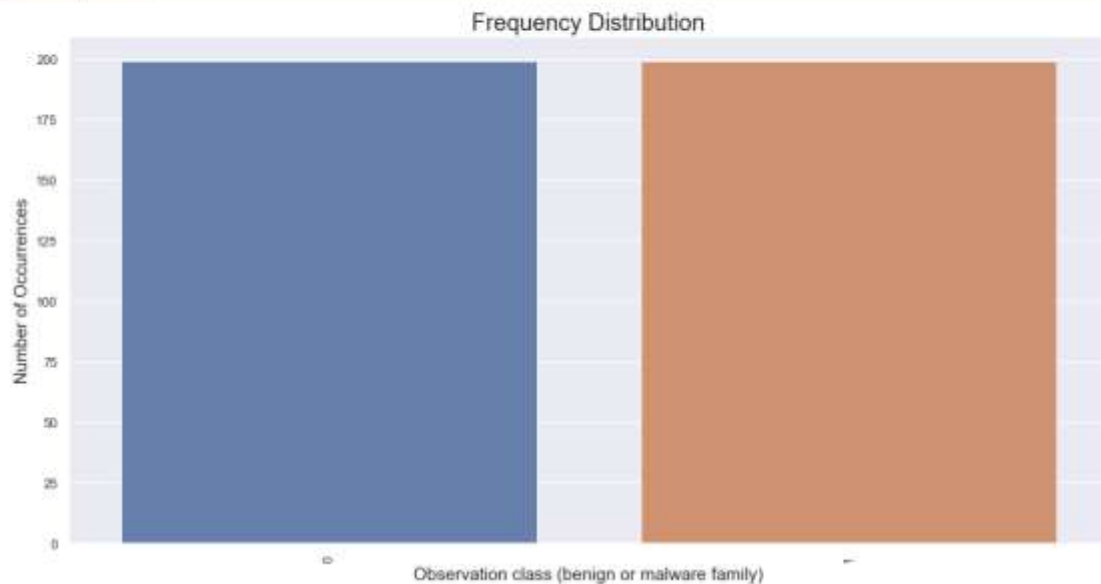


Fig.no.B.2: FREQUENCY DISTRIBUTION GRAPH

```
gnb = GaussianNB()
gnb.fit(X_train, y_train)
pred = gnb.predict(X_test)
accuracy = accuracy_score(pred, y_test)
print(accuracy)
print(classification_report(pred, y_test, labels=None))
```

```
0.9
```

	precision	recall	f1-score	support
0	0.86	0.91	0.88	33
1	0.93	0.89	0.91	47
accuracy			0.90	80
macro avg	0.90	0.90	0.90	80
weighted avg	0.90	0.90	0.90	80

```
clf = tree.DecisionTreeClassifier()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
accuracy = accuracy_score(pred, y_test)
print(clf)
print(accuracy)
print(classification_report(pred, y_test, labels=None))
```

```
DecisionTreeClassifier()
0.9
```

	precision	recall	f1-score	support
0	0.86	0.91	0.88	33
1	0.93	0.89	0.91	47
accuracy			0.90	80
macro avg	0.90	0.90	0.90	80
weighted avg	0.90	0.90	0.90	80

```
i = 15
neigh = KNeighborsClassifier(n_neighbors=i)
neigh.fit(X_train, y_train)
pred = neigh.predict(X_test)
# accuracy
accuracy = accuracy_score(pred, y_test)
print("kneighbors {}".format(i))
print(accuracy)
print(classification_report(pred, y_test, labels=None))
print("")
```

```
kneighbors 15
0.875
```

	precision	recall	f1-score	support
0	0.80	0.90	0.85	31
1	0.93	0.86	0.89	49
accuracy			0.88	80
macro avg	0.87	0.88	0.87	80
weighted avg	0.88	0.88	0.88	80

```
ada=AdaBoostClassifier()
ada.fit(X_train, y_train)
adapred= ada.predict(X_test)
accuracy = accuracy_score(adapred, y_test)
print(ada)
print(accuracy)
print(classification_report(adapred, y_test, labels=None))
```

```
AdaBoostClassifier()
0.9
```

	precision	recall	f1-score	support
0	0.83	0.94	0.88	31
1	0.96	0.88	0.91	49
accuracy			0.90	80
macro avg	0.89	0.91	0.90	80
weighted avg	0.91	0.90	0.90	80

Fig.no.B.3: ACCURACY SCORE OF ALGORITHMS

```
adaboost_pred = pd.DataFrame(adapred, columns = ['adaboost predictions'])
```

```
adaboost_pred['adaboost predictions'] = adaboost_pred['adaboost predictions'].apply(lambda x: 'yes' if x == 1 else 'no')
```

```
adaboost_pred
```

adaboost predictions	
0	yes
1	yes
2	yes
3	yes
4	yes
...	...
75	no
76	no
77	yes
78	yes
79	yes

80 rows × 1 columns

Fig.no.B.4: ADABOOST PREDICTIONS

```
s = pd.DataFrame.from_dict(cv.cv_results_)
s.columns

for p in params.keys():
    sns.catplot('param_' + p, 'mean_test_score', data=s, kind='box')
```

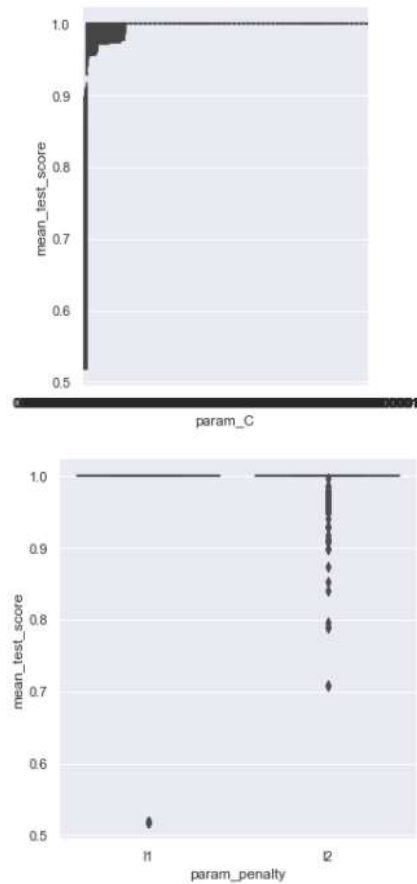


Fig.no.B.5: CATEGORICAL GRAPH

```
print('Best Model Hyper Parameters: {}'.format(cv.best_params_))
this_model = 'random forest'
model_comparison_dict[this_model] = cv.best_estimator_.score(X_test, y_test)
print('Test set accuracy = {}'.format(model_comparison_dict[this_model]))
```

```
Best Model Hyper Parameters: {'C': 0.8191, 'penalty': 'l1'}
Test set accuracy = 0.9
```

Fig.no.B.6: HYPER PARAMETERS TUNING