IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

Impact Factor: 7.301

Volume 3, Issue 16, May 2023

Detection of SQL Injection Attacks Using Ensemblimg Machine Learning Techniques

Madugula Karthik Sharma[1], Y. Rajashekhar Reddy[2], B. Pardhu[3], Preethi Jeevan[4]

[1,2,3]Student, Department of Computer Science and Engineering,

[4]Professor, Department of Computer Science and Engineering

Sreenidhi Institute of Science & Technology, Hyderabad, India

Abstract*: SQL injection remains one of the most harmful security exploits from a privacy perspective Information leakage and exonomic loss. Injection attacks are the biggest Vulnerability on the internet. The latest OWASP top 10 report shows that the number of these attacks continues to grow. Traditional Defense strategies often include static, signature-based Intrusion Detection System (IDS) rules. In most cases, they are only effective against previously observed attacks. Many current research uses machine learning techniques that can recognize the unknown. However, the attack can be performance intensive depending on the algorithm. Moreover, recently an intrusion detection strategy involves capturing traffic entering your web application. Collect data from network devices or web application hosts, or from databases in other strategies server log. This project collects traffic from two points: Host web applications and a dataphy applicance mode between your web application host and its MySQL database server.*

Keywords: Decision Tree, Logistic Regression, SQL Injections

I. INTRODUCTION

In relation to the disclosure of confidential information and monetary loss, SQL Injection remains one of the most dangerous security weaknesses.In light of the most recent The most frequent vulnerability is injection assaults, according to the OWASP Top 10 report. still increasing. IDS (Intrusion Detection System) is often used in traditional defence strategies. static, signature-based restrictions that are typically effective against known threats but useless against unidentified or zero-day threats. Approaches to machine learning are frequently employed in contemporary even though their performance can be inconsistent, studies and are capable of spotting unidentified assaults Depending on the procedure, costly.

The majority of intrusion detection systems also the web application is now collected using methods that either leverage network devices or while other techniques access the database server for information from the web application logs host. We are just using one source to collect traffic for this project: together with the associated MySQL database server, and the web application hosting entity. These datasets, along with a trained one that is associated with the first two, were all analysed by us. Using rule-based and statistical models, we were able to demonstrate that the accuracy gained with the correlated dataset was Decision-tree algorithms get results that are nearly equivalent to those from XGBOOST, ADABOOST, LGBMBOOST, but with noticeably improved performance

II. BACKGROUND STUDY(LITERATURE)

In the existing system the detection and prevention of SQLI has been addressed by a number of strategies, some of which focus on statistical analysis [4, 66, 67, 68, 69, 70] or dynamic analysis, while rest takes a hybrid strategy. The above methods have been employed in the study, scanning, detection, mitigation, and attack avoidance of web application vulnerabilities. A complete I investigation on the safety of a web application issues had been done through numerous research to investigate vulnerability analysis. By using vulnerability scanning techniques, it has also been looked into in other studies.

In the proposed system we use advanced algorithms to detect the SQL attacks ,we employ machine learning techniques,first load the dataset then familiarize the data ,visualize the data,later perform data pre-processing and EDA,,split the data and apply the machine learning algorithms like XGBOOST, ADABOOST, SVM, Random Forest, Decision Tree, AutoEncoder Neural Network,Light Gradient Boost then finally compare the models .This approach is way more accurate in preventing injection attacks and yields better results.

III. METHODOLOGY

Terms like "system analysis" and "requirements analysis" are equivalent. Additionally, it can be used to help the decision maker come up with a better course of action and choice than they otherwise would have. This process involves brainstorming and breaking the system down into its component elements in order to analyse the scenario, assess the project objectives, and break down what needs to be constructed and used to engage people so that clear requirements can be delivered.

The above diagram gives a depiction how the software functions,the user generates normal and attack traffic SQL attack and sends the data to the software, then the pre-processing of the data takes place and the data is cleaned, then the machine learning models are applied and the most accurate model is chosen.

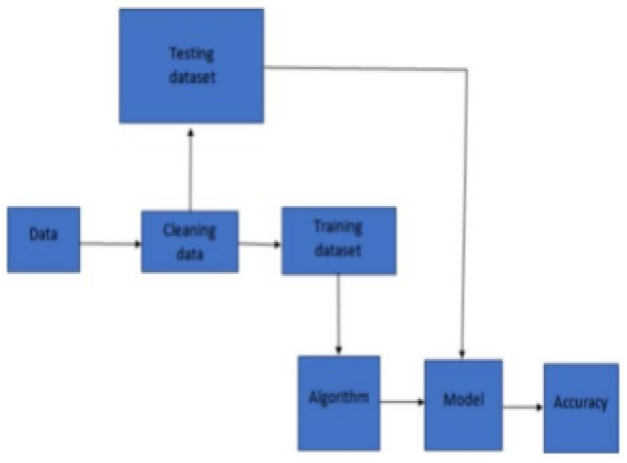


Figure 1: Software Architecture for proposed system

System design is primarily the process of developing the interfaces, architecture, components, modules, and data for a system. meet its needs. In a nutshell, it's the application of systems theory to product production. One of the most popular approaches for creating computer systems is increasingly good.

The above figure represents the architecture of the project.The architecture depicts that the data is given by a user then the data is cleaned and the testing data is made from cleaned data and given to the models for learning . Then data is loaded as a dataset and then data is cleaned.The data is then sent to the machine learning algorithms different algorithms are used and the models are compared with respect to accuracy,the model with best accuracy is chosen. The flow diagram consists of three components they are input data,model,Prediction.

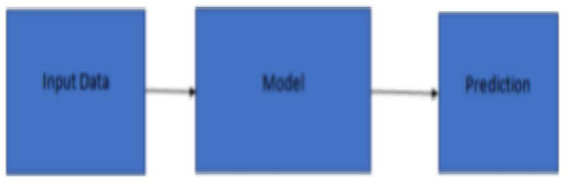


Figure 2: Data flow of mode

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 16, May 2023

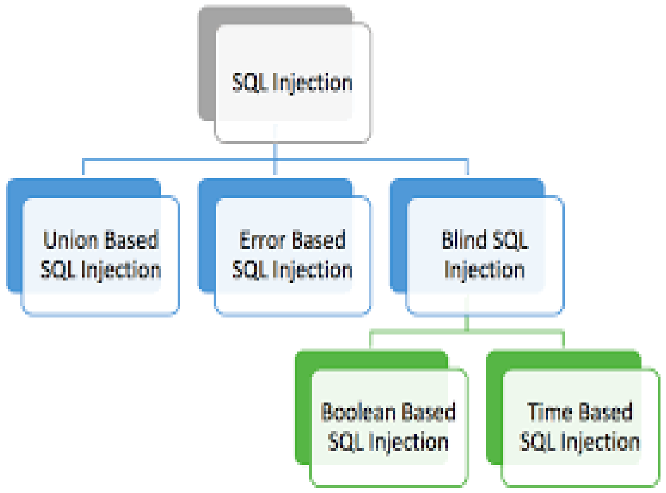

SQL Injection
Union Based SQL Injection
Error Based SQL Injection
Blind SQL Injection
Boolean Based SQL Injection
Time Based SQL Injection

Figure 3: Types of SQL Injections attacks

IV. IMPLIMENTATION:

The code is developed in the Python programming language. Python includes a vast number of libraries that can be used for scientific and computational reasons. Library examples include scikit-learn, pandas, NumPy, matplotlib, seaborn, lightgbm, xgboost, and adaboost. The following are the primary features of the project:

1. **Collection of data and Extraction of feature:**
   We got the dataset from the internet in this phase, and cleaning the dataset entails removing outliers, removing extraneous columns, and so on.

2. **Model training:**
   Following the division of the cleaned dataset into train and test splits, we will train the model with AdaBoost, LightGBM, and XGBoost. Model for testing: In this step, we will test the trained model and determine its correctness so that we may select the most accurate model.

V. ALGORITHMS IMPLEMENTEDENSEMBLE LEARNING

Ensemble learning is the process of developing and merging a large number of to solve problems using models such as classifiers or experts a given the difficulty of computational intelligence. Ensemble learning is a type of learning. commonly accustomed to help a group of individuals perform better (classification, prediction, function approximation, and so on).

LIGHTGBM:

LightGBM (Light Gradient Boosting Machine), Microsoft's distributed gradient boosting platform for machine learning, is free and open source. It is based on decision trees and is used for classification, rating, as well as other machine learning techniquesapplications. The advancement strategy places a premium on performance.
GBT, GBDT, GBRT, GBM, MART, and RF algorithms are all supported by the LightGBM framework. Sparse optimisation, training in parallel, multiple loss functions, regularisation, bagging, as well as being early pausing are just a few of the benefits of LightGBM over XGBoost.
The structure of trees distinguishes between the two. LightGBM, in contrast to the bulk of prior implementations, does not grow Row by row, build a tree. Instead, it grows trees from the ground up, one leaf at a time. It chooses the leaf in question believes will cause the least amount of loss. LightGBM, unlike XGBoost and other versions, does not use the popular sorted-based decision tree learning algorithm, which seeks the best point of division on Feature values are sorted. LightGBM, on the other hand, employs an extremely optimised decision tree learning technique according to

histograms improves performance while conserving memory. To run faster, the LightGBM method employs the novel techniques GOSS (Gradient-Based One-Side Sampling) and EFB (Exclusive Feature Bundling).
Keeping remarkable precision.The distributed version of LightGBM can train a CTR predictor on the Criteo dataset, which comprises 1.7 billion records and 67 features, in about one to two hours on a cluster of 16 PCs. LightGBM runs on Linux and C++, Python, R, and C# are supported on Windows.

Differences between other tree-based algorithms:
Light GBM builds trees vertically, whilst other algorithms grow them horizontally. Because of this distinction, Light GBM appears to grow tree leaf by leaf rather than level by level. LightGBM and other boosting methods are depicted in the diagrams below.

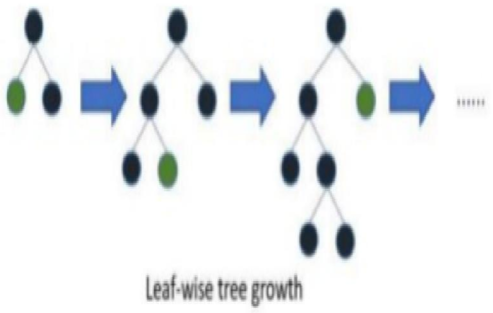


Figure 4.1: Leaf-Wise tree growth-1

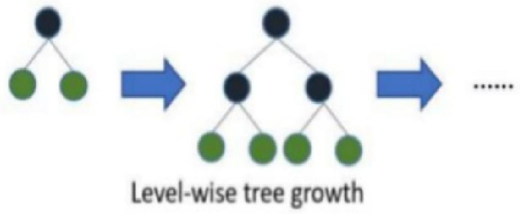


Figure 4.2: Leaf-Wise growth -2

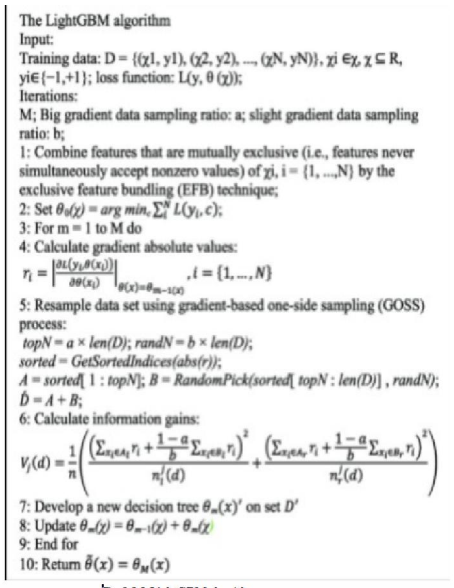
Describes how boosting algorithm works other than LightGBM

Light GBM getting so much traction:
The exponential expansion of data makes it conventional data science finds tough methodologies to provide Results are obtained more quickly. The term "light GBM" is prefixed. with the word "Light" as a result of its fast speed. Light GBM is capable of handling big volumes of information while requiring less storage space. Light GBM's appeal can also be related to its emphasis on precision. Because it supports GPU learning, data scientists are increasingly using LGBM to create data science applications.

Benefits of Light GBM:
1. Faster training: LGBM categorises and arranges continuous featureSpeedis into discrete bins using a bar graph technique, which speeds up the training process.
2. Reduced memory consumption: Continuous values are replaced with discrete bins, resulting in less memory utilisation.
3. It produces substantially more detailed trees by employing a leaf-wise split strategy as opposed to a level-wise split strategy, which is essential for achieving greater accuracy. In terms of accuracy, it outperforms all other

boosting techniques. However, it may eventually Overfitting occurs as a result of this, which can be avoided by increasing the maximum depth choice.

4. Large-Dataset's Compatibility: It can handle large datasets just as well as XGBOOST while taking significantly less time to train.



```
The LightGBM algorithm
Input:
Training data: D = {(χ1, y1), (χ2, y2), ..., (χN, yN)}, χi ∈χ, χ ⊆ R,
yi∈{−1,+1}; loss function: L(y, θ (χ));
Iterations:
M; Big gradient data sampling ratio: a; slight gradient data sampling
ratio: b;
1: Combine features that are mutually exclusive (i.e., features never
simultaneously accept nonzero values) of χi, i = {1, ...,N} by the
exclusive feature bundling (EFB) technique;
2: Set $\theta_0(\chi) = arg\ min_c \sum_i^N L(y_i, c)$;
3: For m = 1 to M do
4: Calculate gradient absolute values:
   $r_i = \left|\frac{\partial L(y_i, \theta(x_i))}{\partial \theta(x_i)}\right|_{\theta(x)=\theta_{m-1(x)}}, i = \{1, ..., N\}$
5: Resample data set using gradient-based one-side sampling (GOSS)
process:
   $topN = a \times len(D); randN = b \times len(D);$
   $sorted = GetSortedIndices(abs(r));$
   $A = sorted[1 : topN]; B = RandomPick(sorted[topN : len(D)], randN);$
   $\acute{D} = A + B;$
6: Calculate information gains:
   $V_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} r_i + \frac{1-a}{b}\sum_{x_i \in B_l} r_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} r_i + \frac{1-a}{b}\sum_{x_i \in B_r} r_i\right)^2}{n_r^j(d)}\right)$
7: Develop a new decision tree $\theta_m(x)'$ on set $D'$
8: Update $\theta_m(\chi) = \theta_{m-1}(\chi) + \theta_m(\chi)$
9: End for
10: Return $\tilde{\theta}(x) = \theta_M(x)$
```

Figure 5: Light BGM Algorithm

Applications for LightGBM:

The following major difficulties are best solved using LightGBM:

1. The log loss objective function is used to categorisebinary data.
2. Model of regression based on the L2 loss
3. There are numerous methods to categorise things.
4. The log loss objective function is used to computecross-entropy
5. LambdaRank using lambdarank and NDCG as objective functions.

When it comes to obtaining quick and accurate results, LightGBM is known as a very fast algorithm and the most often used method in machine learning. There are some in the LightGBM manual, there are nearly 100 settings to choose from.XGBoost effectiveness. Tree-based methods such as LGBM, GBM, and XGBoost use the gradient descent process to strengthen underperforming trees. XGBoost also optimises the system and enhances the algorithms to improve the underlying GBM framework.

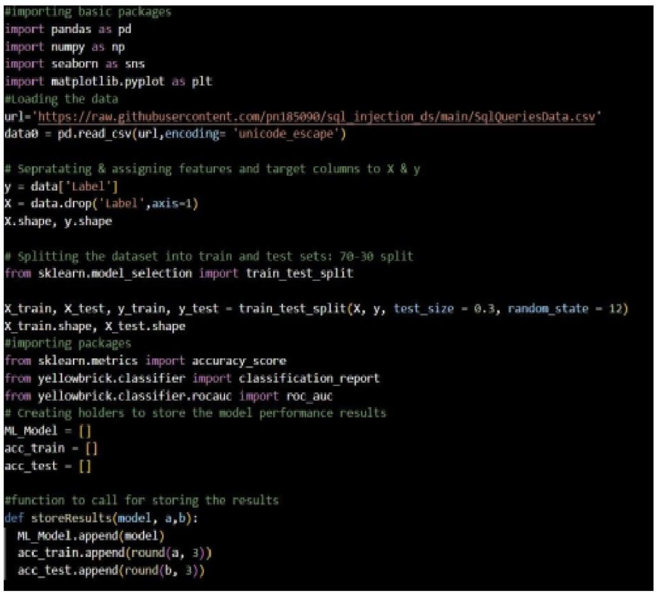
Optimization of System:

Parallelization:

To manage the sequential tree construction, XGBoost uses a parallelized technique. This is possible because the loops used to build base learners can be employed interchangeably; for example, the outermost loop counts the tree leaf nodes, while the innermost loop computes the features. Parallelization is limited because the inner loop, which is more computationally intensive, must be completed before the outer loop can be executed begin. As a result, the sequence of the loops is modified to reduce runtime using parallel thread sorting and initialization is accomplished by conducting a global scan of all instances. This option boosts algorithmic speed by accounting for compute parallelization overheads.

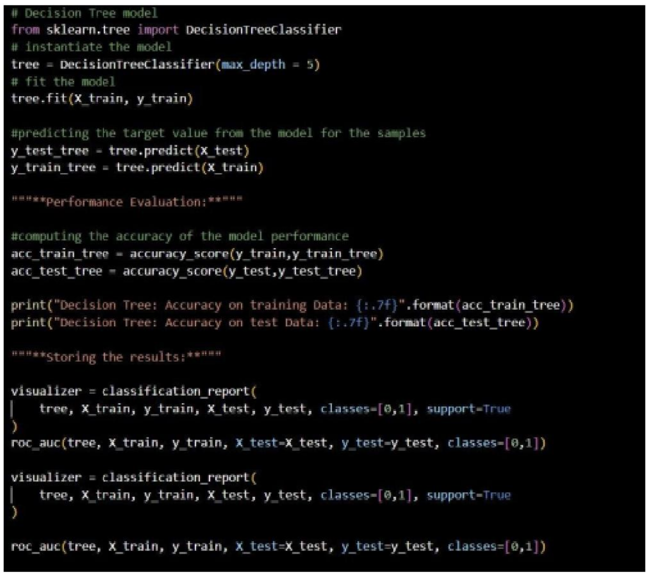
Optimisation of Hardware:

This strategy was developed to create best use in terms of hardware resources available. This is performed by cache awareness, which necessitates each thread creating internal buffers to store gradient statistics. Out-of-core computing, for example, makes the most of available resources. storage capacity while working with big data sets that cannot fit in memory.In this research, we used three different ensemble machine techniques: XGBoost, AdaBoost, Random Forest, and LGBM.
Initially, we import all of the required packages before loading the data from the dataset.



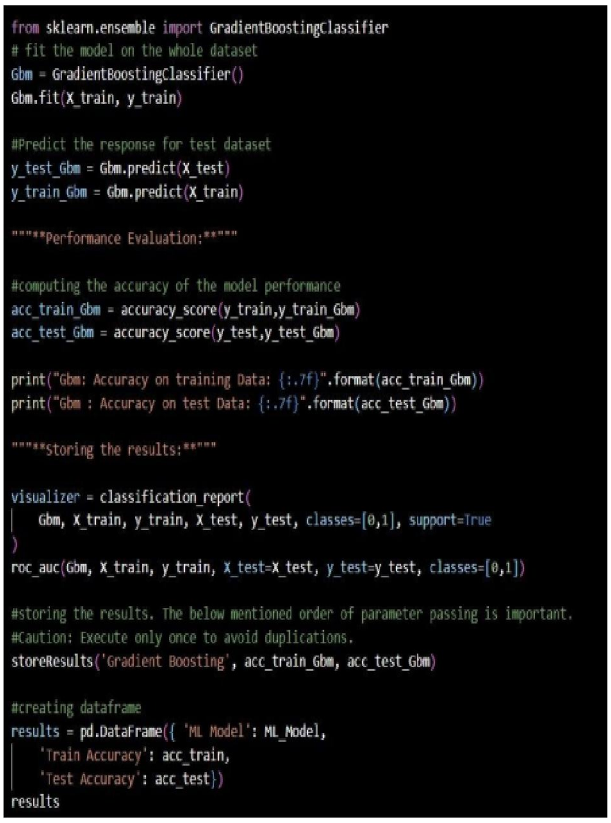
```
#importing basic packages
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
#Loading the data
url='https://raw.githubusercontent.com/pn185090/sql_injection_ds/main/SqlQueriesData.csv'
data0 = pd.read_csv(url,encoding= 'unicode_escape')

# Sepratating & assigning features and target columns to X & y
y = data['Label']
X = data.drop('Label',axis=1)
X.shape, y.shape

# Splitting the dataset into train and test sets: 70-30 split
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 12)
X_train.shape, X_test.shape
#importing packages
from sklearn.metrics import accuracy_score
from yellowbrick.classifier import classification_report
from yellowbrick.classifier.rocauc import roc_auc
# Creating holders to store the model performance results
ML_Model = []
acc_train = []
acc_test = []

#function to call for storing the results
def storeResults(model, a,b):
  ML_Model.append(model)
  acc_train.append(round(a, 3))
  acc_test.append(round(b, 3))
```

Figure 6: Importing Packages



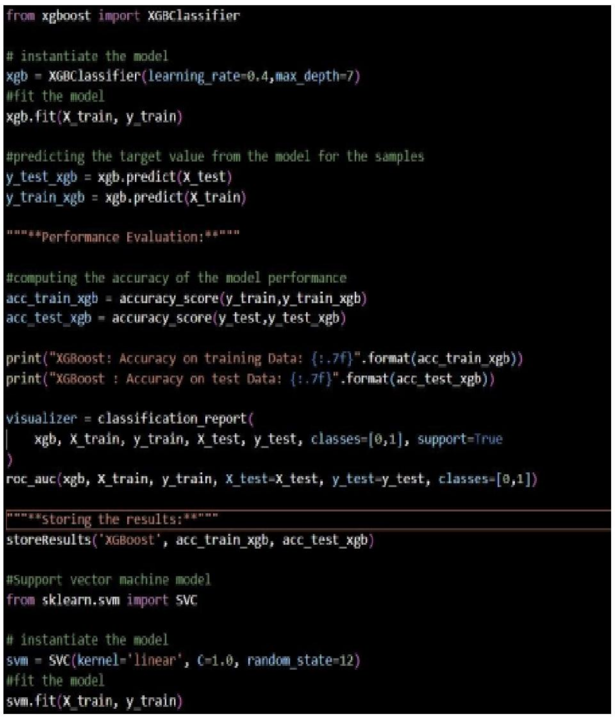
```
# Decision Tree model
from sklearn.tree import DecisionTreeClassifier
# instantiate the model
tree = DecisionTreeClassifier(max_depth = 5)
# fit the model
tree.fit(X_train, y_train)

#predicting the target value from the model for the samples
y_test_tree = tree.predict(X_test)
y_train_tree = tree.predict(X_train)

"""**Performance Evaluation:**"""

#computing the accuracy of the model performance
acc_train_tree = accuracy_score(y_train,y_train_tree)
acc_test_tree = accuracy_score(y_test,y_test_tree)

print("Decision Tree: Accuracy on training Data: {:.7f}".format(acc_train_tree))
print("Decision Tree: Accuracy on test Data: {:.7f}".format(acc_test_tree))

"""**Storing the results:**"""

visualizer = classification_report(
    tree, X_train, y_train, X_test, y_test, classes=[0,1], support=True
)
roc_auc(tree, X_train, y_train, X_test=X_test, y_test=y_test, classes=[0,1])

visualizer = classification_report(
    tree, X_train, y_train, X_test, y_test, classes=[0,1], support=True
)

roc_auc(tree, X_train, y_train, X_test=X_test, y_test=y_test, classes=[0,1])
```

Figure7: Decision Trees



```
from sklearn.ensemble import GradientBoostingClassifier
# fit the model on the whole dataset
Gbm = GradientBoostingClassifier()
Gbm.fit(X_train, y_train)

#Predict the response for test dataset
y_test_Gbm = Gbm.predict(X_test)
y_train_Gbm = Gbm.predict(X_train)

"""**Performance Evaluation:**"""

#computing the accuracy of the model performance
acc_train_Gbm = accuracy_score(y_train,y_train_Gbm)
acc_test_Gbm = accuracy_score(y_test,y_test_Gbm)

print("Gbm: Accuracy on training Data: {:.7f}".format(acc_train_Gbm))
print("Gbm : Accuracy on test Data: {:.7f}".format(acc_test_Gbm))

"""**Storing the results:**"""

visualizer = classification_report(
    Gbm, X_train, y_train, X_test, y_test, classes=[0,1], support=True
)
roc_auc(Gbm, X_train, y_train, X_test=X_test, y_test=y_test, classes=[0,1])

#storing the results. The below mentioned order of parameter passing is important.
#Caution: Execute only once to avoid duplications.
storeResults('Gradient Boosting', acc_train_Gbm, acc_test_Gbm)

#creating dataframe
results = pd.DataFrame({ 'ML Model': ML_Model,
    'Train Accuracy': acc_train,
    'Test Accuracy': acc_test})
results
```

Figure 8:XGB Classifier

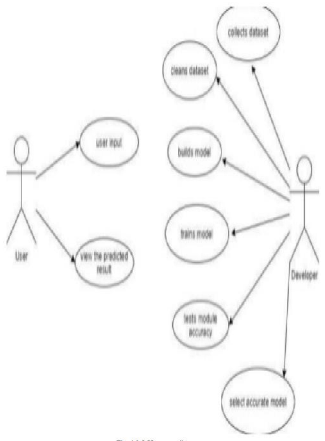


```
from xgboost import XGBClassifier

# instantiate the model
xgb = XGBClassifier(learning_rate=0.4,max_depth=7)
#fit the model
xgb.fit(X_train, y_train)

#predicting the target value from the model for the samples
y_test_xgb = xgb.predict(X_test)
y_train_xgb = xgb.predict(X_train)

"""**Performance Evaluation:**"""

#computing the accuracy of the model performance
acc_train_xgb = accuracy_score(y_train,y_train_xgb)
acc_test_xgb = accuracy_score(y_test,y_test_xgb)

print("XGBoost: Accuracy on training Data: {:.7f}".format(acc_train_xgb))
print("XGBoost : Accuracy on test Data: {:.7f}".format(acc_test_xgb))

visualizer = classification_report(
    xgb, X_train, y_train, X_test, y_test, classes=[0,1], support=True
)
roc_auc(xgb, X_train, y_train, X_test=X_test, y_test=y_test, classes=[0,1])

"""**Storing the results:**"""
storeResults('XGBoost', acc_train_xgb, acc_test_xgb)

#Support vector machine model
from sklearn.svm import SVC

# instantiate the model
svm = SVC(kernel='linear', C=1.0, random_state=12)
#fit the model
svm.fit(X_train, y_train)
```

Figure 9: Gradient boosting

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 16, May 2023

UML Diagrams:



USE CASE DIAGRAM

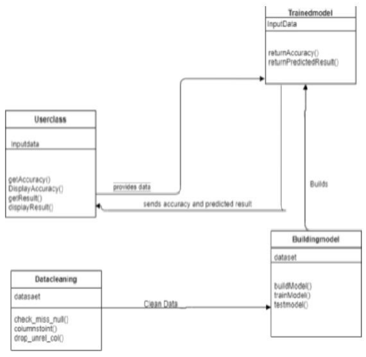
Using a use case diagram, you can show the traits of your system's users and how they interact with it. Actors are referred to as system users in use cases. We can create one by combining a number of symbols. An effective use case helps the team understand how people or organisations utilise the programme or service. helps actors achieve the scope of your system using your system or application. A use case diagram does not go into great detail; it does not, for instance, show the order in which steps are carried out. A good use case diagram gives us general understanding about how use cases in addition to actors interact. In contrast, a proper use case diagram provides an interaction. In contrast, a proper use case diagram displays the interaction between use cases and actors in great detail.

In this use case we have two actors they are user, developer the user sends the data and receivesoutput, the developer collects the dataset cleans dataset, builds model trains model,tests model accuracy selects accurate model.

Class Diagram:

Class diagram can also be called as a static diagram. It presents a static image of the programme. Many aspects of the system can be described, illustrated, and documented using a class diagram. It can also be used to generate source codes for a software programme. A class diagram outlines a class's characteristics, behaviours, and limitations. A set of classes, interfaces, collaborations, restrictions, and interactions are represented visually in a class diagram. It is also known as a structural diagram.

We have four classes, they are Trained model,Userclass building model data cleaning the trained model has two functions,userclass has 4 functions,the building model and data cleaning has three functions.The Trained model has input data and sent by the user class which also has input data the building model and data cleaning has dataset dash under it. Sequence diagrams use a variety of message symbols.It presents a static image of the programme. Many aspects of the system can be described, illustrated, and documented using a class diagram.



IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal

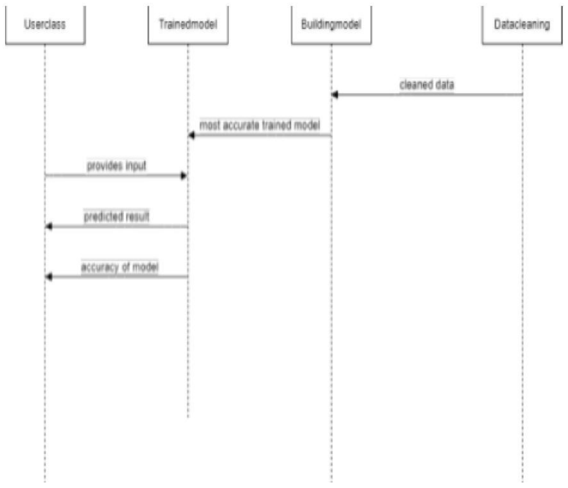
Impact Factor: 7.301

Volume 3, Issue 16, May 2023

Sequence Diagram:

The sequence diagram displays interactions among items or pieces in a time-based manner. This graphic is used by software developers and business experts to comprehend system requirements. The method can also be described in writing using this diagram. They are also referred to as event diagrams. This diagram provides a thorough view of use cases. The amount of time it takes an object to finish its job is represented by activation. An object's consecutive tasks are indicated with dash under it. Sequence diagrams use a variety of message symbols.It presents a static image of the programme. Many aspects of the system can be described, illustrated, and documented using a class diagram.

Here we have four UserClass, Trainedmodel, Buildingmodel, Datacleaning. The userclass provides input,the datacleaning sends cleaned data,building model then pre-processes data and sends most accurate trained model finallyit predicts result and accuracy of model and sends it to the userclass.



Feature Extraction:

With the present project, tokenization has been utilized to extract attributes. To prepare data for machine learning model training and testing, we divided the SQL queries into tokens and used those tokens. 20 features from the tokens have been taken in this section. In essence, each of these attributes is predicated on a token's count. A few examples of characteristics include the following: the amount of single quotes, double quotes, one-line comments, multiple-line comments, spaces, regular keywords, harmful keywords, NA values, percentage symbols, alphabets, and numbers.Twenty of these features are obtained from the queries after these kinds of features are retrieved from each query.

The majority of the time, database table activities are carried out using SQLi keywords. Due to the unexpected actions, they take, these keywords are essential for starting a SQL injection attack. As a result, it's crucial to distinguish between legitimate and fraudulent requests while using these terms. To implement such a process, the tokenization approach is utilised to extract tokens from actual requests. A query is tokenized by being divided into tokens (keywords). Model will extract the properties from these tokens that were generated as a result of the queries. Every integer in a numerical sequence representing each question that corresponds to a Tab property. Accurate identification of these traits is required in order to detect SQLi attacks. The recognition of most SQIA types, including those that are treated similarly to SQLqueries, such as stored procedures, redundancies/tautologies, unions, piggybacked, illegal, and logically flawed, served as the foundation for the selection of these features.

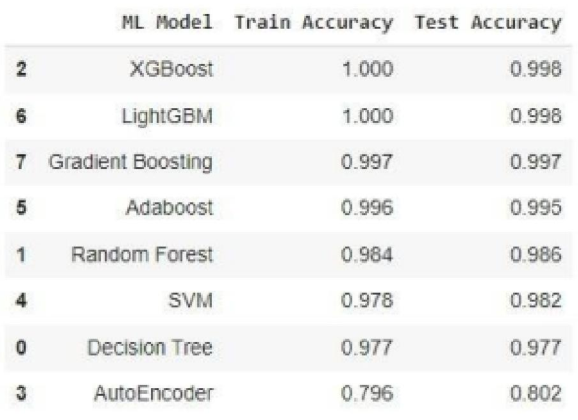
Training the Model:

Using the two approaches outlined in the aforementioned module, we divided this dataset to train and evaluate data. Using this train dataset, we will now build a model and train it. LightGBM, XGBoost, AdaBoost, GBM, and Random Forest are some of the different Ensembling machine learning techniques implementations that we will employ in this project to train our model.

Testing the Model:

After the model has been trained, it will be tested using test data to determine model accuracy. Because we built models using various techniques and strategies for dividing test and train data, we may choose based on accuracy, the most precise representation of all evaluated models so that we can more accurately forecast attacks.

VI. RESULT



	ML Model	Train Accuracy	Test Accuracy
2	XGBoost	1.000	0.998
6	LightGBM	1.000	0.998
7	Gradient Boosting	0.997	0.997
5	Adaboost	0.996	0.995
1	Random Forest	0.984	0.986
4	SVM	0.978	0.982
0	Decision Tree	0.977	0.977
3	AutoEncoder	0.796	0.802

Figure10: Result

VII. CONCLUSION

In comparison to other machine learning techniques, we achieved better accuracy and F1 scores using the ensembling machine learning algorithms, XGBoost, Light GBM, and AdaBoost, with a split of 70% train and 30% test datasets.A hacker can exploit the flaws and submit harmful SQL queries to the database. It is critical to detect and prevent SQLi attacks in order to secure people's sensitive information. Because these losses can cause a slew of issues for both individuals and organisations. As a result, harnessing AI and machine learning can assist us in detecting and preventing these types of cyber threats.

VIII. FUTURE SCOPE

In order to achieve high accuracy in SQL injection detection methods, we'll be measuring a number of web-based application codes available in the public domain as part of a forthcoming study. Using SQL integration with the Nikto HTTP scanner, HTTP scanning proxies, and Metasploit, many web-based threats will be found. In this instance, CNN is being used to build a detection model and for future studies, however we can enhance the model by utilising other algorithms and identifying the best model. The static analysis approach will be used in this instance, and run-time analysis will be added as a subsequent project. We've only built a detection system for SQL Injection so far, but we can later develop a preventive method.

REFERENCES

[1] Sonali Mishra, “SQL Injection Detection using Machine Learning”, from https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?arti c le=1727context=etdprojects, on23May2019 pp.10 - 29.

[2] Bojken Shehu and Aleksander Xhuvani, “A LiteratureReviewandComparativeAnalyseson SQL Injection: Vulnerabilities, Attacks and their Prevention and Detection Techniques”from https ://pdfs.semanticscholar.org,Vol.11,Issue4,No 1,July 2014 pp 20 - 34.

[3] SuhaimiIbrahim, “SQLInjectionDetectionand Prevention Techniques” from https://pdfs.semanticscholar.org/ Volume 3, Number 7, August 2011, pp 85 - 89.

[4] G.Wassermann,Z.Su,“Analysisframeworkfor security in web applications,” In: Proceedings of the FSE Workshop on Specificationand Verification of Component-Based Systems, fromhttps://link.springer.com/chapter/10.1007/978 −0 −387 − 44599−15 SAVCBS,pp. 70–78, 2004.

[5] Mei Junjin, “An Approach for SQL Injection Vulnerability Detection,” Proceedings. of the 6th Int. Conf. on InformationTechnology: NewGenerations,LasVegas,Nevada,pp.14-19, Apr. 2009.

[6] V.Haldar, D.Chandra, and M.Franz, "Dynamic Taint Propagation for Java," Proc. 2 1s t Annual Computer Security Applications Conference, Dec 2005.

[7] S.W.Boyd and AD.Keromytis, "SQLrand: Preventing SQL Injection Attacks," Proc. the 2nd Applied Cryptography and Network Security (ACNS) Conference, pp. 292-302, Jun 2004.

[8] G.T.Buehrer, RW.Weide, and P.AG.Sivilotti, "Using Parse Tree Validation to Prevent SQL Injection Attacks," International Workshop on Software Engineering and Middleware (SEM), 2005.

[9] Evans Dogbe, Richard Millham, Prenitha Singh “A Combined Approach to Prevent SQL Injection Attacks,” Science and Information Conference 2013 October 7-9, 2013, London, UK.

[10] Ryohei Komiya, Incheon Paik, Masayuki Hisada, “Classification of Malicious Web Code by Machine Learning,” Awareness Science and Technology (iCAST), 2011 3rd International Conference on, vol., no., pp.406,411, 27-30 Sept. 2011