Malnad College of Engineering

(An Autonomous Institution under Visvesvaraya Technological University, Belagavi, Accredited by NBA under Tier-I and NAAC)

Hassan – 573 202

"A NOVEL MACHINE LEARNING BASED HYBRID MODEL FOR WEBSHELL DETECTION"

A dissertation submitted to Malnad College of Engineering, Hassan, during the academic year 2020-2021 in partial fulfilment for the award of the degree of

Bachelor of Engineering

in

Computer Science & Engineering

By

Megha D	(4MC17CS031)
Pooja K P	(4MC17CS033)
Roja Mallik S P	(4MC17CS042)
Syeda Shafain Fathima	(4MC17CS057)

Under the guidance of

Dr. Chandrika J
Professor
(Department of Computer Science and Engineering)

Department of Computer Science & Engineering Malnad College of Engineering Hassan - 573 202

Tel.: 08172-245093 Tel.: 08172-245020 (Dept.) URL: www.mcehassan.ac.in

2020-2021

Malnad College of Engineering

(An autonomous Institution under Visvesvaraya Technological University, Belagavi, Accredited by NBA under Tier-I and NAAC)

Hassan – 573 202

Department of Computer Science & Engineering

CERTIFICATE

Certified that the Project titled

"A NOVEL MACHINE LEARNING BASED HYBRID MODEL FOR WEBSHELL DETECTION"

is a bonafide work carried out by

Megha D	(4MC17CS031)
Pooja K P	(4MC17CS033)
Roja Mallik S P	(4MC17CS042)
Syeda Shafain Fathima	(4MC17CS057)

in partial fulfillment for the award of

Bachelor Degree in Computer Science & Engineering

of

Malnad College of Engineering

affiliated to

Visvesvaraya Technological University, Belagavi

during the year 2020-2021. It is certified that all corrections/ suggestions indicated for Internal Assessment have been incorporated in the report deposited in the Department Library. The Project Report has been approved as it satisfies the academic requirements in respect of Project Work prescribed for the Bachelor of Engineering Degree.

(Dr. Geetha Kiran A)	(Dr. Chandrika J)	(Dr. C V Venkatesh)
B.E., M.Tech, Ph.D,	B.E.,M.Tech, Ph.D,	B.E., M.Tech, Ph.D,
Professor and Head of Department	**Prof. Dept of CSE**	**Principal**

External Viva

Name of the Examiners	**Signature with Date**
1.	
2.	

ACKNOWLEDGEMENT

We present with immense pleasure this work titled "**A NOVEL MACHINE LEARNING BASED HYBRID MODEL FOR WEBSHELL DETECTION**".

An endeavor over a long period can be successful with the advice and support of many well-wishers. We take this opportunity to express our sincere gratitude and appreciation to all of them.

The satisfaction and delight that accompany the successful completion of any task would be incomplete without mentioning the people who made it possible. So, with gratitude we acknowledge all those whose guidance and encouragement made to successfully complete this project.

We would like to express sincere thanks to our Principal **Dr. C.V. Venkatesh**, Malnad College of Engineering for his encouragement that motivated us for successful completion of this project work.

We wish to express our gratitude to **Dr. Geetha Kiran A**, Professor and Head, Department of CS&E for providing a good working environment and for her constant support and encouragement.

It gives us an immense pleasure in placing on record a deep sense of gratitude to our guide **Dr. Chandrika J**, B.E., M. Tech., Ph.D., Dept. of CSE for expert guidance, initiative and encouragement that led us throughout this project work.

We would also like to thank all the staffs of Computer Science and Engineering department who have directly or indirectly helped us in the completion of the project work.

At last we would hereby acknowledge and thank our parents who have been a source of inspiration and also instrumental in the successful completion of this project work.

Project Associates,

Megha D	(4MC17CS031)
Pooja K P	(4MC17CS033)
Roja Mallik S P	(4MC17CS042)
Syeda Shafain Fathima	(4MC17CS057)

CONTENTS

ABSTRACT

With fast turn of events and development of the web, webshell is one of major digital dangers nowadays. Henceforth, web shell detection is an important factor in the security of computer systems. Nowadays, attackers generally design polymeric webshell, it is usually a type of web shell that continuously changes its recognizable feature to fool detection techniques that uses typical signature based methods. That is why the need for Machine Learning based detection arises. In this work, we are going to obtain behavioral-pattern that may be achieved through static or dynamic analysis, afterward we can apply dissimilar ML techniques to identify whether it's web shell or not. Behavioral based Detection methods will be discussed to take advantage from ML algorithms so as to frame social-based web shell recognition and classification model.

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Webshells have posed threat and challenge to the security of websites and applications hosted in the internet. The weekly safety report of National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC) in 2019, has crucially stated that the backdoors for the websites are growing in a rapid phase[1]. Vulnerabilities in web-based applications are strategically exploited through backdoors. Malicious file inclusion through SQL injection is achieved where loop hole in server security configuration is made use of. Webshells with ransomware capabilities can induce brute-force attacks, botnet command and control for denial of service(DDoS) attacks and many other dangerous activities. Obfuscation is often used to bypass endpoint security software and antivirus software deployed.

Web shells are installed through vulnerabilities in web application or weak server security configuration, include SQL injection, vulnerabilities in applications and services (e.g. web server software such as NGINX or content management system applications such as WordPress), remote file inclusion (RFI) and local file inclusion (LFI) vulnerabilities; file processing and uploading vulnerabilities, vulnerable plugg-ins in web servers. Webshells for maintaining persistent access after initial attack vector is into the system(PAS webshell) is also established. Different detection methods are widely employed to mitigate webshell attacks. This project model detects based on behaviors related to network using TCP dump data.

1.1 What is Web Shell?

A web shell is a malicious script used by an attacker with the intent to escalate and maintain persistent access on an already compromised web application. A web shell itself cannot attack or exploit a remote vulnerability, so it is always the second step of an attack (this stage is also referred to as post-exploitation). An attacker can take advantage of common web page vulnerabilities such as SQL injection, remote file inclusion (RFI), or even use cross-site scripting (XSS) as part of a social engineering attack in order to attain file upload capabilities and transfer the malicious files. The common functionality includes but is not limited to shell command execution (access to cmd/command line), code execution, database

enumeration, and file management. Web shells could be written in many web languages, for example, PHP web shells are very common. They can affect you no matter whether your system is based on custom software or on a common content management system such as WordPress with plugins[2]. Web shells might also not get detected by antivirus or anti-malware software because they do not use typical executable file types. With the development of Internet technology, web-based applications have been assimilated into all aspects of our lives. With the rapid growth of the number of website visitors, websites also store large amounts of our personal information; thus, the issue of how to protect this private information has become the primary task of website maintenance staff. According to the "Overview of China's Internet Network Security Situation in 2019" released by the National Internet Emergency Center in 2019, CNCERT monitors found that approximately 45,000 IP addresses inside and outside China implanted backdoors into approximately 85,000 websites in China and that the number of websites with backdoors implanted in China has increased by more than 2.59 times compared to the number found in 2018. As the number of backdoors implanted in websites increases year by year, the issue of how to detect backdoors in websites is critical for data security. Malicious WebShell files can function as website backdoors, so the detection of WebShell files on websites is also very important.

WebShell is an executable program language written with web scripts such as ASP, PHP, and JSP. It is always referred to as a web backdoor because users can upload malicious files to a web page and obtain database information by executing OS commands to view the database. As PHP is the preferred language for website development, it is also very important to study the detection method of PHP-type WebShell. Webshell attack can be divided into two categories, "large Trojan" file for attack and "micro Trojan" file for attack. "micro Trojan" file code is small, usually a few lines to dozens of lines, its main function is used to assist" large Trojan"file upload, execution script command. Compared with "micro Trojan" file size is much larger, "large Trojan" file size even more than 1 MB, its functions are complex, including the execution of command line procedures, database operations, etc. In addition, "large Trojan" to complete its function can also cooperate with other offensive files to operate jointly, to achieve the purpose of attack.Webshells have become the main threat challenges for protecting the security of the websites.

Web shell detection is any product tenaciously intended to make harm a PC, server, customer or PC organization. With the fast advancement and development of the web, web

shell has turned out to be one of the major digital dangers in nowadays. In the year 2017, a cybersecurity and anti-virus provider like Kaspersky Labs defined web shell as a kind of PC program planned to taint an authentic client's PC and perpetrate hurt on it in different manners." As the decent variety of web shell is expanding nowadays, antivirus tools are not capable of fulfilling the need of protection and results in millions of hosts being hacked. In addition to that, the skills required for web shell development is decreasing due to high availability of attacking tools on the Internet. According to data of AV-TEST Institute, they reported over three lac fifty thousand novel web shell (malicious projects) and PUA (Potentially Unwanted Applications) consistently. Therefore, to protect computer system from web shell is one significant tasks of cybersecurity for a single user as well as for businesses because even a single attack can result in huge information and financial loss.

1.2 Types of Web Shell

Web shell can be isolated into a few classes relying upon its purpose. The classes are as per the following:

- **Adware:** It is the slightest risky and the most rewarding web shell, it shows advertisements on PC.
- **Spyware:** As it implies from the name, the web shell that uses for spying. Some run of the mill activities of spyware incorporate following inquiry history to send customized advertisements, following exercises to offer them to the outsiders in this way.
- **Virus**: This is the most straightforward type of the software. It is basically any bit of programming that is stacked and propelled without client's authorization while replicating itself or contaminating changing other programming. Regularly this is spread by sharing records or programming between PCs. • Worm: It is a program that imitates itself and obliterates data and records on the PC.
- **Trojan**: It is a kind of malicious code and computer software to look authorized but can control the system or machine.
- **RootKit**: An assortment of vindictive programming created to enable access to a framework or on particular area of the system.
- **Backdoors**: It is a method to convert the bypassing normal authentication and encryption.

- **Keyloggers**: It is totally depend on Keyboard working style like the action of keyboard typically covertly unaware their actions are being method and monitored.
- **Ransomware**: This is web shell software but extreme use of this software is for Accounts section like access to system until a sum of money is paid.
- **Browser Hijacker**: It is a type of undesirable programming that changes a program's setting without client authorization, to infuse the profitless promoting into program.

1.3 Web Shell Discovery Investigation Procedures

Different detection methods are widely employed[3][4], which is classified as follows:

- **Static and Dynamic Detection**: This kind of detection method extracts information from keywords, file permissions, file modification times, file owners, high-risk functions, and other dimensions around the characteristics of the script file.
- **Flow Analysis Detection:** The core of this method is to visualize the traffic by establishing a gateway, and then to monitor the payload network traffic generated during the webshell access process. After a certain amount of payload accumulation and related rules are formulated, a traffic-analysis-based detection engine is built up in combination with other detection processes, and then embedded in an existing gateway device or cloud device to achieve the webshell's in-depth analysis and detection.
- **Log Analysis Detection:** As an analytical tool for forensics and prediction, log analysis reveals complete attacks that have occurred, are occurring, or will occur in the future Webshell detection using log analysis performs event backtracking based on certain attack events and prevents the next attack according to the characteristics of the first.
- **Behavior Analysis Detection**: Behavioral analysis technology involves the parsing of source code files in the system environment. Behaviors related to files include their reading and writing, as well their creation and deletion. Behaviors related to networks include socket monitoring behavior and TCP/UDP/HTTP request sending (DDOS attack).
- **Statistical Analysis:** The focus of this technique is on identifying obfuscated webshell based on how webshell scripts differ from normal files. It mainly involves five features: Information entropy, which uses ASCII code to measure the uncertainty of a file; The longest word, the length of the string in the normal file is in line with the English

specification. The long string appears to mean that the code is coded and obfuscated; Index of coincidence, a low coincidence Index indicates that the file is obfuscated; Sig-Nature, matching feature function and sensitive code; Compression ratio.

1.4 Problem Statement

The design of efficient framework for webshell attack detection is the target. The model proposed is aimed on detecting malicious connections based on behaviors related to network using TCP dump data. Hybrid model is proposed after feature extraction, data pre-processing, experimentation and comparison of different machine learning methods.

The main concepts of this project is as follows:

i. The presented model utilizes behavioral based detection methods to take advantage from ML algorithms so as to frame social-based webshell attack recognition and classification model.
ii. The project compares KNN, Logistic Regression, Decision tree and naïve bayes bernouli, naïve bayes guassian classification models. High performing models are chosen for hybrid ensemble.
iii. Hybrid model of the selected models is performed which yields the highest accuracy of all other individual models.

CHAPTER 2

LITERATURE SURVEY

[5] **Félix Iglesias et al., (2014)** have proposed a model that addresses the feature selection problem for network traffic based anomaly detection. A multi-stage feature selection method using filters and stepwise regression wrappers are proposed. This paper has shown the elimination of expensive 13 features out of 41 features, significantly reducing the computational cost and effort enhancing the experimentation and feature generation from live traffic observations at network nodes.

[6] **Firdausi et al., (2010)** have proposed a system to study the behavior of every malware on an emulated (sandbox) environment. This environment will automatically analyze and generate behavior reports. Preprocessing of the reports into sparse vector models is employed for subsequent machine learning(classification). KNN, Naïve Bayes, J48 Decision Tree, Support Vector Machine (SVM), and Multilayer Perceptron Neural Network (MlP) are the classifiers experimented in this research. The overall best performance was achieved by J48 decision tree. It can be concluded by the proof -of-concept achieved through behavior-based malware analysis using machine learning techniques is effective and efficient for malware detection.

[7] **Zhuang Ai et al., (2020)** have proposed a deep super learner for attack detection. First, the collected data are deduplicated to prevent the influence of duplicate data on the result. Second, to detect the results of the algorithm, static and dynamic feature are taken as the feature of the algorithm to construct a comprehensive feature set. Word2Vec algorithm is used to vectorize the features. During this period, to prevent the outbreak of the number of features, genetic algorithm is used to extract the validity of the feature dimension. Finally, deep super learner is used to detect Web Shell. The experimental results show that this algorithm can effectively detect Web Shell, and its accuracy and recall are greatly improved.

[8] **Yixin Wu et al., (2019)** have proposed detection of web shell communication through an intelligent and innovative framework that employs precise sessions derived from the weblogs. Features were extracted from the raw sequence data in weblogs. Session identification is achieved specifically by the time interval-based statistical method. The paper incorporates long

short-term memory and hidden Markov model to constitute the framework, respectively. The framework was then evaluated with real-time data. The experiment shows that the LSTM based model can achieve a higher accuracy rate. The experiment results in higher efficiency of the proposed approach in terms of the quick detection without source code, especially when it only considers detecting for a certain period, as it takes 98.5% less time than the cited related approach to get the result.

[9] **Muataz Salam Al-Daweri et al., (2020)** have proposed an extensive analysis of the relevance of individual features in the KDD99 and UNSW-NB15 datasets. The major 3 methods employed consist of rough-set theory (RST), a back-propagation neural network (BPNN), and a discrete variant of the cuttlefish algorithm (D-CFA). Initially, the dependency ratio was calculated between the features and the classes, using the RST. In the next step, each feature in the datasets is fed to the BPNN as input, inorder to measure the ability for a classification task concerning each class. Later, a feature-selection process was undertaken iteratively, to indicate the frequency of the selection of each feature. The results indicate that certain features in the KDD99 dataset can be used to achieve a classification accuracy above 84%. High contributing features were obtained which are a combination of features that yield higher accuracy. This study is expected to fuel cybersecurity enthusiasts for the creation of a lightweight and accurate IDS model with apt features.

CHAPTER 3

METHODOLOGY

The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between "bad" connections called attacks/anomalies and "good" normal connections. It is composed of several components as explained in further sections.

3.1 Proposed Approach

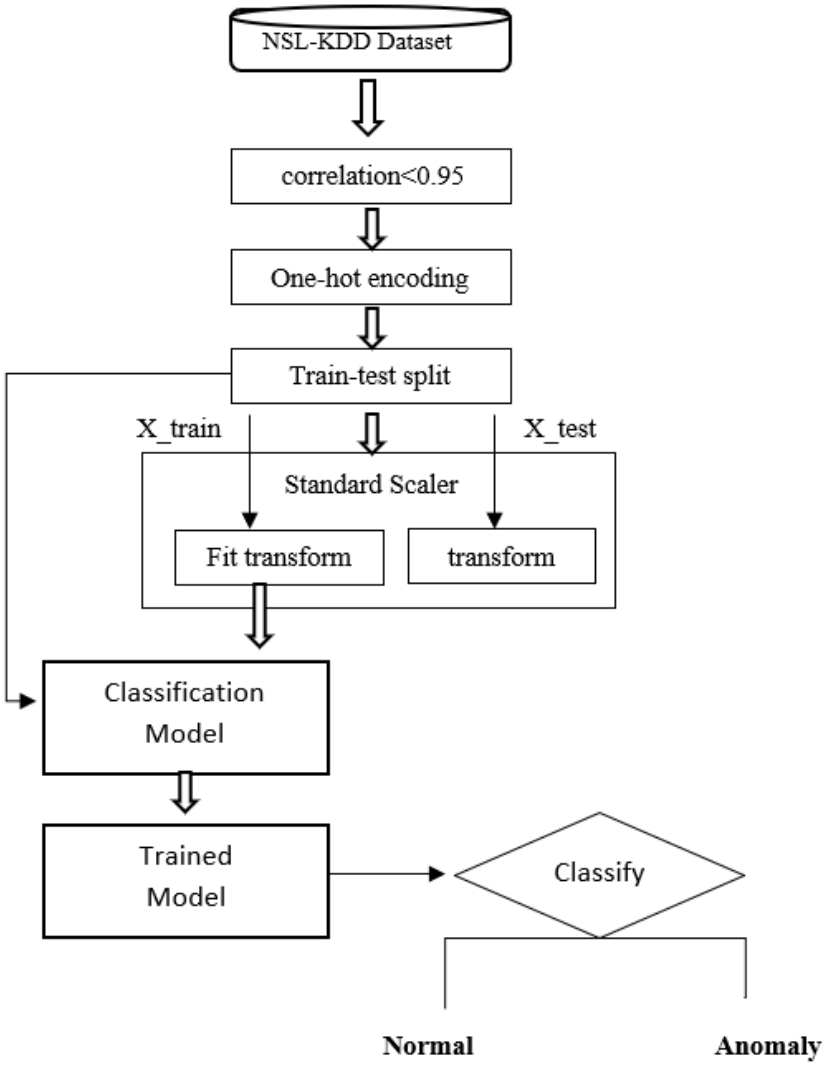

Figure 3.1: Architecture of the proposed model

Dataset:

The dataset used is NSL-KDD. NSL-KDD is a data set suggested to solve some of the inherent problems of the KDD'99 data set. The 1998 DARPA Intrusion Detection Evaluation Program was prepared and managed by MIT Lincoln Labs. The objective was to survey and evaluate research in intrusion detection. Lincoln Labs set up an environment to acquire nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment, but peppered it with multiple attacks. NSL KDD does not include redundant records in the train and test set. Therefore, the performance of the learners are not biased by the methods which have better detection rates on the frequent records.

Sample Tuple of NSL-KDD dataset:

"0,tcp,http,SF,335,241,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,4,18,0,0,0,0,0,1,0,0.11,91,246,1,0,0.01,0.03,0,0,0,0"

41 features of NSL-KDD dataset are classified into 4 types:

1. **Basic features**: Features of individual TCP connections
   Features included: duration, protocol_type, service, flag, src_bytes, dst_bytes, land, wrong_fragment, urgent

2. **Content features**: Features within a connection suggested by domain knowledge
   Feature included: hot, num_failed_logins, logged_in, num_compromised, root_shell ,su_attempted, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, is_host_login, is_guest_login

3. **Traffic features**: Features computed using a two-second time window.
   Features included: count,srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate

4. **Host features**: Host features are the attributes designed to assess attacks which last for more than two seconds.
   Features included: dst_host_count, dst_host_srv_count, dst_host_same_srv_rate , dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate

Data Preparation: The dataset synthesized for this project contains approximately 1lakh rows and 42 columns including label. As the project aims at detecting if the connection is normal or attack all the types of attacks are labelled as 'anomaly' and good connections are labelled as 'normal'.

Data Processing :

Feature Selection: Out of 41 independent variables, not relevant feature 'service' which states the service at the destination is removed. Correlation between the columns are checked Pearson correlation is applied.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

r = correlation coefficient

xi = values of the x-variable in a sample

$\bar{x}$ = mean of the values of the x-variable

yi= values of the y-variable in a sample

$\bar{y}$ = mean of the values of the y-variable

Columns with correlation greater than 0.95 are removed as higher correlation makes the model more sensitive. This reduces the dataset to 35 features.

Encoding: Furthermore, one hot encoding is applied to categorical variables. (protocol type and flag). The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are encoded using a one-hot (aka 'one-of-K' or 'dummy') encoding scheme. This creates a binary column for each category and returns a sparse matrix.

Train-test-split: The data is now ready for scaling. After the encoding we split the data into train set and test set.

Standardization: Standard Scaler function performs Standard Normal Distribution (SND).

$$Z=(x-u) / s$$

where u is the mean of the training samples and s is the standard deviation of the training samples.

3.2 Experimentation and Evaluation

Different classification model are experimented and evaluated for accuracy to get the basic idea of well performing models. The selected models include Naïve Bayes Bernoulli, KNN, Decision Tree, Naïve Bayes Gaussian , Logistic Regression. All the experimentation is performed on Jupyter notebook.

Hyperparameter Tuning: The selected models upon experimentation are subjected to tuning their respective parameters. RandomizedSearchCV is fed with the set of possible parameters. Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. RandomizedSearchCV implements a “fit” and a “score” method. .The parameters of the estimator used to apply these methods are optimized by cross-validated search over parameter settings.

Classification Model : The best performing models were decision tree and KNN.

KNN: K-Nearest Neighbor is a data mining supervised classifier. The output of the target variable is predicted by finding the k closest neighbor, by calculating the Euclidean Distance. It is a non-parametric classification technique which does not make any assumptions about underlying data.

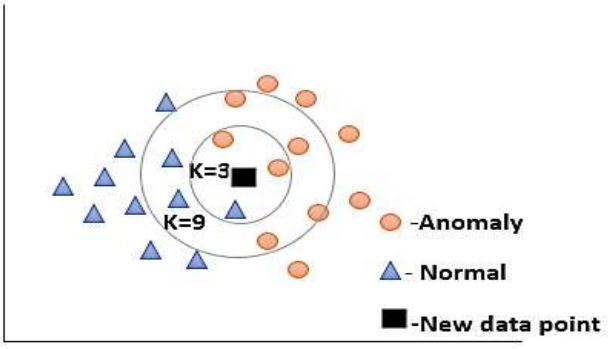

Figure 3.2 : KNN model classification

Decision Tree: Decision trees split the node based on information gain. The Decision tree classifies the given data item using the values of its attributes. The main approach is to select the attributes, which best divides the data items into their classes. According to the values of these attributes the data items are partitioned(figure 3.3). This process is recursively applied to each partitioned subset of the data items. The process terminates when all the data items in current subset belongs to the same class. A node of a decision tree specifies an attribute by which the data is to be partitioned.

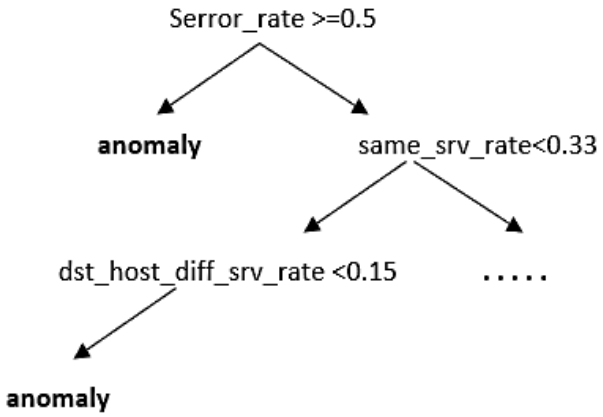

Figure 3.3: Sample decision tree

Hybrid Model: KNN and Decision Tree models are selected to ensemble for hybrid models. We have ensembled five KNN models with best parameters obtained from different iterations of random search as shown in figure(3.4) and one decision tree and two KNN models are ensembled for another hybrid model in figure(3.5).

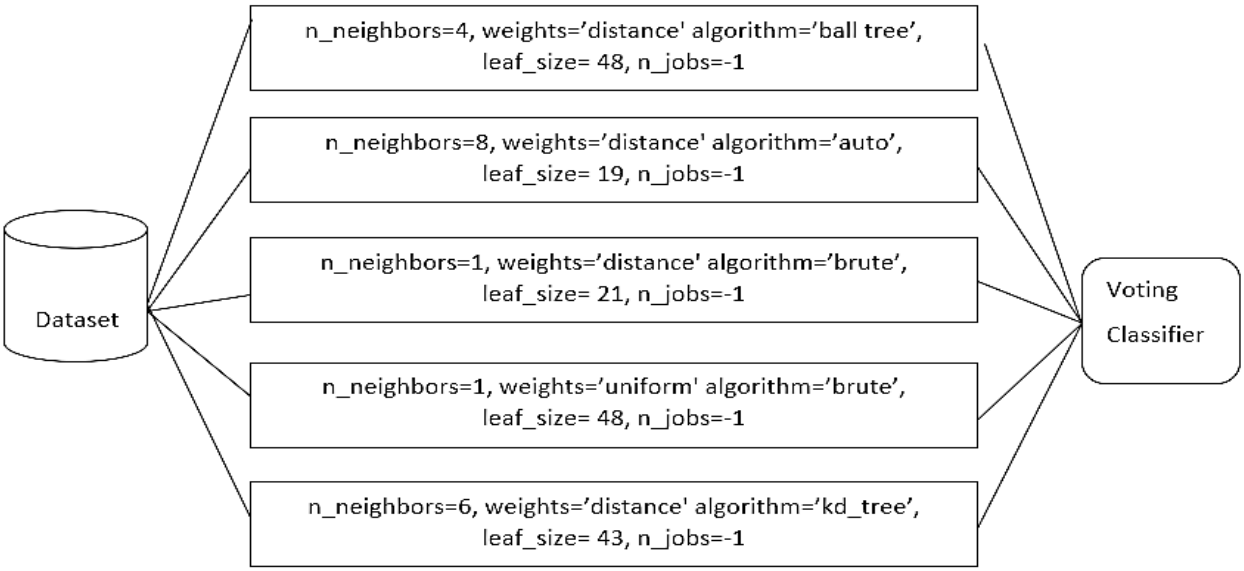

Figure 3.4: Hybrid model of 5 KNN with different parameters.

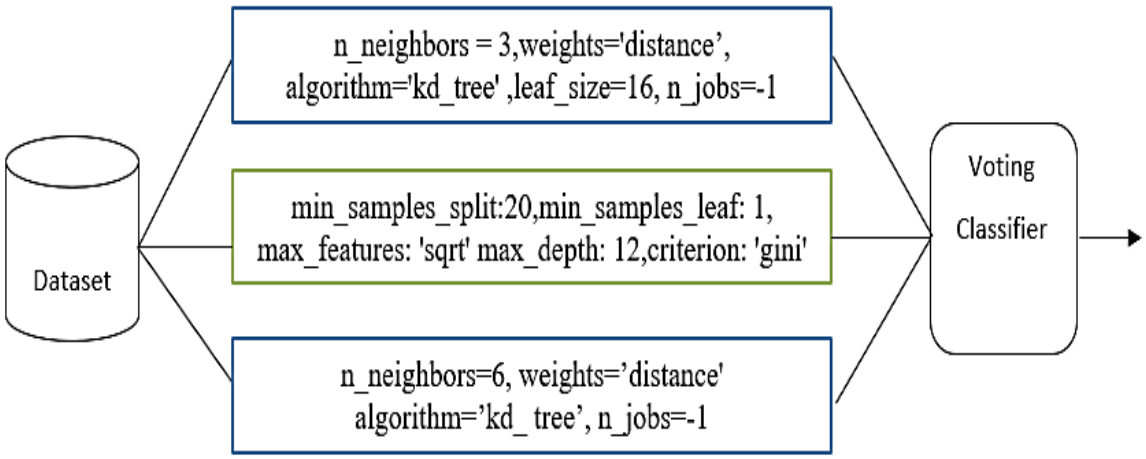

Figure 3.5: Hybrid model involving 1 Decision tree and 2 KNN

Voting Classifier: Voting classifier is applied with hard voting for output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting.

Hard Voting: Suppose three classifiers predicted the output class(anomaly, anomaly, normal), so here the majority predicted is 'anomaly'. Hence 'anomaly' will be the final prediction.

CHAPTER 4

RESULT ANALYSIS

Accuracy score plots of different models in comparison with the hybrid model is attained, classification report, confusion matrix of the test result, graphs and visualizations of the results is done for understanding and evaluating the performance of the model.

Accuracy Score: The accuracy score of different models are plotted in the figure 4.1. We observe that ensemble methods have yielded better accuracy than individual models. Amongst the hybrid models , the model involving decision tree has performed with higher accuracy rate.

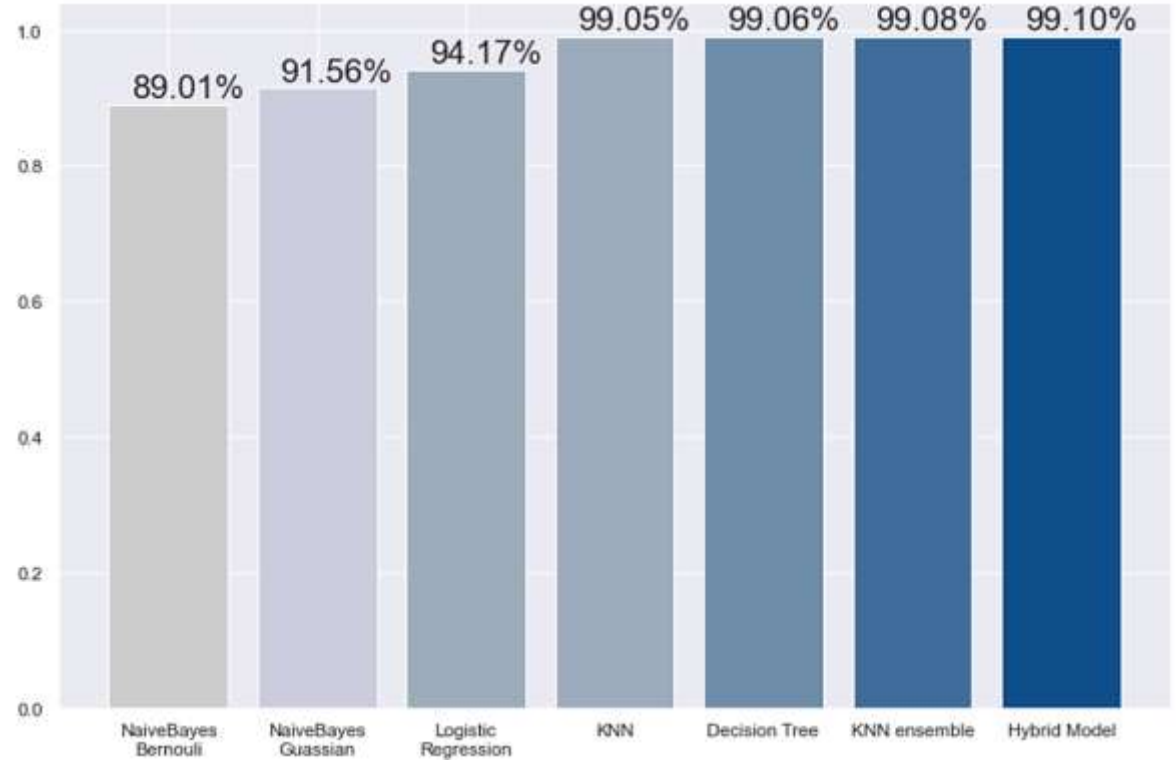

Figure 4.1: Plot of scores of different models

Classification Report: The dataset used for the experiment is unbalanced, hence accuracy, recall, precision, F1 score ,the receiver operating characteristics (ROC) curves, and area under curve (AUC) is used for evaluating the proposed method. Besides, we can visualize the relation between TPR and FPR of a classifier. These indicators can be expressed as

$$\text{Accuracy} = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{\text{TP}_i + \text{TN}_i}{\text{TP}_i + \text{TN}_i + \text{FP}_i + \text{FN}_i},$$

$$\text{Recall} = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i},$$

$$\text{Precision} = \frac{1}{|C|}\sum_{i=1}^{|C|}\frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i},$$

$$F1\ \text{score} = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}},$$

$$\text{AUC} = \frac{\sum_{i \in \text{positiveclass}} \text{rank}_i - (M(1+M)/2)}{M \times N}.$$

where M is the number of positive samples and N is the number of negative samples. The score indicates the probability that each test sample belongs to a positive sample, while rank is a positive sample set sorted in the descending order based on score.

Report of highest performing hybrid model(2 KNN and 1 Decision Tree) is as follows:

	precision	recall	F1-score	support
anomaly	0.9899	0.9911	0.9905	10465
normal	0.9921	0.9910	0.9915	11735
accuracy			0.9910	22200
Macro avg	0.9910	0.9910	0.9910	22200
Weighted avg	0.9910	0.9910	0.9910	22200

Cross Validation: 10-folder cross validation is used to calculate accuracy. The dataset is equally divided into 10 subsets, each of which is tested once and the rest as a training set. The cross validation is repeated 10 times, one subset is selected each time as a test set, and the average cross validation recognition accuracy rate of 10 times is found to be 99.36486 %.

Characteristic Curve: The plot of number of nearest neighbors versus the corresponding model scores is obtained as shown in figure 4.2.

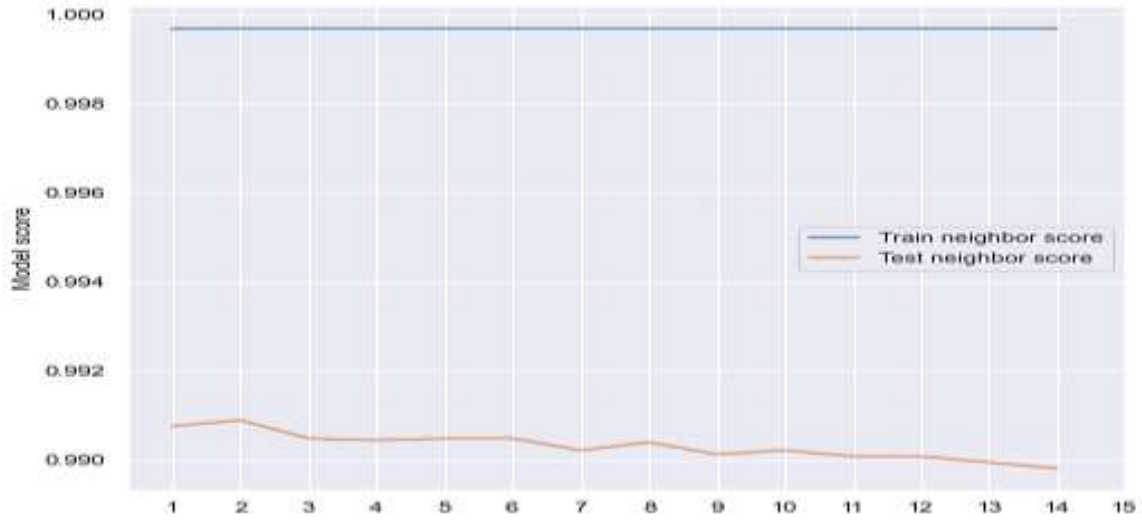

Figure 4.2 : Plot of KNN model with 1-15 nearest neighbors versus model score

The plot of minimum sample split of the decision tree versus accuracy score is plotted in figure(4.3). The plot gives interesting results. It shows a gradual decrease in the accuracy as the splits increases. But there is frequent spike in the graph along the minimum split. It is to be noted that other factors are kept constant.

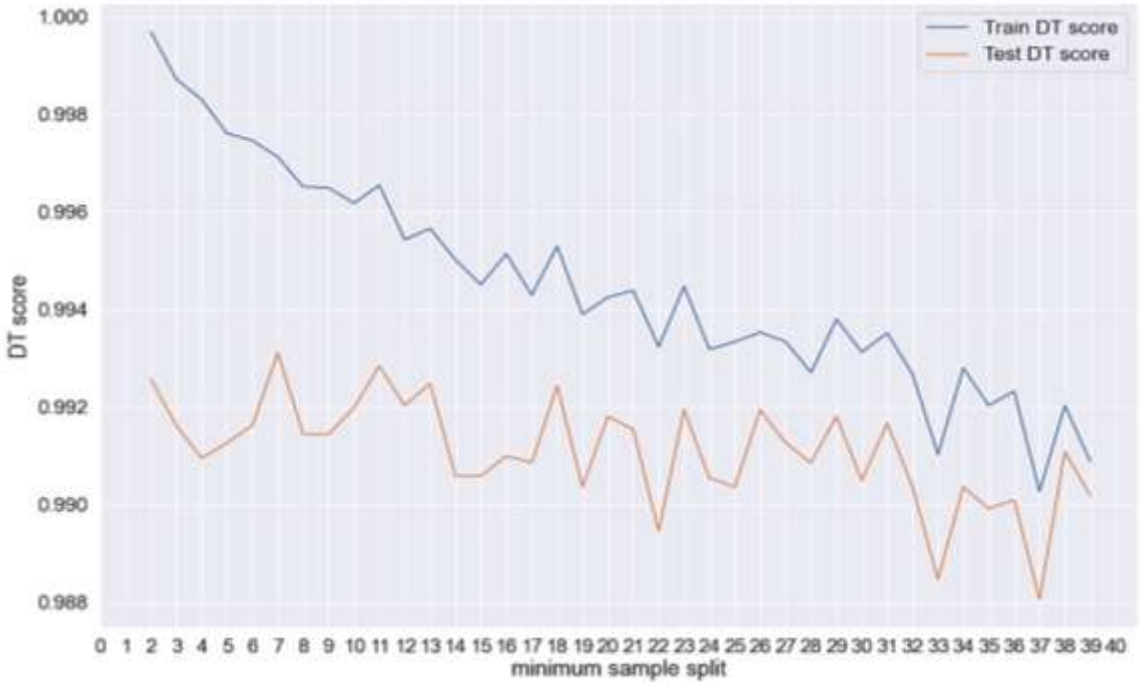

Figure 4.3: Characteristics of minimum sample split with respect to Decision tree

ROC Curve: It is the relation between TPR and FPR of a classifier. ROC curve of KNN and Decision tree are plotted in figure(4.4) and figure(4.5).

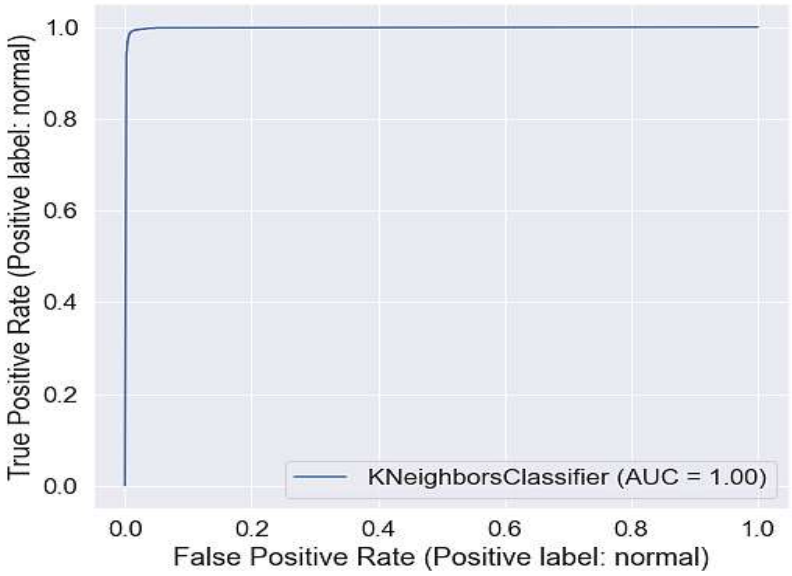

Figure 4.4 : KNN ROC Curve

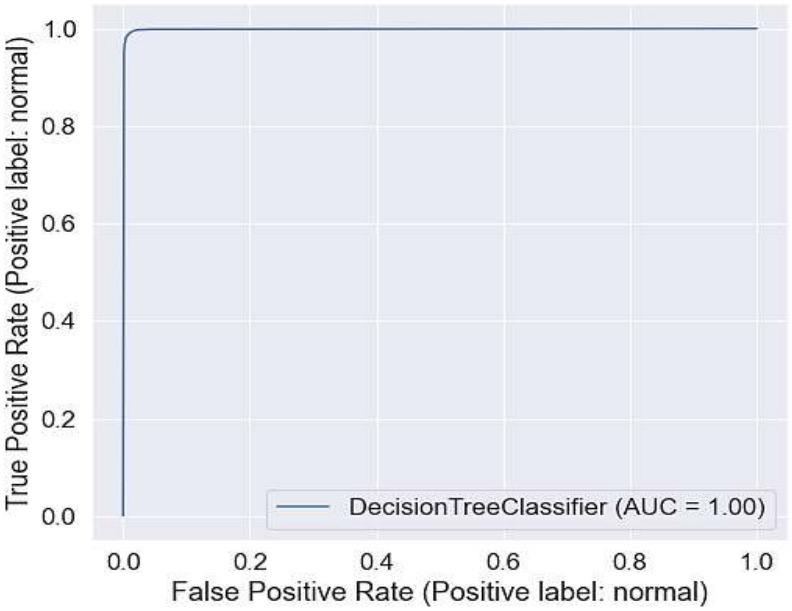

Figure 4.5: Decision Tree ROC Curve

Confusion Matrix: The performance of the hybrid model on the set of test data is depicted. The number of correct and incorrect predictions is summarized.

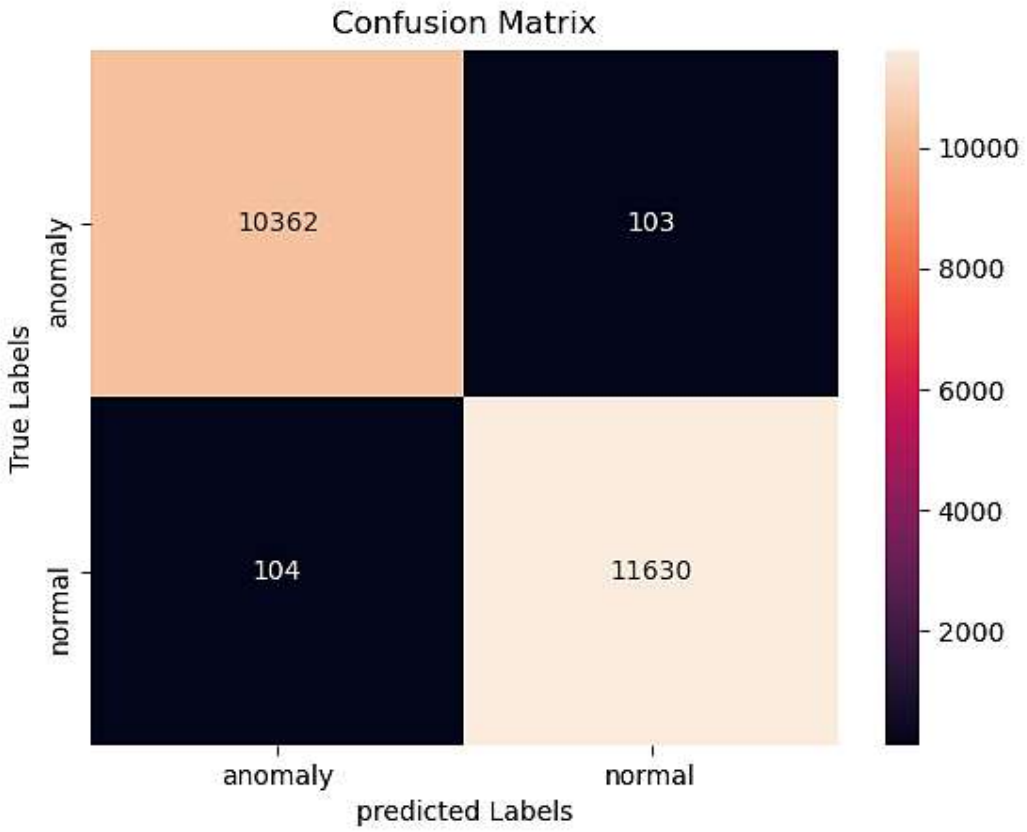

Figure 4.6: Confusion Matrix

We observe that 103 samples of malicious connections are predicted as normal and 104 normal connections are predicted as malicious.

CHAPTER 5

IMPLEMENTATION WITH GUI

The GUI implementation of the hybrid model is done using html, css, bootstrap, flask technologies with python as major language. Trained model is saved and used to predict new connections which are tuples of dataset held for validation.

5.1 Technologies

Html: The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

CSS: Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Bootstrap: Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project.

Flask: Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. Its features includes development server and debugger, integrated support for unit testing, RESTful request dispatching, uses Jinja templating, support for secure cookies (client side sessions), 100% WSGI 1.0 compliant, Unicode-based, extensive documentation, Google App Engine compatibility, extensions available to enhance features desired.

5.2 Implementation

The major sections include:

- Data folder : It contains dataset for testing purpose.
- Pickle folder: It contains trained model saved in the pickle format.
- Static folder: It contains CSS file, images and bootstrap framework.
- Templates folder: HTML templates (index.html and explore.html) is contained.
- app.py file: It is an app object, which is an instance of the Flask object. It will act as the central configuration object for the entire application. It is used to set up pieces of the application required for extended functionality.
- model.py file: Saved model is loaded.
- test.py file: It contains logic for testing the dataset whose results are displayed in explore.html

Virtual environment is setup for the project which includes necessary libraries and packages.

The algorithm of the finalized hybrid-model training steps and functionalities implemented are as follows:

hybrid_model.ipynb

import libraries and packages
initialize column header of dataset
data = read dataset
extract column names of numerical datatype
extract categorical data columns
perform correlation matrix of the data
extract upper triangular matrix of correlation matrix

to_drop = loop upper triangular matrix for columns with correlation> 0.95
drop columns from data in to_drop
drop service column
predictors = independent columns of data
target = dependent/ target column
load preprocessing.OneHotEncoder()
apply fit and transform on categorical data columns'
rename the labels of onehotencoded array
concatenate to predictors
perform train test split
perform standardization on train(fit_transform) and test(transform) set
ensemble = initialize VotingClassifier with voting = hard
train ensemble with x_train, y_train
save the model in pickle format
save standard scaler in pickle format

app.py

import numpy
import pandas
import model.py and test.py
from flask import Flask, request, jsonify, render_template, redirect, send_file
initialise variables
app = create instance of flask

@app.route('/') -decorate the initial route
def home():
    return index.html

@app.route('/predict')-decorate the predict route
def predict():
    get the form input string
    split the string into features array
    pop the features that were removed with correlation <0.95

```
        convert the string input features into respective datatype
        // initialize protocol_array, flag_array for onehotencoding
        protocol_array = protocol_onehot(variable)
        flag_array = flag_onehot(variable)
        concatenate arrays with changed datatype and respective onehot encoded matrix
        apply standardisation
        predict
        if (output ==’anomaly’)
                res = ‘The connection is Malicious’
        else:
                res =’The connection is normal’
        return index.html with res

def protocol_onehot(key):
        define dictionary of protocol type and corresponding sparse matrix
        if key in dictionary:
                return matrix

def flag_onehot(flag):
        define dictionary of flag and corresponding sparse matrix
        if flag in dictionary:
                return matrix

@app.route(‘/explore’)-decorate the explore route
def explore():
        return render explore.html

@app.route(‘/back’)-decorate the back route
def back():
        return render index.html

@app.route(‘/test’)-decorate the test route
def test():
```

```
        test.score = score_test()
        return render explore.html with test.score

@app.route('/figure')-decorate the figure route
def figure():
        val = display_confusion()
        if val ==1:
                return render explore.html with score and confusion matrix image

@app.route('/report')-decorate the results route
def report():
        cr =display_report()
        if cr ==1:
                return render explore.html with score, confusion matrix  and classification
                report image

if __name__ == "__main__":
        run app
```

model.py

```
import pickle
sc = load saved standard scaler
model = load trained hybrid model
print "model ready"
```

test.py:

```
import sys
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split , KFold
from sklearn. preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import VotingClassifier
from sklearn import model_selection
from sklearn import metrics
import pickle
sc = load saved standard scaler
model = load trained hybrid model
initialise column headers
read x_test.csv
read y_test.csv
y_preds = predict(x_test)

def score_test():
        score = score for x_test, y_test
        return score

def display_confusion():
        initialize plt.figure()
        get confusion matrix for y_test, y_preds
        draw plot by initialzing sns.heatmap
        set labels, title, ticklabels
        save the plot
        return 1
```

```
def display_report():
        initialize plt.figure()
        get classification report for y_test, y_preds
        draw plot by initialzing sns.heatmap
        save the plot
        return 1
```

5.3 Output

Index page

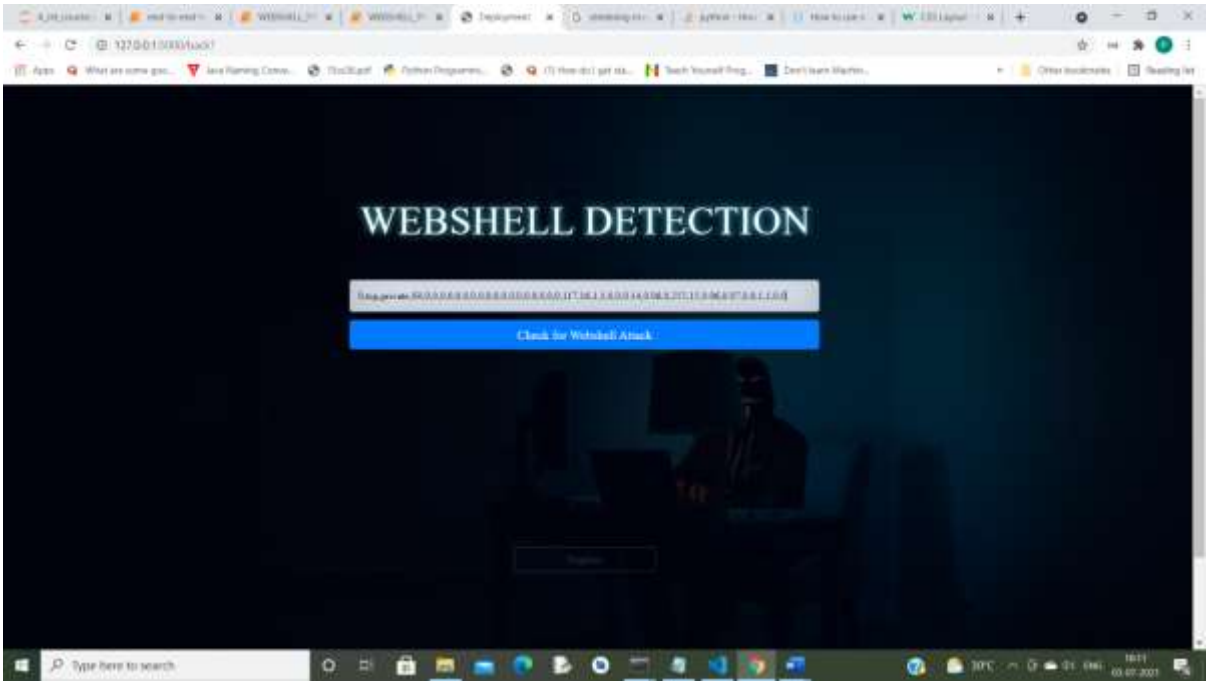

Figure 5.1 : Entering the input string

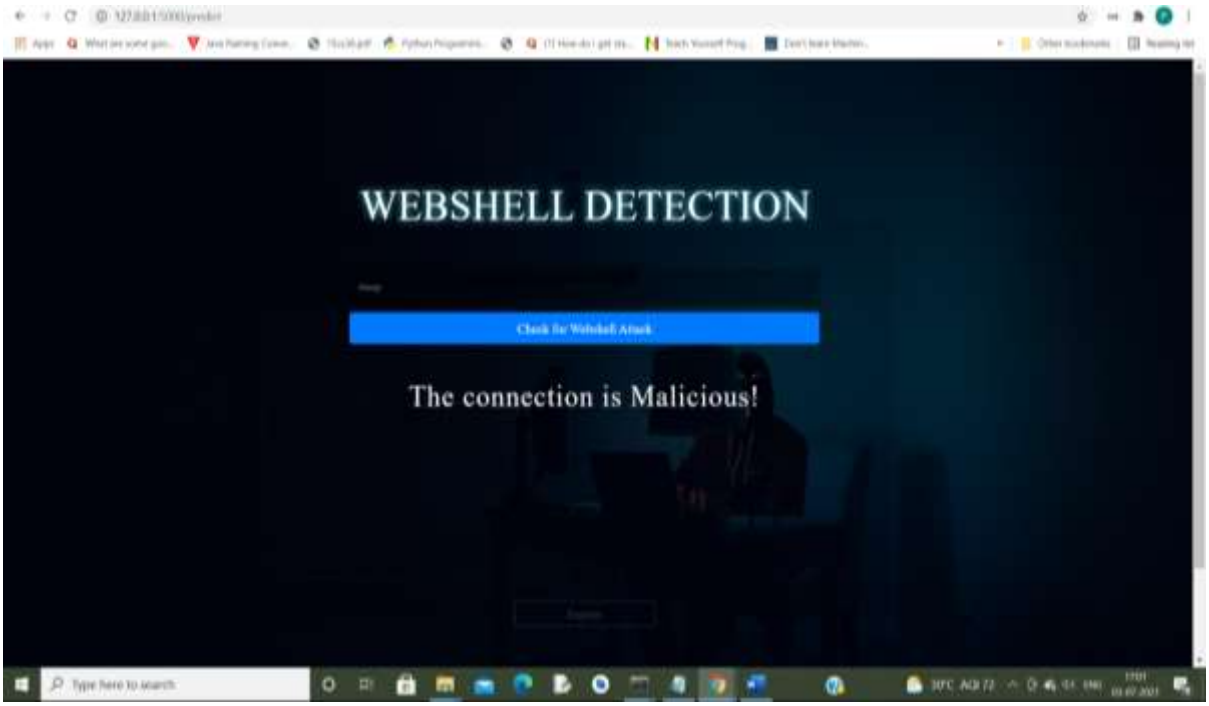

Figure 5.2: Predicted output

Explore Page

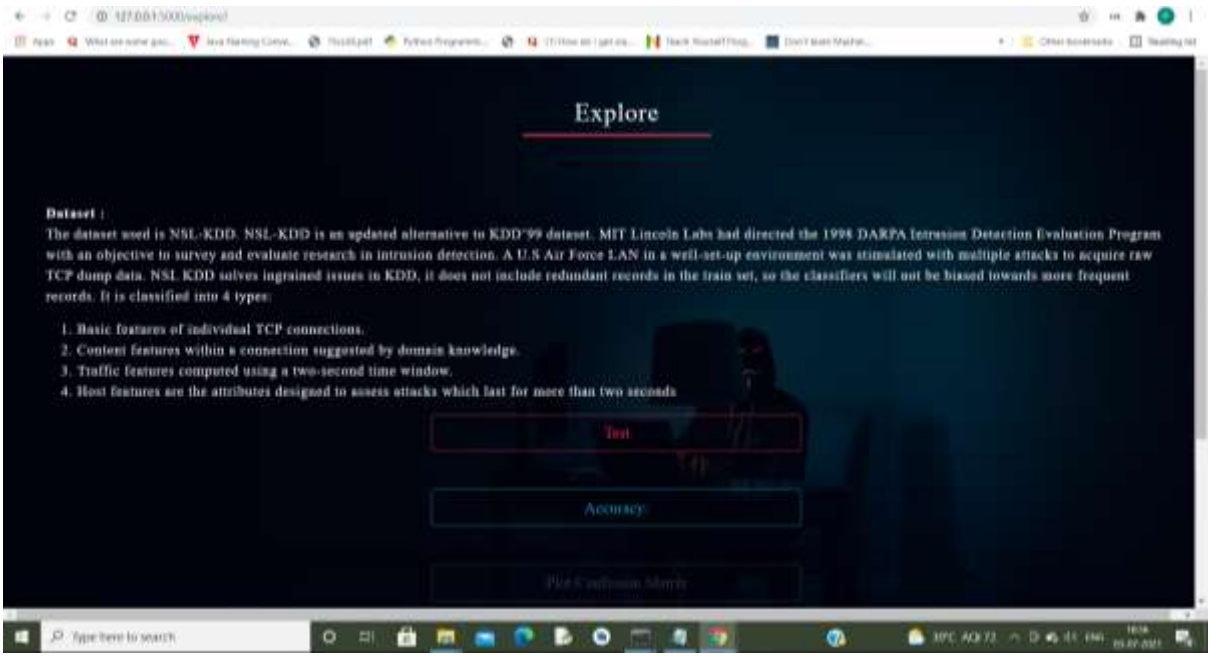

Figure 5.3: Explore Page View

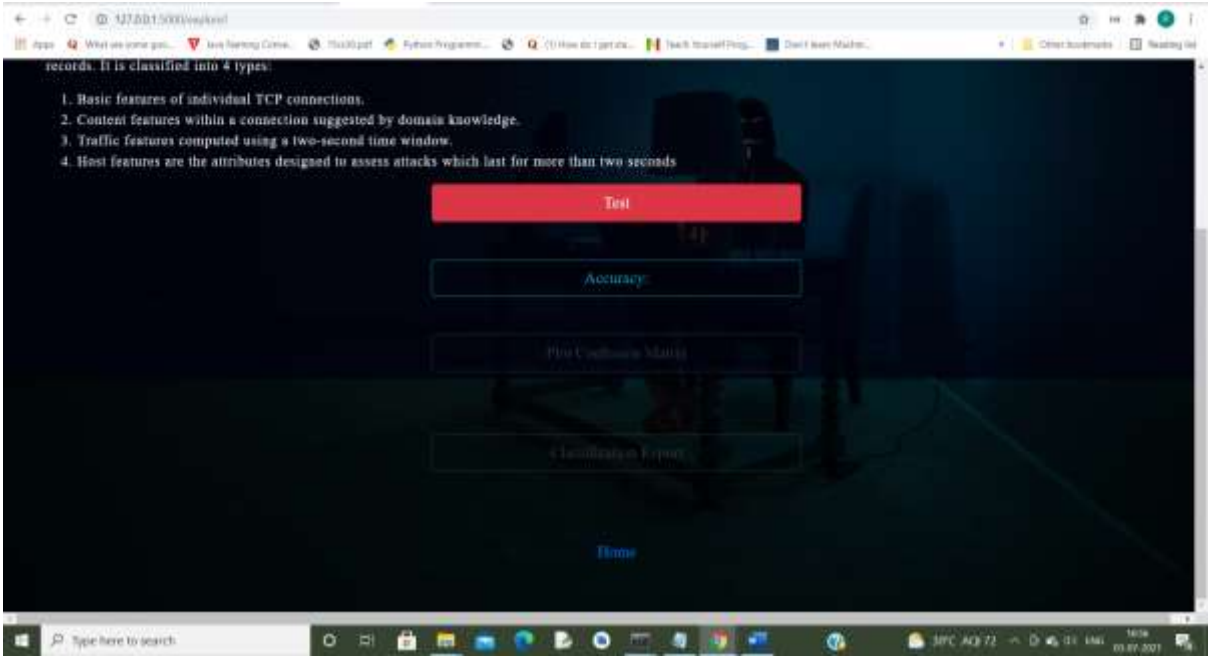

Figure 5.4: Performing Test

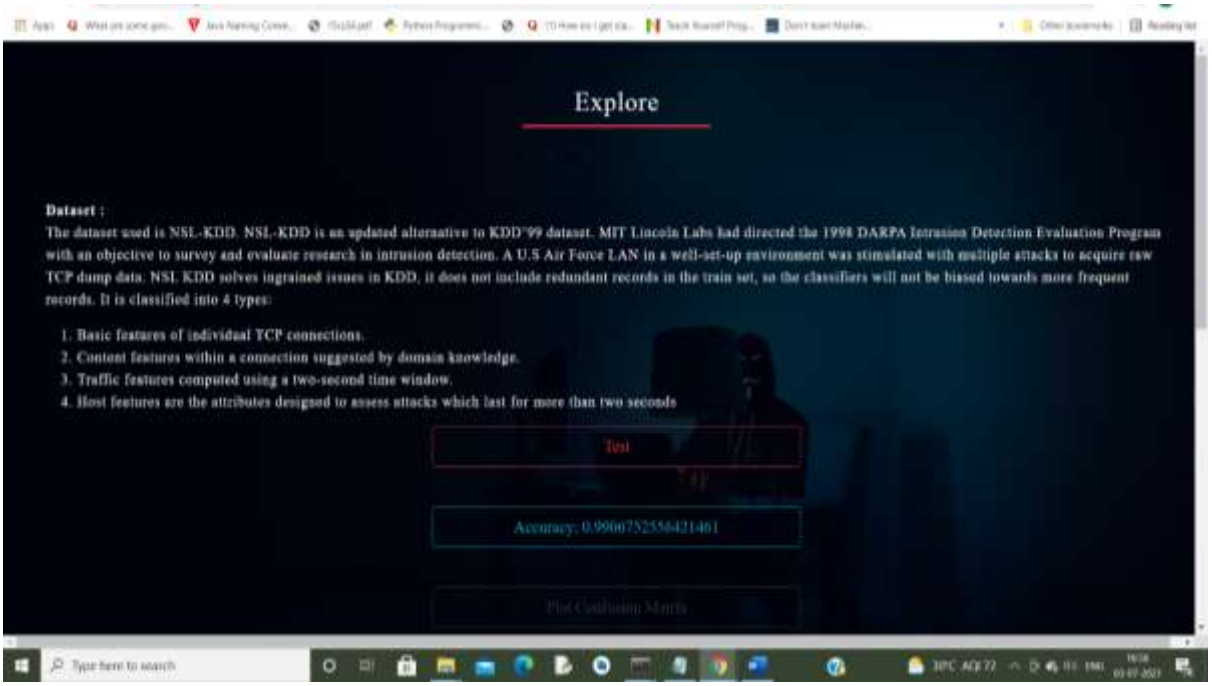

Figure 5.5: Accuracy score displayed

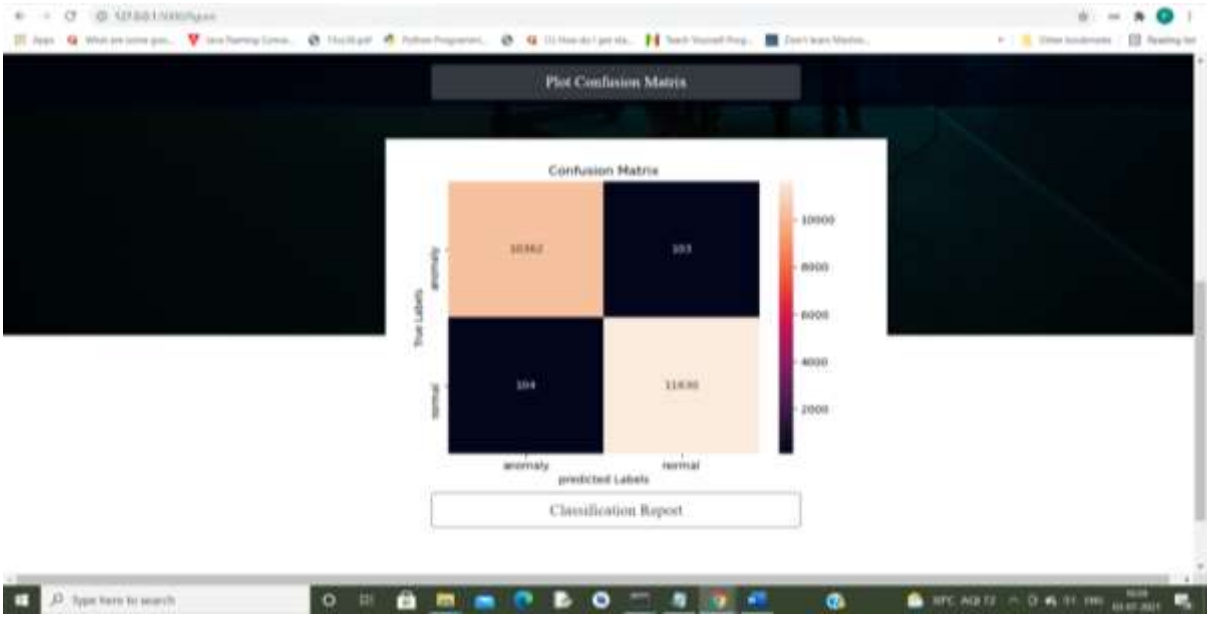

Figure 5.6 : Confusion Matrix Displayed

Figure 5.7 : Display of Classification Report

CHAPTER 6

CONCLUSION AND FUTURE WORK

According to the study and observation we can see the potential of machine learning algorithm over traditional methods that are used by anti- virus tools for web shell detection. And we have also discussed about different machine learning algorithms that can be of great help in detecting web shell as with the quick development and advancement of web, web shell is major threat.

In this project, we applied different algorithms to NSL-KDD data set to classify normal and anomaly connections. Proposed method is validated by 10 fold cross validation for the classification. The Experimental analysis shows that when compared to other classification methods, proposed hybrid model have increased the accuracy, precision, recall and f-measure values. In future, we plan to apply optimization techniques for the classification of the dataset.

CHAPTER 7

PUBLICATION DETAILS

Dr. Chandrika J, Megha D, Pooja K P, Roja Mallik S P, Syeda Shafain Fathima, A Novel Machine Learning based Hybrid Model for Webshell Detection, International Journal for Research in Applied Science & Engineering Technology (IJRASET), ISSN:2321-9653; IC Value:45.98; SJ Impact Factor:7.429, DOI: https://doi.org/10.22214/ijraset.2021.35644

CHAPTER 8

REFERENCES

[1]Weekly Report of CNCERT [CrossRef]

[2]Article, Dave McMillen(2016) in Security Intelligence "Got WordPress? PHP C99 Webshell Attacks Increasing" [CrossRef]

[3] You Guo, Hector Marco-Gisbert and Paul Keir(2020) Mitigating Webshell Attacks through Machine Learning Techniques [CrossRef]

[4] Cybersecurity Information (2020). CSI- Detect and Prevent Web Shell Malware[CrossRef]

[5] Iglesias, Félix; Zseby, Tanja (2015). Analysis of network traffic features for anomaly detection. Machine Learning, 101(1-3), 59–84. doi:10.1007/s10994-014-5473-9 [CrossRef]

[6] Firdausi, Ivan; lim, Charles; Erwin, Alva; Nugroho, Anto Satriyo (2010). [IEEE 2010 Second International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT) - Jakarta, Indonesia (2010.12.2-2010.12.3)], Second International Conference on Advances in Computing, Control, and Telecommunication Technologies - Analysis of Machine learning Techniques Used in Behavior-Based Malware Detection. , (), 201–203. doi:10.1109/ACT.2010.33 [CrossRef]

[7] Zhuang Ai , Nurbol Luktarhan , AiJun Zhou and Dan Lv (2020). WebShell Attack Detection Based on a Deep Super Learner [CrossRef]

[8] Yixin Wu, Yuqiang Sun, Cheng Huang , Peng Jia, and Luping Liu (2019) Session-Based Webshell Detection Using Machine Learning in Web Logs [CrossRef]

[9] Muataz Salam Al-Daweri, Khairul Akram Zainol Ariffin, Salwani Abdullah, Mohamad Firham Efendy Md. Senan (2020). An Analysis of the KDD99 and UNSW-NB15 Datasets for the Intrusion Detection System [CrossRef]

Name of the Student (USN, DOB, Email id)	**Address for Communication (Permanent Address)**	**Stamp size recent color photo**
MEGHA D 4MC17CS031 18/06/1999 meghadhsn1999@gmail.com	#25 , Hentagere Road, Ragibychanahalli, Arkalgud , Hassan- 573102	
Pooja KP 4MC17CS033 07/05/1999 poojanhaviputtur@gmail.com	Pooja Nilaya, Sampya Moole, Kuriya village, Aryappu Post, Puttur, D.K- 574210	
ROJA MALLIK SP 4MC17CS042 12/02/2000 rojamallik00@gmail.com	"Amma" 1st main, 2nd cross, Shankaracharya Road ,Vidhyanagar, Hassan-573201	
SYEDA SHAFAIN FATHIMA 4MC17CS057 11/12/1998 shafainsyeda@gmail.com	#285 Shariff colony, HN Pura Road, Hassan- 573201