A PROJECT REPORT ON

DETECTION OF PHISHING WEBSITES

Submitted in partial fulfilment of the requirements

For the Award of Degree of
BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

Submitted By

P.SUBASH CHANRDA BOSE	20B21A4579
Y.SATYA SAI SUBRAHMANYAM	20B21A4566

Under the Esteemed Guidance of

MRS. RAYUDU UMA MAHESHWARI, (M.Tech)
Associate Professor

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

KAKINADA INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Approved by AICITE & Affiliated to JNT University Kakinada) Yanam Road, Korangi-533461 E.G Dist(A.P).

Phone No :0884-234050,2303400 fax No: 0884-2303869 2020-2024

KAKINADA INSTITUTE OF ENGINEERING & TECHNOLOGY FOR WOMEN

(Approved by AICITE & Affiliated to JNT University Kakinada)

Yanam Road, Korangi-533461 E.G Dist(A.P).

Phone No :0884-234050,2303400 fax No: 0884-2303869

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project entitled “DETECTION OF PHISING WEBSITES” that is being submitted by **punnam subhash chandra bose(20B21A4579),yalla satya sai subrahmanyam (20B21A4566)**partial fulfilment of the requirements for the award of the Degree

Of Bachelor of Technology in **“Computer Science & Engineering“** in the academic year 2018-2022 to Kakinada Institute of Engineering And Technology affiliated to **Jawaharlal Nehru Technological University Kakinada,** is a record of bonafide work carried out by him/her under my guidance and supervision.

The results embodied in this project have not been submitted to any other University or Institute for the award of any degree or diploma.

INTERNAL SUPERVISOR **EXTERNAL EXAMINAR** **HEAD OF THE DEPARTMENT**

DECLARATION

I hereby declare that the project work entitled “DETECTION OF PHISHING WEBSITES” submitted to **Kakinada Institute of Engineering and Technology** is a record of original done by us under the guidance of **Mrs. R. Uma Maheshwari, (**M.Tech), Associate Professor in Department of **Computer Science &**

Engineering and this project work submitted in the partial fulfillment of requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering**. The results embodied in this thesis have not been submitted to any other university or institute for the award of any degree or diploma.

We further declare that we or any other person has not previously submitted this project report to any other institution or university for any other degree or diploma.

Place:

Date:

ACKNOWLEDGEMENT

We would like to take the privilege of the opportunity to express our gratitude

P.SUBHASH CHANDRA BOSE	20B21A4579
Y.SATYA SAI SUBRAHMANYAM	20B21A4566

in to the project work of "**DETECTION OF PHISHING WEBSITES**" enabled us to express our special thanks to our honorable Chairman of the institution **P.V. VISWAM** Sir.

We are thankful to our honorable Principal **Prof. Y. RAMA KRISHNA, (M. Tech)** who has shown keen interest in us and encouraged us by providing all the facilities to complete our project successfully.

We owe our gratitude to our Beloved head of the Department CSE
Mr. S. DEVANAM PRIYA, (M. Tech), for assisting us in completing the project work.

We would like to express our special thanks to our Project coordinator

Mrs. UMA MAHESHWARI, (M. Tech), for guiding us through this project.

We wish to express our sincere Thanks to all teaching staff of computer science and engineering .We wish to express our special Thanks to all the Faculty members of our college of their concern in subjects and their help throughout our course.

We are the Thankful to our parents and all our friends who had given us good cooperation and suggestions thought our project and helped us successful compilation.

P.SUP.SUBHASH CHANDRA BOSE(20B21A4579) Y.SATYA SAI SUBRAHMANYAM (20B21A4566)

ABSTRACT

Throughout the years there have been numerous assaults of Phishing and numerous individuals have lost tremendous totals of cash by turning into a casualty of phishing assault. In a phishing assault messages are sent to client professing to be a real association, where in the email requests that client enter data like name, phone, financial balance number significant passwords and so forth such messages direct the client to a site where in client enters these individual data. These sites otherwise called phishing site currently take the entered client data and does unlawful exchanges in this way making hurt the client. Phishing site and their sends are sent to a huge number of clients day by day and hence are as yet a major worry for digital security. This venture utilizes Machine-learning system for displaying the forecast task and managed learning calculations to be specific Decision tree acceptance, Naive Bayes characterization and Random Forest are utilized for investigating the outcomes.

INDEX

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Term/Abbreviations	**Definition**
UCI	UC Irvine
XGBOOST	extreme Gradient Boosting
SVM	Support vector machine
GUI	Graphical user interface

NN	Neural Networks
RF	Random Forest

a. INTRODUCTION

1.1 INTRODUCTION OF PROJECT

Phishing is an online wrongdoing that attempts to deceive unsuspected clients to uncover their touchy (and significant) individual data, for instance, usernames, passwords, financial account subtleties, street numbers, SSN, and social connections, to the heel, of ten for noxious reasons. Phishing is a type of extortion wherein the aggressor attempts to learn touchy data, for example, login accreditations or record data by sending as a respectable element or individual in email or other correspondence channels.

Normally a casualty gets a message that seems to have been sent by a known contact or association. The message contains malevolent programming focusing on the client's PC or has connections to guide casualties to vindictive sites

so as to fool them into disclosing individual and budgetary data, for example, passwords, account IDs or charge card subtleties.

Phishing is famous among aggressors, since it is simpler to fool somebody into clicking a malignant connection which appears to be real than attempting to get through a PC's protection frameworks. The malevolent connections inside the body of the message are intended to cause it to create the impression that they go to the parodied association utilizing that association's logos and other real substance

Phishing area (or Fraudulent Domain) attributes, the highlights that recognize them from genuine areas, why it is imperative to identify these spaces, and how they can be identified utilizing AI and characteristic language preparing strategies

1.2 PROBLEM STATEMENT

Phishing is a form of fraud in which the attacker tries to learn sensitive information such as login credentials or account information by sending

URL‟s to email or other communication channels. Typically a victim receives a message that appears to be sent by a known contact or organization. The message contains malicious software targeting the user's computer or have a link that directs victims to malicious websites in orderto trick them, into personal and financial information, such as passwords, account IDs or credit card details.

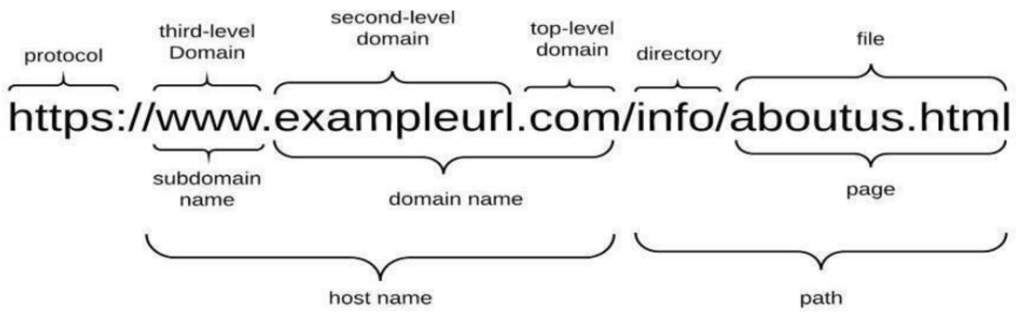

Fig 1.2: URL Structure

1.3 PROJECT OBJECTIVES

Our work in this thesis focuses mainly on detecting phishing websites with machine Learning. There has been very some exertion seeing comparable subjects, for example, Pernicious space boycotting and email spam filtering. Moreover, it is progressively well known to use AI in these regions. Existing malevolent sites identification at preaches can be mainly divided into two categories based on the features leveraged: static feature feature- based approaches and dynamic feature- based approaches. Static highlight put together approaches depend with respect to highlights extricated from the URL, page content, HTML DOM structure, space- based data, (for example, WHOIS and DNS records) etc. Then again, dynamic element put together solution basically centre with respect to analysing behaviours captured when the page is load dan rendered, or investigating system logs when a few contents are executed. Right now, focus on abusing static highlights.

1.3.1 DATA DESCRIPTION

Sticking with simplicity, we will use a Random Forest Classifier (RF) from scikit- learn. For training the classifier, we split the data into 90% training and 10% testing. No cross- validation is done since we are not trying to extensively tune any hyper-parameters. We will stick with the default hyper parameters of Random Forest from the scikit-learn implementation. Contrary to deep learning models that take a long time to train, RF takes less than 2 minutes on a CPU to train and demonstrate effective results as are shown next. To show robustness in performance, we train the model 5 times on different splits of data and report average test results

1.3.2 FILE DESCRIPTION

The features represented by the training dataset can be classified into four categories;

1 Address Bar based features

2 Abnormal based features

3 HTML and JavaScript based features

4 Domain based features

CSV FILES:

Phishing.csv-999 rows, 17 columns.

legitimate.csv - 870 rows, 17columns

File Name	File Description
Train.csv	Constraint train data
Test.csv	Constraint test data

Table 1.3.2: Data set files

1.4 PROJECT REPORT ORGANIZATION

In this “Project Report” a detailed description about the design challenges, proposed methodologies and the implementation of application to solve the real- world problem is given. Different functionalities of the application are broken down into modules and explained with the help of use case diagrams and class diagrams. The working model of the application is shown using screenshots in this report.

This report is organized into six chapters. After this introductory chapter,

Chapter 2: Describes about the literature survey, characteristics and design challenges of the existing system and provides a proposed solution.

Chapter 3: This chapter defines about the software requirements which include functional requirements, non-functional requirements, system architecture and system specifications which include software requirements.

Chapter 4: It gives a description about the UML diagrams like *use- case diagrams, class diagrams, activity diagrams, architecture diagrams.*

Chapter 5: In this chapter, implementation and testing is discussed in detail.

Chapter 6: Focuses on providing the conclusion and defines the future scope of the application being developed.

1.3LITERATURE SURVEY

2.1 EXISTING SYSTEM

This is most commonly used approach in which list of Phishing URL is stored in database and then if URL is found in database, it is known as Phishing URL otherwise it is called legitimate. This approach is easy and faster to implement as it checks whether URL is present in database or not.

2.2 LIMITATIONS OF EXISTING WORK

Features Used for Phishing Domain Detection:

There are a lot of algorithms and a wide variety of datatypes for phishing detection in the academic literature and commercial products. A phishing URL and the corresponding page have several features which can be differentiated from a maliciousURL. For example, an attacker can register long and confusing domain to hide the actual domain Name (Cybersquatting, Typo squatting). In some cases, attackers can use direct IP addresses instead of using the domain name. This type of event is out of our scope, but it can be used for the same purpose. Attackers can also use short domain names which are irrelevant to legitimate brand names and do not have any Free URL addition. But these type of web sites are also out of our scope because they are more relevant to fraudulent domains instead of phishing domains.

Beside URL-Based Features, different kinds of features which are used studies are used. Features collected from academic studies for the phishing domain detection with machine learning techniques are grouped as given below.
URL-Based Features Domain-Based Features Page-Based Features Content-Base Features

Presence of IP address in URL, length of URLs, having “@” symbol or not, etc. The first thing to do after the datasets are obtained is data slicing. Here, we divide the datasets into two parts: testing dataset and training dataset. The training dataset is used to train a model. The testing dataset is only used once the trained model is ready. Once the model is trained, we test its accuracy on the testing dataset.We are going to perform the classification task on the datasets. We make use of algorithm to perform the classification task:Random forest tree algorithm.

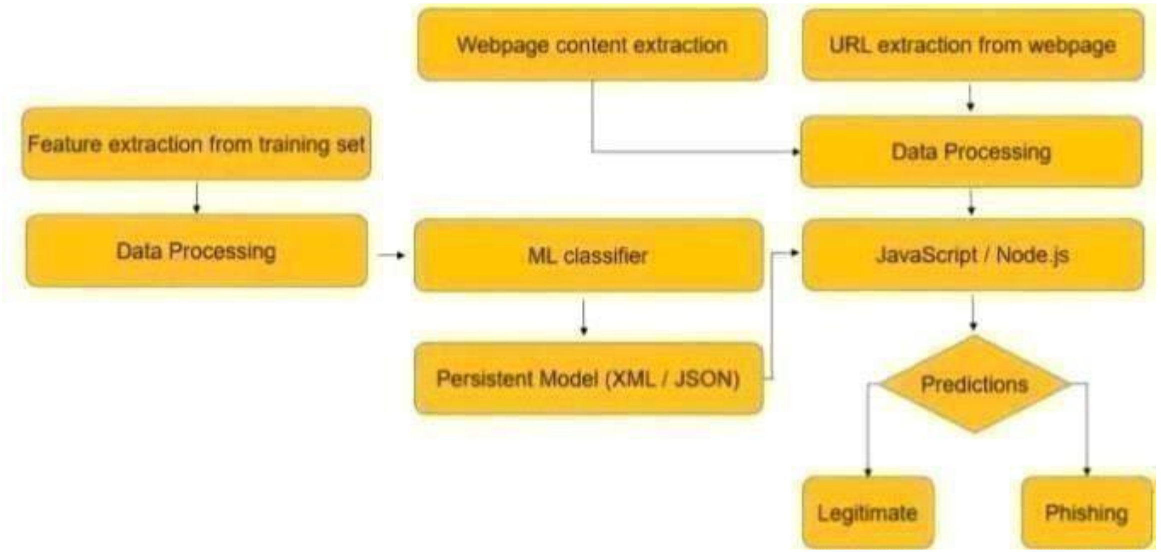
Webpage content extraction
URL extraction from webpage
Feature extraction from training set
Data Processing
Data Processing
ML classifier
JavaScript / Node.js
Persistent Model (XML / JSON)
Predictions
Legitimate
Phishing

Fig 2.3: Proposed phishing detection

1.4 SYSTEM SPECIFICATIONS AND COMPONENTS

1.4.1 Hardware Requirements

Minimum requirements:

4GB RAM

8GB HDD

Intel 1.66 GHz Processor Pentium 4

1.4.2 Software Requirements:

Python 3

Windows 7 (or) above

1.5 PROPOSED SYSTEM DESIGN

4.1 ARCHITECTURE DESIGN

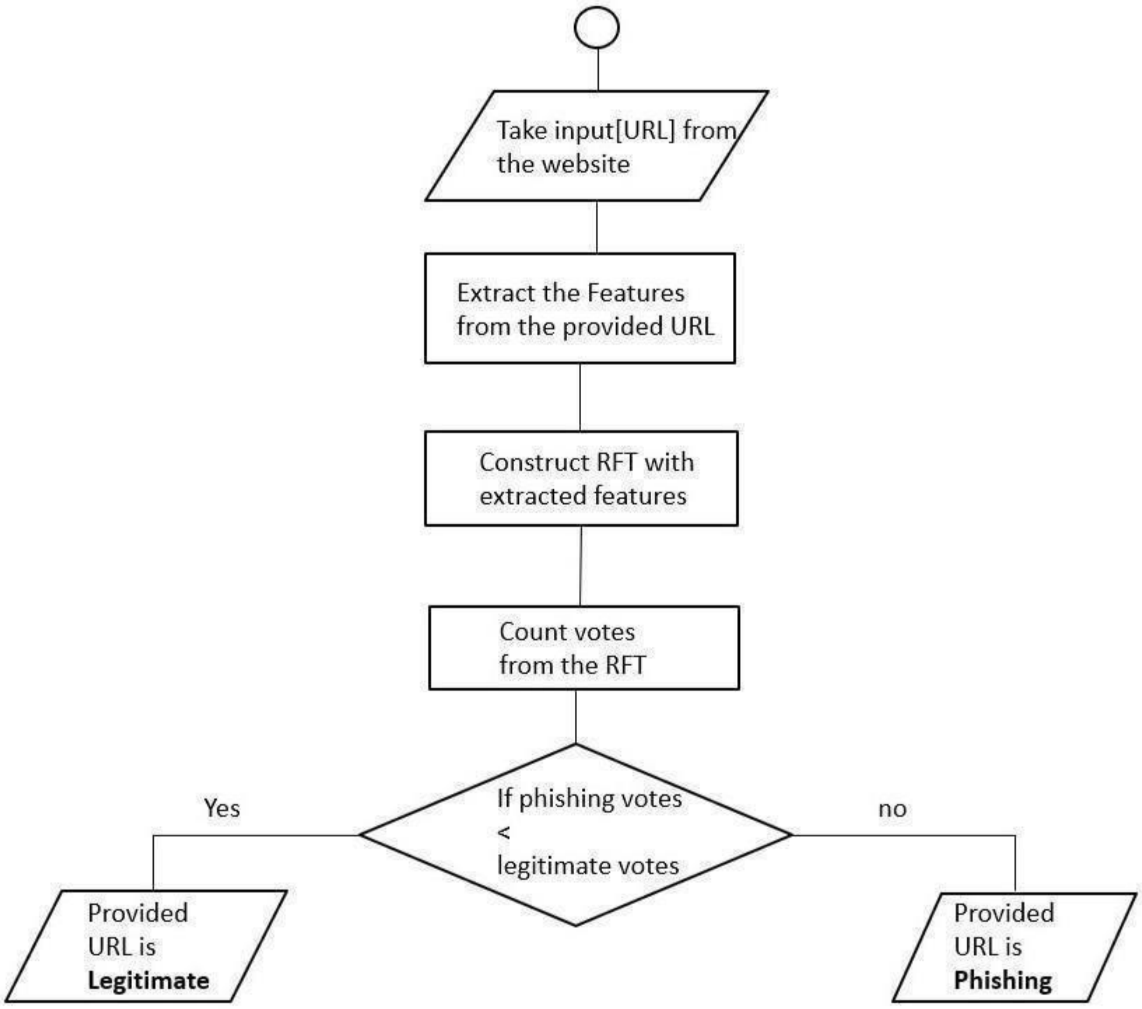
Take input[URL] from the website
Extract the Features from the provided URL
Construct RFT with extracted features
Count votes from the RFT
If phishing votes < legitimate votes
Yes
no
Provided URL is **Legitimate**
Provided URL is **Phishing**

Fig 4.1: Architecture Design

4.2 USECASE DIAGRAM

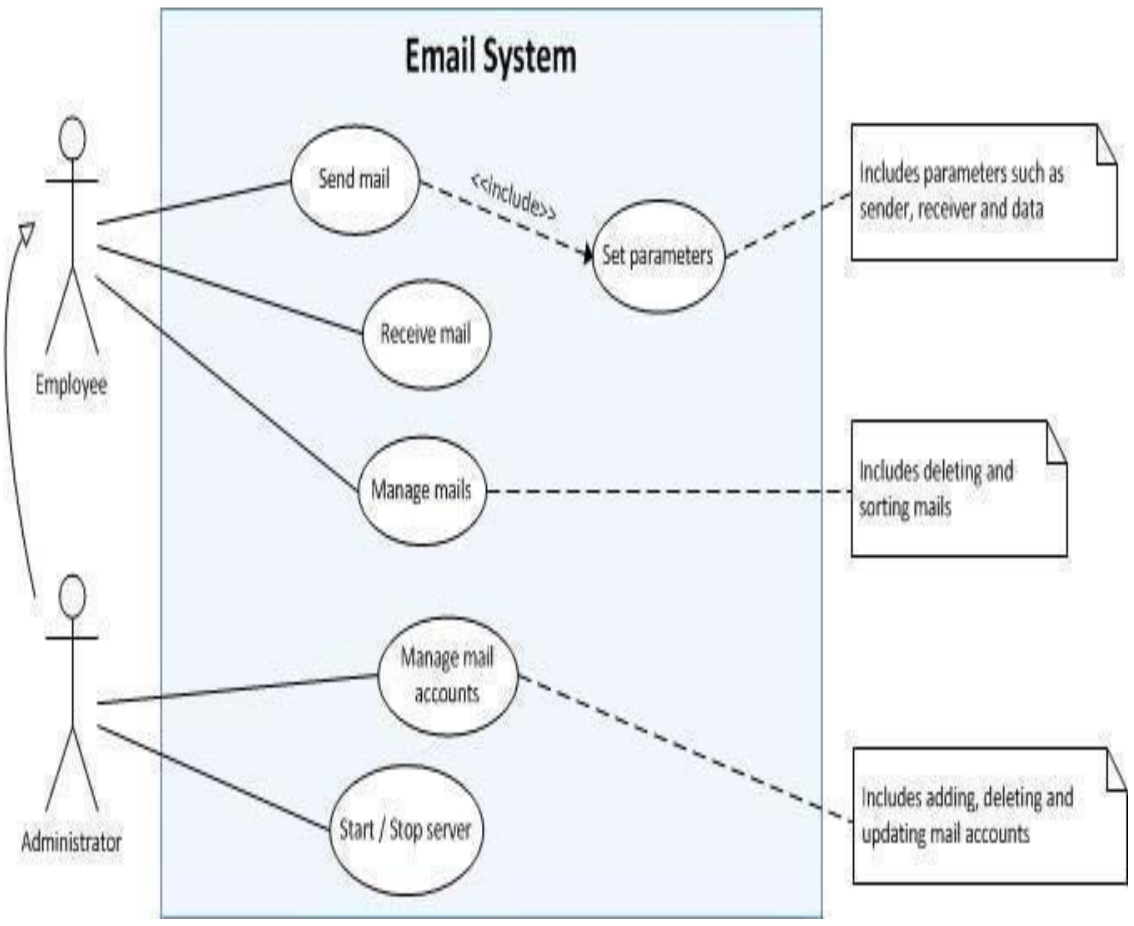
Email System
Employee
Administrator
Send mail
<<include>>
Set parameters
Receive mail
Manage mails
Manage mail accounts
Start / Stop server
Includes parameters such as sender, receiver and data
Includes deleting and sorting mails
Includes adding, deleting and updating mail accounts

Fig 4.2: UseCase Diagram

4.3 CLASS DIAGRAM

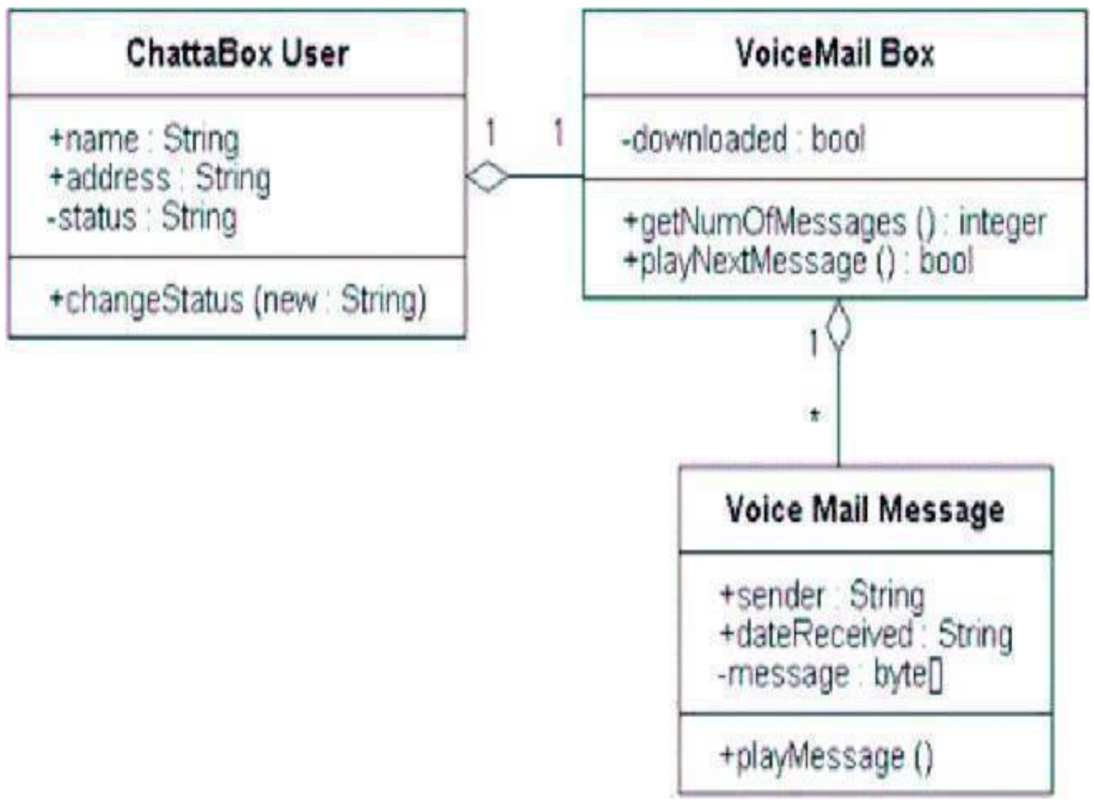

Fig 4.3: Class Diagram

4.4 SEQUENCE DIAGRAM

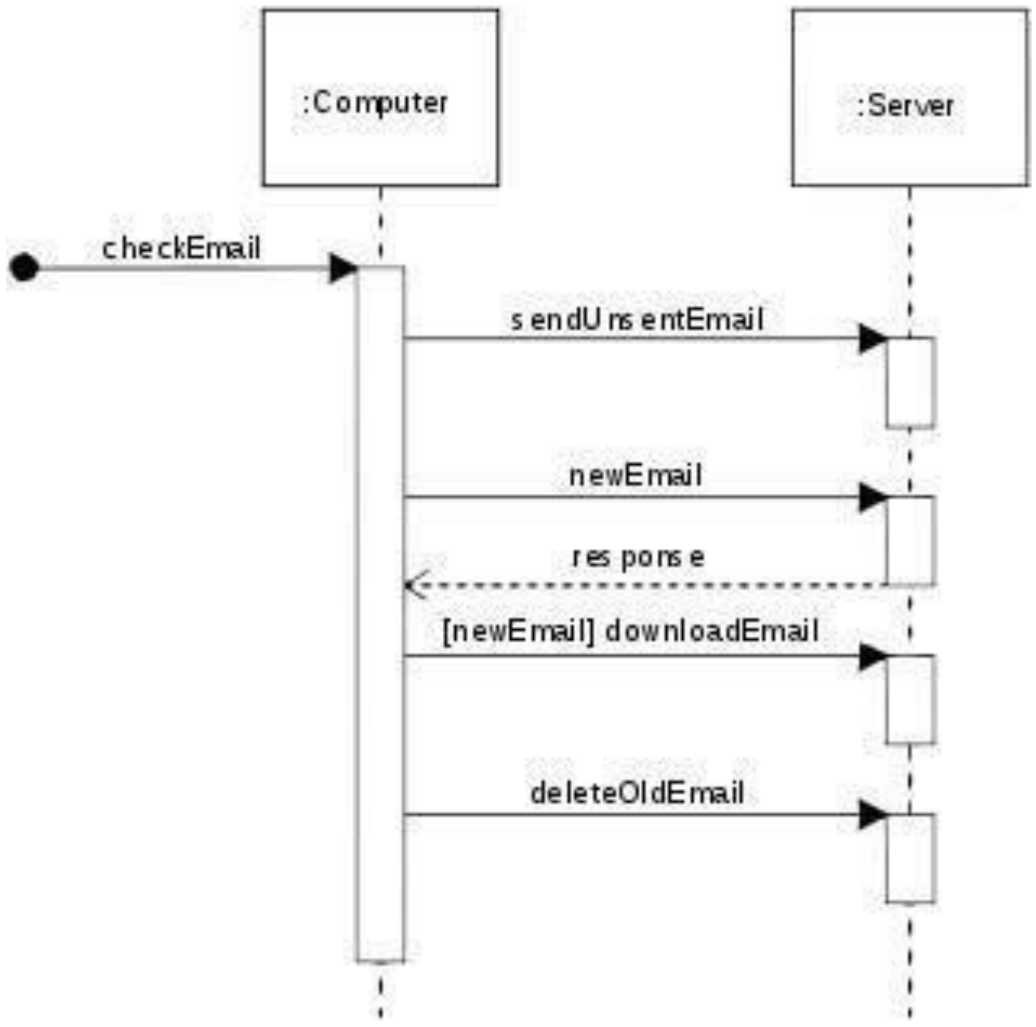

Fig 4.4: Sequence Diagram

4.5 ACTIVITY DIAGRAM

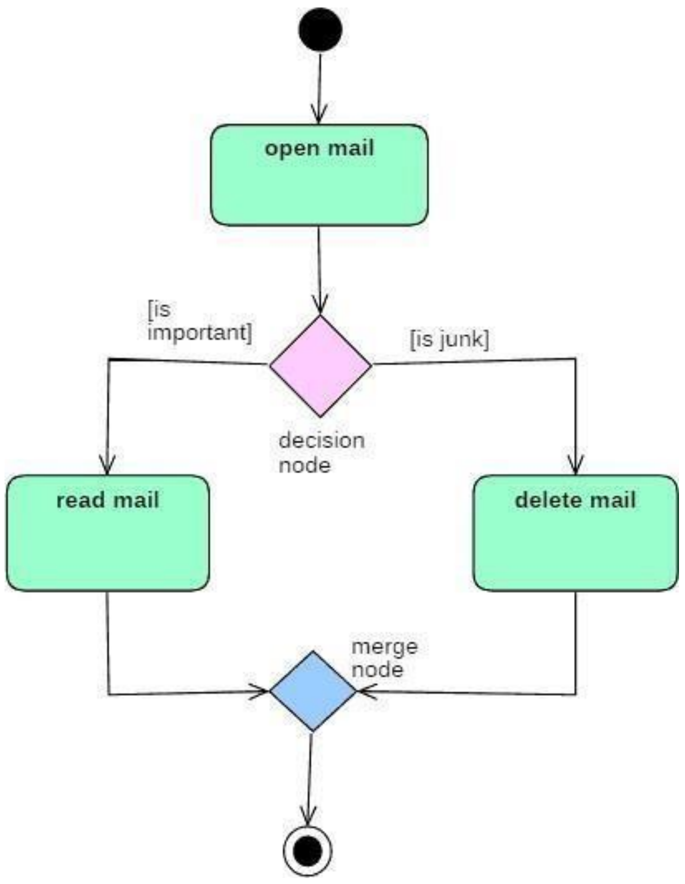

Fig 4.5: Activity Diagram

1.6 IMPLEMENTATION

1.6.1 MODULES

1.6.1.1 MODULE 1

Extracting parameters from URL using python

We have implemented python program to extract features from URL.Below are the features that we have extracted for detection of phishing URLs.

6 Presence of IP address in URL. 7 Presence of

@ symbol in URL.

8 Number of dots in Hostname.

9 URL redirection.

10 Number of slash’s in URL

1.6.1.2 MODULE 2

Creating a model using clustering.

Creating a model using Random forest algorithm. This algorithm works by creating a number of classification trees randomly. These trees are created by making use of different samples from the same dataset and also they may use different types of features each time to create the trees. Thus, all the trees are created randomly by making use of different sub sets of the dataset, and also the features are taken formed, we can do the classification by finding the results of each tree and then assigning it to the class that has been determined by the most number of trees.

1.6.1.3 IMPLEMENTATION STEPS

Right now will talk about the real advances which were actualized while doing the m analyse. We will clarify the stepwise strategy used to break down the information and to foresee the phishing. The framework comprises of the accompanying primary advances, we have utilized unstructured information which comprises just URLs. There are 2905 URLs acquired from Phish tank site which comprises of both phishing and real URL where the vast majority of URLs got are phishing.

We have collected unstructured data of URLs from Phish tank website

In pre-processing, feature generation is done where nine features are generated from unstructured data. These features are length of URL, URL has http, URL has suspicious character, prefix/suffix, number. Of dots, number of slash, URL has phishing term, length of subdomain, URL contains IP address.

After this a structured dataset is created in which each feature contains binary value (0,1) which is then passed to the different classifiers. Next we train the four different classifiers and compare their performance on the basis of accuracy four classifiers used are XG Boost, SVM, Naive Bayes and Stacking, where stacking uses XG Boost and SVM as its base classifier and Random Forest as its meta classifier. Then classifier detects the given URL based on the legitimate it opens that page in browser.

. We compare the accuracy of different classifiers and found XG Boost and Stacking are the best classifiers which gives the maximum accurary.

5.2. PROPOSED ALGORITHM

5.2.1 Random Forest

Random forests are the classifiers that combine many tree possibilities, where each trees are depends on the values of a random vector sampled independently. Then, all trees in the forest will have same allotment. To construct a tree, we assume that n is the number of training observations and p is the number of variables (features) in a training set. To determine the decision node at a tree we choose k « p as the number of variables to be selected. We select a bootstrap sample from the n observations in the training set and use the rest of the observations to estimate the error of the tree in testing phase. Hence, we randomly choose „k" variables as a decision at certain node in the tree and calculate the best split based on the k variables in the training set. Trees are always grown and never pruned compared to other tree algorithms. Random forests can handle large number of variables in a data set. Also, during the forest building process they generate an internal unbiased estimate of the generalization error. Additionally, they can estimate missing data closely.

A major disadvantage of random forests algorithm is it does not gives precise continuous forecast.

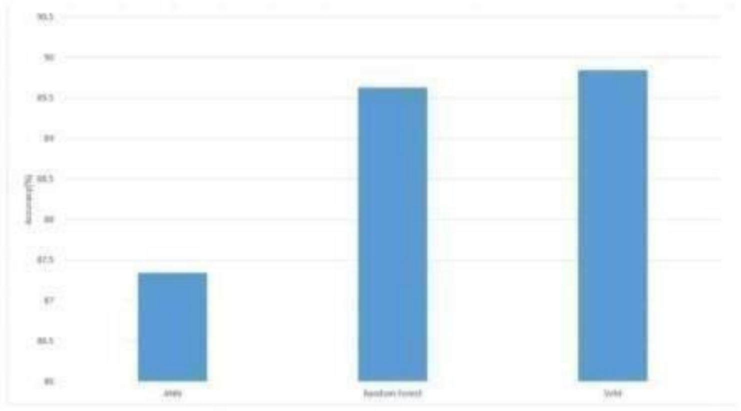

Fig. 5.2.1: Accuracy of classifiers

5.2.2 FEATURE EXTRACTION OF DATASETS:

	Predicted Phishing URLs	Predicted Legitimating URLs
Ground Truth Phishing URLs	1205	250
Ground Truth Legitimate URLs	170	1692

TABLE 5.2.2: Neural Network Confusion Matrix

5.2.2 SAMPLE CODE

```
        from flask import Flask,render_template,request

        import FeatureExtraction import pickle
app = Flask(_name_)

        @app.route('/') def index():   return
        render_template("home.html")


        @app.route('/about') def about():
        return render_template("about.html")


        @app.route('/getURL',methods=['GET','POST'])
        def getURL():  if request.method == 'POST': url
        = request.form['url'] print(url)  data =
        FeatureExtraction.getAttributess(url)  print(data)
            RFmodel = pickle.load(open('RandomForestModel.sav', 'rb')) predicted_value
            = int(RFmodel.predict(data)[0])
            #print(predicted_value) if
            predicted_value == 0: value =
            "Legitimate"  return
            render_template("home.h
            tml",error=value) else:
              value = "Phishing"
              return render_template("home.html",error=value)
```

```
if_name_== "_main_": app.run(debug=True)
```

5.2.2.1 featureextraction.py

```
import pandas as pd  from urllib.parse
import urlparse, urlencode import re from
bs4 import BeautifulSoup import  requests
import  whois import urllib.request from
datetime import datetime import time
import socket  from urllib.error import
HTTPError cd

= None



class FeatureExtraction:  def
    init_(self):   pass


    def getProtocol(self, url):   return
        urlparse(url).scheme


    def getDomain(self, url):  return
        urlparse(url).netloc


    def getPath(self, url):   return
        urlparse(url).path
def havingIP(self, url):
            """If the domain part has IP then it is phishing otherwise legitimate""" match =
            re.search(
           '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-
4]\\d|25[0-5])\\/)|' # IPv4
             '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)'      #      IPv4      in
hexadecimal
             '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url) # Ipv6 if match:
             # print match.group() return
             1 # phishing  else:
             # print 'No matching pattern found' return
             0 # legitimate
def long_url(self, url):
            """This function is defined in order to differntiate website based on the length of the URL""" if len(url)
            < 54:   return 0 # legitimate elif len(url) >= 54 and len(url) <= 75:
```

32 return

```
             2 # suspicious
            else:   return 1 #
             phishing
 def have_at_symbol(self, url):
            """This function is used to check whether the URL contains @ symbol or not""" if
            "@" in url: return 1 # phishing
```

```
                else:        return   0   #
                  legitimate

        def redirection(self, url):
                """If the url has symbol(//) after protocol then such URL is to be classified as phishing """
                if "//" in urlparse(url).path:   return 1 # phishing
                else:        return   0   #
                  legitimate

        def prefix_suffix_separation(self, url):
                """If the domain has '-' symbol then it is considered as phishing site""" if "-" in
                urlparse(url).netloc:
                  return 1 # phishing
                else:        return   0   #
                  legitimate


              def sub_domains(self, url):
                """If the url has more than 3 dots then it is a phishing""" if
                url.count(".") < 3: return 0 # legitimate
                 elif url.count(".") == 3:
                  return 2 # suspicious
                else:   return 1 #
                  phishing


              def shortening_service(self, url):
                """Tiny          URL          ->          phishing          otherwise          legitimate"""                    match          =
                re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
                         'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
                         'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
                         'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
                         'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
                         'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
                         'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip
                         \
            .net',
                  url) if match:
                  return 1 # phishing
                else:            return  0          #

              legitimate        def web_traffic(self, url):

              try:

                  rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(),
                  "xml").find("REACH")['RANK']   except
                TypeError: return
                  1
                except HTTPError:
                  return 2
                rank = int(rank) if (rank
                < 100000):  return 0
                else:
                  return 2

              def domain_registration_length(self, url):
                dns = 0 try:
                  domain_name = whois.whois(urlparse(url).netloc)
                except: dns
        = 1


    if dns == 1:   return 1
        # phishing
    else:
        expiration_date = domain_name.expiration_date today
        = time.strftime('%Y-%m-%d') today =
        datetime.strptime(today,     '%Y-%m-%d')  if
        expiration_date is None:   return 1
        elif type(expiration_date) is list or type(today) is list:  return 2 # If it is a type of list then we can't select a single value
            from list. So,it is regarded as suspected website
        else:
            creation_date   =   domain_name.creation_date   expiration_date   =
            domain_name.expiration_date  if  (isinstance(creation_date,  str)  or
            isinstance(expiration_date, str)): try:
                    creation_date = datetime.strptime(creation_date, '%Y-%m-%d') expiration_date
                    = datetime.strptime(expiration_date, "%Y-%m-%d")   except:
                    return 2
            registration_length = abs((expiration_date - today).days)
            if registration_length / 365 <= 1:   return
                1 # phishing
            else:         return   0   #
                legitimate


def age_domain(self, url):
    dns = 0 try:
        domain_name = whois.whois(urlparse(url).netloc)
    except: dns
        = 1  if dns ==
    1:        return   1
    else:
    creation_date =
    domain_name.c
    reation_date
    expiration_date
    = domain_name.e
    xpiration_date
    if
    (isinstance(crea
    tion_date, str)
    or
    isinstance(expir
    ation_date,
    str)): try:
                creation_date = datetime.strptime(creation_date, '%Y-%m-%d')
                expiration_date = datetime.strptime(expiration_date, "%Y-%m-%d")   except:
                return 2
        if ((expiration_date is None) or (creation_date is None)): return
              1         elif
          ((type(expiration_date) is list)
          or (type(creation_date) is
          list)):
        return
              2         else:
            ageofdomain = abs((expiration_date - creation_date).days)
```

```
                    if ((ageofdomain / 30) < 6):  return 1
                    else:   return 0


            def dns_record(self, url):
                dns = 0 try:
                    domain_name = whois.whois(urlparse(url).netloc)
                # rint(domain_name) except: dns = 1


                if dns == 1: return
                    1  else:   return
                    0

    def statistical_report(self, url): hostname
                = url  h = [(x.start(0), x.end(0)) for x in re.finditer('https://|http://|www.|https://www.|http://www.', hostname)] z =
                int(len(h)) if z !=
                0:
                    y = h[0][1]  hostname = hostname[y:]
                    h = [(x.start(0), x.end(0)) for x in re.finditer('/',
                    hostname)] z = int(len(h)) if z != 0:
                        hostname = hostname[:h[0][0]]
                url_match = re.search(
                    'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es
        |hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly', url)   try:
                    ip_address = socket.gethostbyname(hostname)
                    ip_match = re.search(
                        '146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\
        .217\.116|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.40|
        83\.125\.22\.219|46\.242\.145\.98|107\.151\.148\.44|107\.151\.148\.107|64\.70\.19
        \.203|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\
        .83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|118\.184\.25\.86|67\.208\.74\.71| 23\.253\.12
        6\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43
        \.229\.108\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225
        \.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170\.160\.61|213\.19\.128\.7
        7|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.
        196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|
        198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98\.2
        55\.18|209\.99\.17\.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.4
        6\.211\.158|54\.86\.225\.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.4
        8|110\.34\.231\.42', ip_address)  except:
                return
                    1

    if url_match:
                    return 1
                else:   return
                    0


            def https_token(self, url):  match = re.search('https://|http://',
                url)
                try:   if match.start(0) == 0 and match.start(0) is not None:
                    url = url[match.end(0):] match = re.search('http|https',
                                    url)        if  match:
                        return 1  else: return
                            0  except:
                return
                    1
```

```
def     getAttributess(url):            fe          =
        FeatureExtraction() protocol =
        fe.getProtocol(url)             path       =
        fe.getPath(url)  domain  = fe.getDomain(url) having_ip =
        fe.havingIP(url)  len_url  =
        fe.long_url(url)  having_at_symbol = fe.have_at_symbol(url)
           redirection_symbol = fe.redirection(url)
           prefix_suffix_separation = fe.prefix_suffix_separation(url)
           sub_domains = fe.sub_domains(url) tiny_url
           = fe.shortening_service(url) web_traffic  =
                   fe.web_traffic(url)
           domain_registration_length =
           fe.domain_registration_length(url) dns_record =  fe.dns_record(url)
           statistical_report = fe.statistical_report(url)  age_domain =
           fe.age_domain(url) http_tokens = fe.https_token(url)
d = {
              'Having_@_symbol': pd.Series(having_at_symbol), 'Having_IP': pd.Series(having_ip),
              'Prefix_suffix_separation': pd.Series(prefix_suffix_separation),
              'Redirection_//_symbol': pd.Series(redirection_symbol), 'Sub_domains': pd.Series(sub_domains),
              'URL_Length': pd.Series(len_url), 'age_domain': pd.Series(age_domain),
              'dns_record': pd.Series(dns_record), 'domain_registration_length': pd.Series(domain_registration_length),  'http_tokens':
              pd.Series(http_tokens),
              'statistical_report': pd.Series(statistical_report), 'tiny_url': pd.Series(tiny_url), 'web_traffic':
              pd.Series(web_traffic)
           }
           data = pd.DataFrame(d)
           return data
```

```
           import pandas as pd from
           urllib.parse import
           urlparse,urlencode import
           re from bs4 import
           BeautifulSoup import
           requests import whois
           import
           urllib.request
        from datetime import datetime  import
        time import socket  from urllib.error
        import HTTPError import pandas as
        pd  from urllib.parse import urlparse
        import re   from bs4 import
```

```
          BeautifulSoup import whois import
          urllib.request import time import
          socket from urllib.error import
          HTTPError from datetime import
          datetime


          cd = None class
          FeatureExtraction: def
          init_(self):   pass

   def getProtocol(self,url): return urlparse(url).scheme

   def getDomain(self,url): return urlparse(url).netloc


          def getPath(self,url):  return
          urlparse(url).path          def
          havingIP(self,url):
          """If the domain part has IP then it is phishing otherwise legitimate"""  match=re.search('(([01]?\\d\\d?|2[0-4]\\d|25[0-
          5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0- 5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|' #IPv4
          '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)' #IPv4 in hexadecimal
          '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}',url)       #Ipv6 if match:
          #print match.group()  return 1   #
phishing   else:     #print  'No  matching
pattern found' return 0  # legitimate   def
long_url(self,url):
          """This function is defined in order to differntiate website based on the length of the URL""" if
          len(url) < 54:  return 0  # legitimate elif len(url) >= 54 and len(url) <= 75: return 2   # suspicious
          else:
          return 1   # phishing  def have_at_symbol(self,url):
          """This function is used to check whether the URL contains @ symbol or not""" if "@" in
          url:
          return 1  # phishing else:
            return 0              # legitimate
def redirection(self,url):
          """If the url has symbol(//) after protocol then such URL is to be classified as phishing """ if "//" in
          urlparse(url).path:
          return 1  # phishing else:
            return 0              # legitimate
def prefix_suffix_separation(self,url):
          """If the domain has '-' symbol then it is considered as phishing site""" if "-" in
          urlparse(url).netloc:
          return 1  # phishing else:
            return 0              # legitimate   def
sub_domains(self,url):
          """If the url has more than 3 dots then it is a phishing""" if url.count(".")
          < 3:   return 0  # legitimate elif
          url.count(".") == 3:
          return 2  # suspicious else:
          return 1           # phishing


          def shortening_service(self,url):
          """Tiny URL -> phishing otherwise legitimate"""
          match=re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'
          'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'
          'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'
          'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'
          'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'
          'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'
          'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|tr\.im|link\.zip\.net',url     )     if
          match:
          return 1  # phishing else:
           return 0             # legitimate

   def web_traffic(self,url):  try:
          rank = BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url=" + url).read(),
          "xml").find("REACH")['RANK'] except TypeError:   return              1  except HTTPError:
          return  2  rank=
          int(rank)        if
          (rank<100000):
          return 0  else:
          return
          2                                    def
domain_registration_length(self,url):  dns
= 0 try:
          domain_name = whois.whois(urlparse(url).netloc) except: dns
          = 1


          if dns == 1:  return 1  #phishing else:
          expiration_date = domain_name.expiration_date today = time.strftime('%Y-%m-%d') today =
          datetime.strptime(today, '%Y-%m-%d') if expiration_date is None:  return 1 elif type(expiration_date) is list or
          type(today) is list :  return 2  #If it is a type of list then we can't select a single value from list. So,it is regarded
          as suspected website else:
          creation_date   =   domain_name.creation_date   expiration_date   =
          domain_name.expiration_date  if  (isinstance(creation_date,str)  or
          isinstance(expiration_date,str)):  try:
          creation_date = datetime.strptime(creation_date,'%Y-%m-%d') expiration_date
          = datetime.strptime(expiration_date,"%Y-%m-%d")  except:
return 2 registration_length = abs((expiration_date - today).days) if
registration_length / 365 <= 1:  return 1 #phishing else:  return 0 #
legitimate   def age_domain(self,url):  dns = 0 try:
          domain_name = whois.whois(urlparse(url).netloc) except: dns
          = 1


          if dns == 1:  return
          1 else:
          creation_date        =          domain_name.creation_date expiration_date
           =          domain_name.expiration_date      if
          (isinstance(creation_date,str)      or   isinstance(expiration_date,str)):   try:
          creation_date         =         datetime.strptime(creation_date,'%Y-%m-%d')
          expiration_date      =      datetime.strptime(expiration_date,"%Y-%m-%d")
          except: return 2 if ((expiration_date is None) or (creation_date is None)):
          return 1 elif ((type(expiration_date) is list) or (type(creation_date) is list)):
          return 2 else:
          ageofdomain = abs((expiration_date - creation_date).days)  if
          ((ageofdomain/30) < 6): return 1 else:
          return 0



          def dns_record(self,url):  dns =
          0 try:
```

```
        domain_name = whois.whois(urlparse(url).netloc)
        #rint(domain_name) except: dns = 1


        if dns == 1:  return
        1 else:   return 0




        def statistical_report(self,url):  hostname = url  h = [(x.start(0), x.end(0)) for x in
        re.finditer('https://|http://|www.|https://www.|http://www.', hostname)]  z =
        int(len(h)) if z != 0: y = h[0][1]  hostname
        = hostname[y:]
        h = [(x.start(0), x.end(0)) for x in re.finditer('/', hostname)] z = int(len(h)) if z != 0: hostname = hostname[:h[0][0]]
        url_match=re.search('at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly',   url)
        try:
        ip_address                                    =                                    socket.gethostbyname(hostname)
        ip_match=re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.158|181\.174\.16
        5
        \.13|46\.242\.145\.103|121\.50\.168\.40|83\.125\.22\.219|46\.242\.145\.98|107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.20
        3|199\.184\.144\.27|107\.151\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.58\.192\.225|1
    18\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.22
    9\.108\.32|103\.232\.215\.140|69\.172\.201\.153|216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.
    170\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102|195\.16\.127\.157|34\.196\.13\.28|103\.2
        24\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.19
        7\.72|87\.98\.255\.18|209\.99\.17\.27|216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\.156|54\.8
        2\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',ip_address) except: return
        1


        if url_match:  return
        1 else:   return 0


        def https_token(self,url):
        match=re.search('https://|http://',url) try: if
        match.start(0)==0 and match.start(0) is not None:
        url=url[match.end(0):]
        match=re.search('http|https',url ) if match: return
        1 else:
        return      0          except:  return  1          def
        getAttributess(url): fe = FeatureExtraction() protocol
          =          fe.getProtocol(url)          path      =
        fe.getPath(url) domain = fe.getDomain(url)
        having_ip = fe.havingIP(url) len_url =
        fe.long_url(url)  having_at_symbol            =
        fe.have_at_symbol(url) redirection_symbol  =
        fe.redirection(url)  prefix_suffix_separation =
        fe.prefix_suffix_separation(url)
        sub_domains = fe.sub_domains(url) tiny_url = fe.shortening_service(url)
        web_traffic = fe.web_traffic(url)  domain_registration_length =
        fe.domain_registration_length(url) dns_record  = fe.dns_record(url)
        statistical_report = fe.statistical_report(url)  age_domain       =
        fe.age_domain(url) http_tokens
        = fe.https_token(url)
d={
        'Having_@_symbol':pd.Series(having_at_symbol),'Having_IP':pd.Series(having_ip),
        'Prefix_suffix_separation':pd.Series(prefix_suffix_separation),
        'Redirection_//_symbol':pd.Series(redirection_symbol),
'Sub_domains':pd.Series(sub_domains),'URL_Length':pd.Series(len_url), 'age_domain':pd.Series(age_domain),
'dns_record':pd.Series(dns_record),'domain_registration_length':pd.Series(domain_registration_length),
'http_tokens':pd.Series(http_tokens),
'statistical_report':pd.Series(statistical_report), 'tiny_url':pd.Series(tiny_url), 'web_traffic':pd.Series(web_traffic)
} data=pd.DataFrame(d)
return data
```

5.2.3 RANDOM FOREST.py

	Predicted Phishing URLs	Predicted Legitimating URLs
Ground Truth Phishing URLs	1249	162
Ground Truth Legitimating URLs	182	1680

TABLE 5.2.3: Random Forest Confusion Matrix

```
#coding:utf- #In[1]:
Importpandasaspd importnumpyasnp
###CollectionofData
#In[2]: legitimate_urls=pd.read_csv("legitimate-urls.csv") phishing_urls=pd.read_csv("phishing-urls.csv")
#In[3]: ###DataPreProcessing
#####Data is in two dataframes so we merge them to make one dataframe  #Note:
two dataframes has same columnnames  #In[4]:
Urls=legitimate_urls.append (phishing_urls)  #In[5]: urls.head
(5) #In[6]:
#####Removing Unnecessary columns
#In[7]: urls=urls.drop(urls.columns[[0,3,5]],axis=1)
#In[8]: distributed urls=urls.sample(frac=1).reset_index(drop=True) 20
```

```
#####Removing class variable from the dataset
urls_without_labels=urls.drop('label',axis=1)
urls_without_labels.columns labels=urls['label']
#labels
#####splitting the data into traindata and testdata importrandom


random.seed(100)
fromsklearn.model_selectionimporttrain_test_split
data_train,data_test,labels_train,labels_test=train_test
test_size=0.20,random_state=100)
#####Checking the data is split in equal distribution or not """
train_0_dist=711/1410 print(train_0_dist)
train_1_dist=699/1410 print(train_1_dist)
test_0_dist=306/605 print(test_0_dist) test_1_dist=299/605
print(test_1_dist) """ data_test=data_test.fillna(0)
data_train=data_train.fillna(0) labels_test=labels_test.fillna(0) labels_train=labels_train.fillna(0)
###RandomForest importnumpyasnp fromsklearn.ensembleimportRandomForestClassifier
RFmodel=RandomForestClassifier()
RFmodel.fit(data_train,labels_train)
rf_pred_label=RFmodel.predict(data_test)
#print((list(labels_test)),print(list(rf_pred_label))
fromsklearn.metricsimportconfusion_matrix,accuracy_score
cm2=confusion_matrix(labels_test,rf_pred_label) print(data_train.columns)
print(data_test.columns) fromsklearn.treeimportDecisionTreeClassifier
DTmodel=DecisionTreeClassifier(random_state=0)  DTmodel.fit(data_train,labels_train)
Import pickle
file_name="RandomForestModel.sav"pickle.dump(DTmodel,open(file_name,'wb')
)
```

5.2.4 DECISION TREE.py:

```
# coding: utf-8 # In[1]: import pandas as
pd
# ## Collection of Data # In[2]: legitimate_urls =
pd.read_csv("legitimate-urls.csv") phishing_urls =
```

pd.read_csv("phishing-urls.csv") # In[3]:

legitimate_urls.head(10) phishing_urls.head(10)

Data PreProcessing

Data is in two data frames so we merge them to make one dataframe

# Note: two dataframes has same column names

# In[4]: urls = legitimate_urls.append(phishing_urls)

# In[5]: urls.head(5)

# In[6]: urls.columns # #### Removing Unnecessary column # #### Removing Unnecessary columns

# In[7]:
urls = urls.drop(urls.columns[[0,3,5]],axis=1)

Since we merged two dataframes top 1000 rows will have legitimate urls and bottom 1000 rows will have phishing urls. So if we split the data now and create a model for it will overfit or underfit so we need to shuffle the rows before splitting the data into training set and test set

# In[8]:

# shuffling the rows in the dataset so that when splitting the train and test set are equally distributed urls = urls.sample(frac=1).reset_index(drop=True)

Removing class variable fromthe dataset # In[9]: urls_without_labels =urls.drop('label',axis=1)

urls_without_labels.columns labels =urls['label']#labels # #### splitting the data into train dataand test data # In[49]:

from sklearn.model_selection import train_test_splittdata_train, data_test, labels_train,labels_ test = train_test_split(urls_without_labels, labels, test_size=0.30, random_state=100)# In[37]:

print(len(data_train),len(data_test),len(labels

_tra in),len(l abels_test))

#In[151]:print(labels_train.value_counts()) print(labels_test.value_counts())

####Checking the data is split in equaldistribution or not # In[158]: train_0_dist = 711/1410 print(train_0_dist) train_1_dist = 699/1410 print(train_1_dist) test_0

_dist = 306/605 print(test_0_dist) test_1_dist = 299/605print(test_1_dist)
DecisionTreeClassifier(random_state=0)

creating the model and fitting the data intothe model # In[50]:

From sklearn.tree import DecisionTreeClassifier

DTmodel.fit(data_train,lbpredicting the result for test data # In[51]: pred_label =model.predict(data_te st)#In[52]:

print(pred_label),print(list(labels_test))

creating confusion matrix andcheckingthe accuracy # In[54]:

from sklearn.metrics import confusion_matrix,accuracy_scorecm=confusion

from sklearn.ensemble import RandomForestClassifierRFmodel
RandomForestClassifier()
RFmodel.fit(data_train,labels_train) # In[56]:

rf_pred_label = rfModel.predict(data_test) #

In[57]:
print(list(labels_test)),print(list(

rf_pr ed_label)) # In[58]:
cm2=confusion_matrix(labels_test,rf_pred_l

abel)cm2 # In[60]: accuracy_score(labels_test,rf_pred_label)

###Improving theefficiency # In[138]:

imp_rf_model =

RandomForestClassifier(n_estimators=100,max_depth=30,max_leaf_nodes=1 000

0) # In[140]:

imp_rf_model.fit(data_train,la bels_train) # In[142]: imp_pred_label

= imp_rf_model.predict(data_test) #

In[144]:

cm3 =

confusion_matrix(labels_test,imp_pred _

label)cm3 # In[146]:

5.2.5 App.py:

From flask import Flask,request import FeatureExtractionimport pickleapp = Flask(name) @app.route('/')

def index():

return render_template("home.html") @app.route('/about') defabout():

```
return  render_template("about.html")@app.route('/getURL',methods=['GET','POST'])    def     getURL():if
request.method == 'POST':
url = request.form['url'] print(url)
 data   =
FeatureExtraction.getAttributess(url) print(data)predicted_value
 =RFmodel.predict(data) #print(predicted_value) if predicted_value == 0:
value = "Legitimate"


return render_template("home.html",error=
value) else: value=
"Phishing"   returnrender_template("home.html",  error=value)if name == " main ":
app.run(debug=True)
```

6 RESULTS

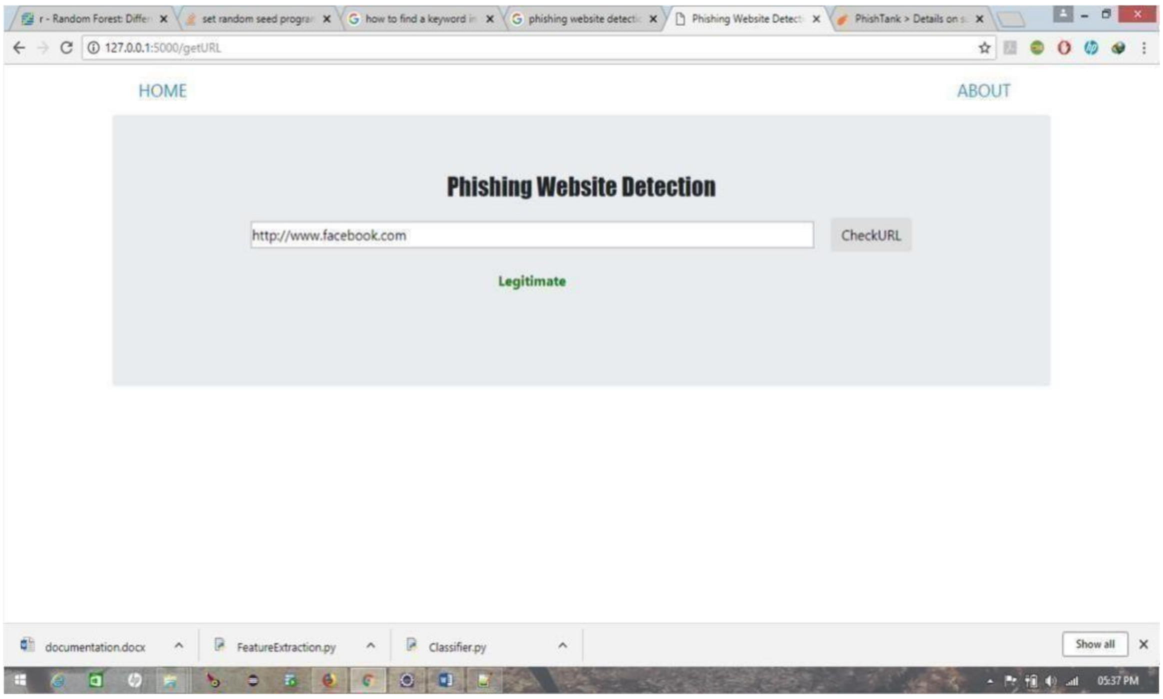

Fig 6.1 Phishing URL

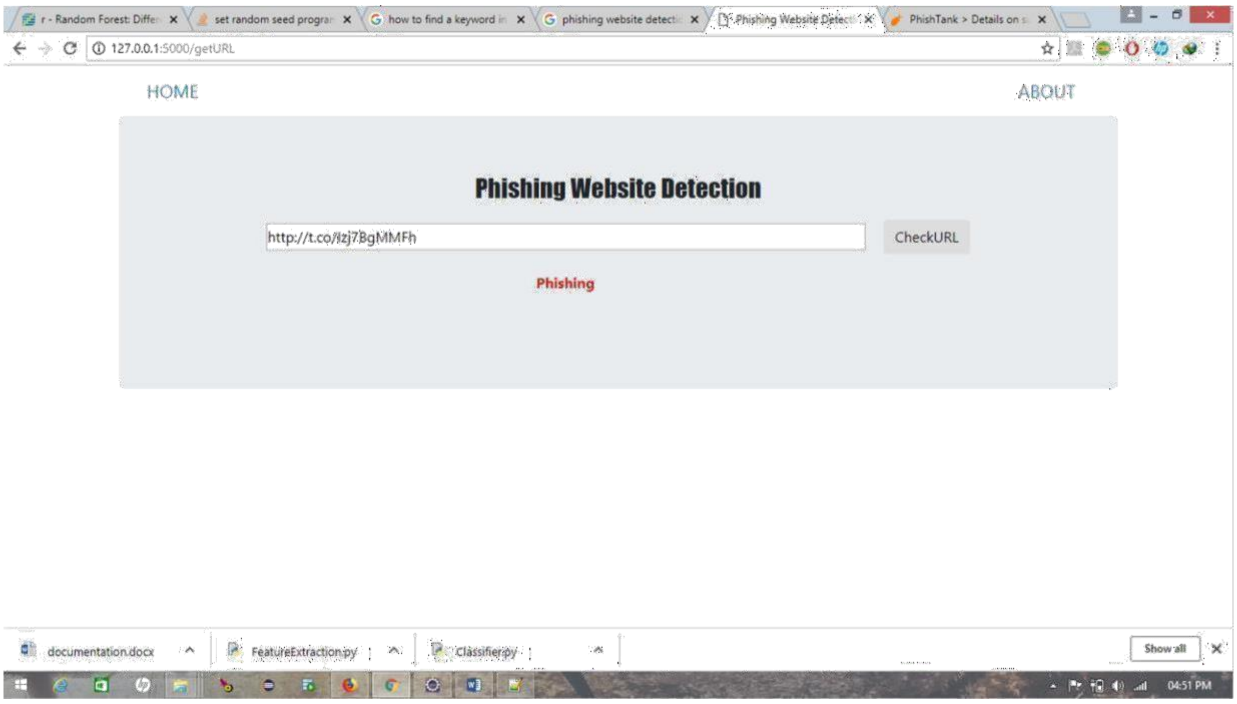

Fig 6.2 Phishing URL

7 CONCLUSION & FUTURE ENHANCEMENTS

This project aims to enhance detection method to detect phishing websites using machine learning technology. We achieved almost 92% detection accuracy using random forest algorithm with lowest false positive rate. Also result shows that classifiers give better performance when we used more data as training data.

In future if we get structured dataset of phishing we can perform phishing detection much more faster than any other technique. In future we can use a combination of any other two or more classifier to get maximum accuracy. We also plan to explore various phishing techniques that uses Lexical features, Network based features, Content based features, Webpage based features and HTML and JavaScript features of web pages which can improve the performance of the system. In particular, we extract features fromURLs and pass it through the various classifiers.

8 REFERENCES

- https://www.researchgate.net/publication/328541785_Phishing_Website

- https://www.researchgate.net/publication/269032183_Detection phishing_URLs_ using_machine_learning_techniques https://www.ijert.org/detection-of-urlbasedphishing-attacks- HYPERLINK "http://www.ijert.org/detection-of-url-based-phishing- attacks-"using-machinelearning https://ieeexplore.ieee.org/document/6731669 https://ieeexplore.ieee.org/document

9 BIBLIOGRAPHY

[1] Object Detection in Underwater Image by Detecting Edges using Adaptive Thresholding, Akshita Saini;Mantosh Biswas-2019

[2] Attentive Layer Separation for Object Classification and Object Localization in Object Detection, Jung Uk Kim;Yong Man Ro-2019

[3] A stereo vision measurement system Based on OpenCV ,Chaohui Lü;Xi Wang;YinghuaShen-2018

[4] OpenCV implementation optimized for a cell broadband engine processor, Hiroki Sugano;Ryusuke Miyamoto-2019

[5] YOLO-compact: An Efficient YOLO Network for Single Category Real-time Object Detection, Yonghui Lu;Langwen Zhang;Wei Xie-2020

[6] Target Classification Using Combined YOLO-SVM in High-Resolution Automotive FMCW Radar, Woosuk Kim;Hyunwoong Cho;Jongseok Kim;Byungkwan Kim;SeongwookLee-2020

[7] Pedestrian Detection Based on YOLO Network Model, Wenbo Lan;Jianwu Dang;Yangping Wang;Song Wang-2018

[8] Application of a Deep Learning Algorithm for Automatic Detection of Unexpected Accidents Under Bad CCTV Monitoring Conditions in Tunnels, Kyu

[9] A framework for abandoned object Detection from video surveillance, Rajesh Kumar Tripathi;Anand Singh Jalal;Charul Bhatnagar-2017

[10] Object Detection Bounding Box-Critic Networks for Occlusion-Robust Object Detection in Road Scene, Jung Uk Kim;Jungsu Kwon;Hak Gu Kim;Haesung Lee;Yong Man Ro-2018