SWE2009 – Data Mining Techniques

Embedded J-Component

WINTER SEMESTER – 2020~2021

E1 +TE1 SLOT

Phishing Website Detection and Classification Using Data Mining

Submitted By

John Arthur Samuel D
18MIS0346
&

Srikar Kotra
18MIS0369

Under guidance of

Prof. Sathiyamoorthy E

M.Tech. Integrated in Software Engineering

Name of the School: SITE

DEPARTMENT OF SOFTWARE ENGINEERING
SCHOOL OF INFORMATION TECHNOLOGY
VELLORE INSTITUTE OF TECHNOLOGY
VELLORE – 632 014
INDIA

Date: 03-06-2021

DECLARATION

I hereby declare that the thesis entitled "Phishing Website Detection and Classification Using Data Mining" submitted by me, for the award of the degree of *Integrated Master of Technology* to VIT is a record of bonafide work carried out by me under the supervision of Prof Sathiyamoorthy E.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore	John Arthur Samuel D	Srikar Kotra
Date: 03/06/2020	**Signature of the Candidate**	**Signature of the Candidate**

CERTIFICATE

This is to certify that the thesis entitled "Phishing Website Detection and Classification Using Data Mining" submitted by **John Arthur Samuel D 18MIS0346** & **Srikar Kotra 18MIS0369** of SITE, VIT, for the award of the degree of *Integrated Master of Technology in Programme,* is a record of bonafide work carried out by him / her under my supervision during the period, 01.02.2021 to 01.06.2021, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 3rd June 2021 **Signature of the Guide**

Internal Examiner *External Examiner*

Head of the Department

(Integrated M.Tech in Software Engineering)

ACKNOWLEDGEMENTS

I cannot express enough thanks to my Data Mining Techniques Faculty Prof Sathiyamoorthy E for his continued support and encouragement in helping me to finish this project. The completion of this project could not have been accomplished without the motivational words from my faculty who encourages us every time to think out of the box to implement simple yet effective solutions to real world problems.

If it weren't for her kind and helpful words, it wouldn't have been possible to finish this project. I'd also extend my thanks to VIT, Vellore and SITE school in particular for parting quality education in form of J-Component projects that helps students to go beyond bookish knowledge to learn industry level skills for the better of the students.

I'd also like to thank youtube channel FreeCodeCamp and for providing a comprehensive introduction to Machine Learning and Deep Learning that helped to learn the basics effectively and implement the learnt basics to a product in form of a capstone project.

John Arthur Samuel D
Student Name

Srikar Kotra
Student Name

Executive Summary

The project Phishing Website Detection and Classification Using ML Algorithms, Neural Networks and Genetic Algorithm uses various ML, DL and Genetic algorithms to determine which model fits best in classifying Phishing Websites from a list of features such as URL age, presence of anchor tags, pop up windows etc.., From this the end user can use this model to determine if a website is legitimate or not. Also, this can be deployed into a web application for generalized usage.

CONTENTS

PHISHING WEBSITE DETECTION AND CLASSIFICATION USING ML ALGORITHMS, NEURAL NETWORKS AND GENETIC ALGORITHMS

1 INTRODUCTION

Cybers attackers and spoofers use various online digital platforms to steal identities of legit businesses and lure in multiple unaware users to bait into revealing their private information or install a malicious software in their devices that range from small scale spywares and adware to highly resilient ransomwares. Many successful anti phishing mechanisms have been proposed and deployed by various anti-virus software companies. The accuracy of these products varies from each other and can sometimes classify and flag non malicious websites as malicious and vice versa.

Phishing is nowadays a very common attack used to obtain sensitive information using visually similar websites to that of legitimate websites. With the growing technology, phishing attacks are on the rise. Data Mining is a very popular approach to detect phishing websites. There are several existing methodologies that help in the classification of the phishing websites from the legitimate websites. Most of the Data Mining algorithm that are used to solvethis problem are discussed in the literature review part of this review.

The purpose of this project is to solve the problem of identification of phishing detection using a various ML, DL and evolutionary algorithm. Using these processes, we can feature select a particular interesting or essential feature that classifies a particular page as a phishing website from the rest of the websites.

2 ABSTRACT

The project involves developing the model with the best fit that solves the problem of classifying the website given its features such as Has_URL. Has_Anchor, URL_age, web traffic, age of domain, etc.., From this given information the model that is to be chosen should be able to classify a website from the user input as a Phishing, Non-Phishing or Suspicious website.

The dataset chosen contains thousands of testcases with the features of 9 different attributes that define the phishing websites. From this using various algorithms such as ML based algorithms, DL based algorithms and Genetic Algorithm, models are to be made. The model with the best fit is to be taken for final consideration.

3 EXISTING MODELS

The existing models mostly emphasize on the usage of Data Mining and Machine Learning classification models for the classification of the phishing websites. This also produces an average accuracy of around 80-90% on an average. They also discuss only 3-4 models in general

4 PROPOSED MODELS

The proposed models are a combination of Data Mining, Neural Networks and Genetic Algorithm in combined to produce 8 distinct approaches and the best of the approach is selected and exported to build a classifier model that can be used to build a client-server web application in the future.

5 PROCESS INVOLVED IN DATA MINING & NEURAL NETWORK ALGORITHMS

1. **Data Preprocessing**

   This is the first step into building every ML model. First data visualization is done and then the data is split into training and test data sets. The typical default is a 70/30 split between training and test sets. Weights, learning rates are also defined in case of neural networks.

2. **Training**

   The data set connects to an algorithm, and the algorithm leverages sophisticated mathematical modeling to learn and develop predictions. This particular problem we deal with classification models

3. **Testing**

   In this step we validate the trained model. We check for the model's accuracy. We change the model until the results are satisfactory.

4. **Performance metrics**

   We do other performance metrics besides testing such as F1 score, Precision, Recall and also plot the confusion matrix. The performances of the various models are analyzed using visualization techniques.

PROCESSES INVOLVED IN GENETIC ALGORITHM

1. **Initialization**

   The method is on account of evolutionary theory. A heuristic random search helps in grouping the basic differences between various elements of the

dataset. For this we can use differential evolution algorithm. DE performs mutation, crossover, and selection operations.

2. Mutation

The DE algorithm implements the mutation operation by the difference method. Random selection of two diverse individuals in a group and scaling vector differences are the common difference strategy. Afterwards, the vector is synthesized with the individual to be mutated. Through mutation, a new intermediary population is now generated

3. Crossover

Crossover involves selection of random features that are distinguished between two vectors and combining them to create a new improved vector that can perform better in the evaluation phase of the process.

4. Evaluation

Evaluation involves calculating the feature score for the various vectors in order to know how well they fare against the selection mechanism and the vectors that perform better than a given threshold can be taken for classification

5. Selection

DE adopts a greedy choice strategy. On the basis of fitness value, better individuals are selected as new ones of new population. Among the given vectors, the vectors with a high level of performance are selected from the bunch using a simple mathematical formula or a function that best helps to describe the region of interest. This helps in classification.

The number of layers may vary which help in classifying a text or a feature through the model. The part of including the evolutionary model in here is in taking only the most interesting feature in each layer and leaving the less interesting ones behind. For this ANN implementation can help in solving the problem.

6 LITERATURE REVIEW

PAPER-1: WC-PAD: Web Crawling based Phishing Attack Detection
AUTHORS: Nathezhtha, Sangeetha, Vaidehi.
DATASET USED:
Datasets are collected from real phishing cases.
ALGORITHMS USED:
• Random Forest
• Naïve Bayes
TOOLS USED:
• Java
• Selenium web driver
SUMMARY:
Here the web crawlers are used for both feature extraction and phishing attack detection. This System shows the overall architecture of the proposed WC-PAD.WC-PAD for detecting phishing attacks has been proposed. WC-PAD performs a three-phase identification mechanism; firstly, the DNS Blacklist based detection is done. Secondly a Web Crawler based detection is accomplished followed by Heuristics based detection. The frequently used phishing IP are easily detected in DNS blacklist testing. The zero-day phishing website attacks are identified in Web crawler and heuristic analysis phase.
ADVANTAGES:
• Heuristic algorithm is used for best results
• Very high accuracy of over 98%
DISADVANTAGES:
• High ratio of false positives which classifies legit pages as phishing sites
METRICS:
Metrics are based the accuracy, sensitivity and specificity. Based on the calculations, classification of phishing website and legitimate website with the random picked website address of 70: 30 ratios of non-phishing and phishing website.
PERFORMANCE MEASURE:
98% accuracy and 1.5 % false positive rate

PAPER-2: A Methodical Overview on Phishing Detection along with an Organized Way to Construct an Anti-Phishing Framework
AUTHORS: Srushti Patil, Sudhir Dhage
DATASET USED:
Authors from have used datasets readily available online like the UCI repository instead of creating their own dataset.
ALGORITHMS USED:
• Random Forest
• SVM ada boost
• Naïve Bayes

TOOLS USED:

• Python

• Java script

SUMMARY:

In this paper, they reviewed various anti-phishing approaches. All methods are discussed to give a clear idea of existing techniques, their limitations and possible improvements. Next, they analyzed the in-use anti-phishing tools available for free. They found that these tools are inefficient to detect all types of phishing sites, especially the newly registered ones. They also mentioned the utmost need to spread awareness regarding the phishing attacks and use of anti-phishing tools while browsing the web.

ADVANTAGES:

• The same dataset consisting of 15 features withsingle algorithm and then using two algorithms. The
accuracy when two algorithms were used was higher

• Very high accuracy of over 98%

DISADVANTAGES:

• The disadvantage of this approach is that a newly created phishing URL may not be present in the
blacklist.

PAPER-3: A predictive model for phishing detection

AUTHORS:

A.A. Orunsolu,A.S. Sodiya,A.T. Akinwale

DATASET USED: The experimental datasets areobtained from data corpus consisting of 40,000 unique phishingURLs fromPhishTank dataset (2018)and 1,000,000,000 legitimateURLs

ALGORITHMS USED:

SVM

NAÏVE BAYES

INCREMENTAL CONSTRUCTION

TOOLS USED:

Python

Feature extractor tool

SUMMARY: The predictive model consists of Feature Selection Module which is used for the construction of an effec-tive feature vector. These features are extracted from the URL, webpage properties and webpage beha-viour using the incremental component-based system to present the resultant feature vector to thepredictive model. The proposed system uses Support Vector Machine and Naïve Bayes which have beentrained on a15-dimensional feature set.

ADVANTAGES:

- High accuracy
- Wide range of dataset

DISADVANTAGES:

There is no exploring the use of design as mobile apps for smartphone-based phishing attacks

METRICS:

- True Positive Rate
- False Positive Rate
- True Negative Rate
- False Negative Rate
- Accuracy.

PERFORMANCE MEASURE:

The experimental resultsindicate a remarkable performance with 0.04% False Positive and 99.96% accuracy for both SVM andNB predictive models.

PAPER-4: An Ensemble Model for Detecting Phishing Attack with Proposed Remove-Replace Feature Selection Technique

AUTHORS: H. S. Hotaa, A. K. Shrivasb, Rahul Hotac

DATASET USED: khonji's anti-phishing website, data set contains 8266 instances, 47 features of type real along with one class with two different class labels: phishing and ham.

ALGORITHMS USED:

Remove-Replace Feature Selection Technique

C4.5 and Classification and Regression Tree (CART)

TOOLS USED:

- Python
- Java script

SUMMARY: This research work emphasizes on constructing an ensemble Machine Learning (ML) based model to detect phishing E-mail with Remove-Replace Feature Selection Technique (RRFST). RRFST reduces features from original feature space by selecting a feature in random manner and removes it if accuracy is being unchanged or increased otherwise feature is being replaced to its original feature space.

ADVANTAGES:

- 99.27% accuracy
- Many algorithms and models are used for best results

DISADVANTAGES: unable to identify phishing E-mail more accurately with minimum number of features.

METRICS: Info Gain (IG) and Gain Ratio (GR).

PERFORMANCE MEASURE: Empirical results indicate that proposed FST produces remarkable performance of 99.27% accuracy using ensemble of C4.5 and CART with only 11 features.

PAPER-5: PhishPreventer: A Secure Authentication Protocol for Prevention ofPhishing Attacks in Mobile Environment with Formal Verification

AUTHORS: Sriramulu Bojjagania,∗, D.R. Denslin Brabinb, P.V. Venkateswara Rao

DATASET USED: Legitimate data fromPhishload (Phishload, 2016) and phishing data from phishtank(PhishTank, 2016) which are publicly available.

ALGORITHMS USED:

- ECDSA

TOOLS USED:

- HTML
- JSP
- Jquery framework
- ECIES

SUMMARY: The current mobile authentication protocols put an extra burden on mobile device users to detect and avoid phishing attacks. In this paper, we propose a novelauthentication protocol that deals with an Authentication Server (AS), which sends a nonce message to the mobile customer deviceto be signed, so that he/she can avoid phishing attacks.

ADVANTAGES

- Applicable for smartphones too
- Customer involvement

DISADVANTAGES:

No potential to improve the securitypractices in real-time mobile app development and security testing.

METRICS: protocols in Scyther is Security Protocol Description Language (SPDL)

PERFORMANCE MEASURE:

It has the potential to both detecting and avoiding phishing attacks.

PAPER-6: PhishSKaPe : A Content based Approach to Escape PhishingAttacks

AUTHORS: Ankit Kumar Jain, Sakshi Parashar, Palak Katare, Isha Sharma

DATASET USED: Alexa dataset and the phishing sites from OpenPhish andPhish Tank

ALGORITHMS USED:

- TF-IDF algorithm
- supportvector machines
- Naıve bayes classifier

TOOLS USED:

- PHP
- JAVASCRIPT
- PYTHON

SUMMARY: The proposed approach involves employing a search engine to match the domain name of the site under scrutiny with the sites thatcome up as result of our search query

ADVANTAGES

improvementby means of difference between the results shown by conventional TF-IDF and the modified proposed approach.

DISADVANTAGES:

These methods are computationally expensiveand require a lot of data.

They are complex and communication intensive.

METRICS:

True positive is when an actual phishing website is detected as a phishing website.

•False positive is when a legitimate website is detected as a phishing website.
•False negative is when an actual phishing website is detected as a phishing website.
•True negative is when a legitimate website is detected as a legitimate website only.

PERFORMANCE MEASURE:

Accuracy is 89.0% for approach as against 88.0% found using conventional TF-IDF approach.

PAPER-7: Using Case-Based Reasoning for Phishing Detection

AUTHORS:

Hassan Y. A. AbutairAbdelfettah Belghith

DATASET USED: Used a small dataset of 572 cases for both legitimate and malicious URLs.

ALGORITHMS USED:

- heuristic search algorithm
- Genetic Algorithms
- K-means Algorithm

TOOLS USED:

CBR shell tool

SUMMARY:

In this study, we propose the CBR-PDS system that can learn and detect phishing attacks. It is also adaptive anddynamic with regards to new cases in contrast to other techniques. CBR-PDS can work effectively with a relativelysmall set of features and data sets as well. Our work is compared with that of[3]. The latter introduces inter-relateddomain similarities to detect phishing websites. However, it stands short to detect new cases of phishing websites forwhich it has not been trained. CBR-PDS can easily adapt and dynamically cover new cases and solve new problemsbased on past experiences.

ADVANTAGES

CBR-PDS system is proposed to predict phishingattacks with a high accuracy. It is scalable to different data set sizes and can adapt proactively.

DISADVANTAGES:

Small data set

METRICS:

- True Positive (TP)
- False Positive (FP)
- True Negative (TN)
- False Negative (FN)
- Precision (P)
- Recall (R)
- Fmeasure
- Accuracy (ACC)

PERFORMANCE MEASURE: A high accuracy with and without Online Phishing Threats(OPT). This accuracy ranges from 95.62% up to 98.07%

PAPER-8: PhiDMA – A phishing detection model with multi-filter approach

AUTHORS:

Gunikhan Sonowal,K.S. Kuppusamy

DATASET USED: Legitimate data fromPhishload (Phishload, 2016) and phishing data from phishtank(PhishTank, 2016) which are publicly available.

ALGORITHMS USED:

- PhiDMA Algorithm
- Transductive Support VectorMachine (TSVM)
- Apriori

TOOLS USED:

Achecker, is one of the leading accessibility evaluation tools.

SUMMARY: A prototype implementation of the proposed PhiDMAmodel is built with an accessible interface so that persons with visual impairments shall access it withoutany barrier. In this paper, we have developed a model which detectsphishing sites with multi-filter approaches and provides hintsregarding the actual site, the user is actually attempting to visit.Each layer in this model acts as a filter to detect phishing using aspecified dimension. Moreover, we have outlined the interface ofthe model as accessible to such an extent that a wide range of indi-viduals, especially persons with visual impairments are capable toutilize this model.

ADVANTAGES

- Skewed dataset
- More metrics used

DISADVANTAGES:

- Conventional old school model
- No significant improvement

METRICS:

- nphish!phish¼TP: Number of phishing URLs correctly classified asphishing.
- nham!ham¼TN: Number of legitimate URLs correctly classifiedas legitimate.
- nham!phish¼FP: Number of legitimate URLs which are classifiedas phish.
- nphish!ham¼FN: Number of phishing URLs which are classifiedas legitimate.
- True Positive (TP)
- False Positive (FP)
- True Negative (TN)
- False Negative (FN)
- Precision (P)
- Recall (R)
- Fmeasure
- Accuracy (ACC):

PERFORMANCE MEASURE:

- accuracy 92.72%
- True positive rate(TPR)638 (95.65%)
- True Negative rate(TNR) 869(87.34%)
- False Positive rate(FPR)126 (12.66%)
- False Negative rate(FNR) 29 (4.35%)

PAPER-9: Phishing detection on tor hidden services
AUTHORS: Martin Steinebach, Sascha Zenglein, Katharina Brandl.Fraunhofer SIT
DATASET USED: Data can be executed as the raw data that is served to the client, forexample by comparison of cryptographic hashes.
ALGORITHMS USED:

- Hash algorithms
- COAV algorithm

TOOLS USED:
any tool that analyses images with crypto-graphic hash values is to subtly manipulate the image can be used
SUMMARY:
This article surveys the literature on the detection of phishing attacks. Phishing attacks target vulnerabilities that exist in systems due to the human factor. Many cyber attacks are spread via mechanisms that exploit weaknesses found in end-users, which makes users the weakest element in the security chain. The phishing problem is broad and no single silver-bullet solution exists to mitigate all the vulnerabilities effectively, thus multiple techniques are often implemented to mitigate specific attacks. This paper aims at surveying many of the recently proposed phishing mitigation techniques. A high-level overview of various categories of phishing mitigation techniques is also presented, such as: detection, offensive defense, correction, and prevention, which we belief is critical to present where the phishing detection techniques fit in the overall mitigation process.

ADVANTAGES
Tor phishing detection.
Site cloning recognition.
Similarity measures.
Automated phishing detection
DISADVANTAGES:
inherently insecure inregards to manually executed phishing
METRICS:
To test the accuracy of the phishing detection metrics, a testing set ofservices with known clones together with randomly chosen individualservices is composed
PERFORMANCE MEASURE:
Results from theimage detection have a high accuracy, since pages that have a similaraddress as well as many shared pages are very likely to actually be phishingclones.

PAPER-10: PhishShield: A Desktop Application to Detect Phishing Webpagesthrough Heuristic Approach
AUTHORS:
Routhu Srinivasa Rao,SyedTaqiAli
DATASET USED: legitimate image database and visual comparison of suspicious website withimage database
ALGORITHMS USED:

- Term Frequency–Inverse Document Frequency(TF-IDF) algorithm
- Genetic algorithm

TOOLS USED:

- NetBeans 8.0.2 IDE
- JAVA compiler,
- JSoup API
- Firebug tool

SUMMARY:

They implemented a desktop application called PhishShield, which concentrates on URL and Website Content ofphishing page. PhishShield takes URL as input and outputs the status of URL as phishing or legitimate website. The heuristicsused to detect phishing are footer links with null value, zero links in body of html, copyright content, title content and websiteidentity. PhishShield is able to detect zero hour phishing attacks which blacklists unable to detect and it is faster than visual basedassessment techniques that are used in detecting phishing.

ADVANTAGES

The main advantage of Application is that it can detect phishing sites which tricks the users by replacing contentwith images, which most of the existing anti-phishing techniques not able to detect, even if they can, they take moreexecution time than our application

DISADVANTAGES:

- Has to improve the response time of the application
- High Computation cost

METRICS:

- True Positive Rate
- False Positive Rate
- True Negative Rate
- False Negative Rate
- Accuracy.

PERFORMANCE MEASURE:

The accuracy rate obtained for PhishShield is 96.57% and covers a widerange of phishing web sites resulting less false negative and false positive rate.

PAPER-11: Systematization of Knowledge (SoK): A Systematic Review of Software-Based Web Phishing Detection

AUTHORS: Zuochao Dou,

DATASET USED:

we study evaluation datasets, detection features, detection techniques, and evaluation metrics.

ALGORITHMS USED:

- Random Forest

- Naïve bayes

TOOLS USED:

- Java
- python

SUMMARY:

Phishing is a form of cyber attack that leverages social engineering approaches and other sophisticated techniques to harvest personal information from users of websites. The average annual growth rate of the number of unique phishing websites detected by the Anti Phishing Working Group is 36.29% for the past six years and 97.36% for the past two years. In the wake of this rise, alleviating phishing attacks has received a growing interest from the cyber security community.

ADVANTAGES:

- Inthis paper, we focus on detective solutions. More specifically,we look at software-based phishing detection schemes that are specialized in identifying and classifying phishing websites.

DISADVANTAGES:

- Accuracy 91% is comparatively low

METRICS:

In this paper, we focus on detective solutions. More specifically, we look at software-based phishing detection schemes that are specialized in identifying and classifying phishing websites.

PERFORMANCE MEASURE:

91% accuracy and 7%% false positive rate

PAPER-12: Favicon – a Clue to Phishing Sites Detection

AUTHORS: Guang-Gang Geng, Xiao-Dong Lee

DATASET USED:

One is evaluation on a dataset with many confusing samples, and another is finding phishing sites actively

ALGORITHMS USED:

- SVM
- Random forest

TOOLS USED:

- Java
- python

SUMMARY:

Phishing is a type of scam designed to steal user's identity. Typically, anti-phishing methods either use blacklists or recognize the phishing pattern with statistical learning. This paper focuses on a tiny but powerful visual element–*favicon*, which is widely used by phishers but ignored by anti-phishing researchers. Indeed, only some lowest-quality phishing campaigns do not use such favicons.

ADVANTAGES:

- In this paper, They analyze the characteristics of favicon in phishing Web sites, and propose a favicon recognition based anti-phishing method, including brand authorization filters to eliminate legitimate brand sites or sites with branding rights.

DISADVANTAGES:

- Cant find email phishing properly

METRICS:

The proposed favicon based anti-phishing method is language independent.

PERFORMANCE MEASURE:

90% accuracy and 9% false positive rate

PAPER-13: A new method for Detection of Phishing Websites: URL

Detection

AUTHORS: Shraddha Parekh

DATASET USED:

The dataset needed for the entire procedure was gathered from Phish tank

ALGORITHMS USED:

- MCAC

TOOLS USED:

- Java
- python

SUMMARY:

Phishing is an unlawful activity wherein people are misled into the wrong sites by using various fraudulent methods. The aim of these phishing websites is to confiscate personal information or other financial details for personal benefits or misuse. As technology advances, the phishing approaches used need to get progressed and there is a dire need for

better security and better mechanisms to prevent as well as detect these phishing approaches.

ADVANTAGES:

- In this paper, a different methodology has been proposed to detect phishing websites by using random forests

DISADVANTAGES:

- Has a flaw that it cannot detect phishing sites which are not listed in the database, including temporarily sites

METRICS:

The proposed favicon based anti-phishing method is language independent.

PERFORMANCE MEASURE:

90% accuracy and 9% false positive rate.

PAPER-14: Phish Box - An approach for phishing validation and detection

AUTHORS: Jen Hao Li, Sheng-de wang

DATASET USED:

Open Phish, Phish Tank a public source for live data from phishing websites.

ALGORITHMS USED:

- Random Forest
- Logistic Regression
- Naïve Bayes

TOOLS USED:

- Python – data visualization, building model, testing and training
- Restful APIs – collection of data and validation of model

SUMMARY:

The paper involves collection of metadata, images and texts from live websites to build a model using various ML classifying algorithms. Then the model that is trained is used to classify test set and the performance measure is calculated for each of the algorithms. The algorithm with the best accuracy and less false positive ratio is taken for deployment and a web page is built that has live data from the classified websites and the input website is checked against this list of classified websites. If found, it returns that the website input on the textbox is a phishing website or not.

ADVANTAGES:

- Classifies both images and text
- Optimally chooses the best algorithm for training the model
- High accuracy of over 95%

DISADVANTAGES:

- High ratio of false positives which classifies legit pages as phishing sites

METRICS:

Metrics are based the validated result from PhishAlexa dataset which has better results than PhishTank, comparing, the results are measured by the ratio number of pages classified vs total data points.

PERFORMANCE MEASURE:

95% accuracy and 3.5 % false positive rate

PAPER-15: Phishing Detection in Websites using Parse Tree Validation

AUTHORS: C. Emilin Shyni, Anesh D Sundar, G.S.Edwin Ebby

DATASET USED: Semantic Link Network's Dataset for Phishing websites

ALGORITHMS USED:

- Parse Trees
- Depth First Search
- Probability

TOOLS USED:

- Python
- Matplotlib
- Numpy
- Pandas

SUMMARY:

From the given dataset a parse tree is made from the phishing websites repository and the functional areas are validated on the basis of the information and from the root node a depth first search algorithm is carried out and until the root node is reached the tree is explored. Based on the root node, the probability of the occurrence of phishing website is calculated relative to the number of existing websites. If the number of occurrences are greater than a said value X, then the cases are considered to be a phishing website. Then the rest of the tree is tested against the entire dataset and the performance measure is calculated.

ADVANTAGES:

- Basic algorithmic approach that is easy to understand and implement
- High percentage of True positives indicating that the algorithm is effective

DISADVANTAGES:

- Pretty basic method to calculate performance measure
- Not enough analysis is done to prove the effectiveness against other existing models and methodologies or is the same dataset taken.

METRICS:

The test matching is done based on the websites that are classified correctly vs the websites that are not and a +1 is given to the true positive score for every correct classification and a ratio of the true positive to that of total testcases is taken as the performance metrics

PERFORMANCE MEASURE: True positive 94% False Positive 4%

PAPER-16: AI Meta-Learners and Extra-Trees Algorithm for the Detection of Phishing Websites

AUTHORS: Yazan Alsariza, Victor Edward Ademo, Abdulateef Alazammo, Amar Karim Bologun

DATASET USED: UCI Kaggle Dataset

ALGORITHMS USED:

- Adaboost
- Bagging
- LegitBoost
- Rotation Forest

TOOLS USED:

- Matlab

SUMMARY:

This method uses ABET algorithm which is the method that combines the AdaBoost.M1 meta-learner and the Extra-tree algorithm. ABET is a boosted Extra-tree, iterated 100 times over a sub-sampled dataset extracted from the original phishing website dataset. In the same vein, RoFET is an ensemble of extra-tree classifiers fitted on a transformed dataset via principal component filter while ensuring high accuracy and the reduction of bias.

BET is a bagged Extra-tree with 150 iterations. In other words, LBET is a mix of the Logit Boost meta-learner and the Extra-Tree algorithm. LBET in simple terms is the best fitted logistic regression of Extra-tree algorithm that handles both noise and outliers inherent in the given dataset.

ADVANTAGES:

- High Accuracy and a proper performance measure is used to calculate the accuracy
- The model is also compared with the existing models along with this proposed model

DISADVANTAGES:

- A model with such high level of accuracy could easily be deployed as a webapp for the usage of public

METRICS:

Metrics in this case are Accuracy, False Positives, False Negatives which are calculated from the total count of true positives, true negatives, false positives and false negatives. Whenever any of these classes are applicable for a given testcase that particular class is given a count+1. And the accuracy is calculated as the ratio of True Positives, True Negatives to that of sum of all the classes.

PERFORMANCE MEASURE: Accuracy: 97.404% False Positives: 0.017% False Negatives: 0.038%, F measure: 0.974%

PAPER-17: Towards the Detection of Phishing Attacks

AUTHORS: Athulya A A, Praveen K

DATASET USED: Blacklisted websites database

ALGORITHMS USED:

- List based approach
- Heuristic based ML approach
- Visual similarity approach
- Search Engine Based approach

TOOLS USED:

- Matlab

SUMMARY:

The paper discusses in details about the various types of phishing that takes place before discussing ways to prevent and algorithmic ways of detecting phishing. The various approaches taken in the paper to detect phishing like image similarity analysis using CNN, text similarity analysis using random forest, SVM, adaboost and the comparative analysis of the effectiveness between them. It also has a heuristic based approach to detect phishing websites. The websites that are classified are then put in a external website for checking and validation through input fields.

ADVANTAGES:

- The paper analyses the various approaches and in essence is a mixture of all the approaches and its effectiveness
- The paper analyses and build a website to put the analysis as an implementation.

DISADVANTAGES:

- The metrics are based on high low classification rather than a percentage to compare between each other

METRICS:

The metrics used in the paper are just used to analyze and check for effectiveness of each of the algorithm used. The algorithms are tabulated against each other and the best one based on accuracy and true positives is taken as the best approach to solve the problem.

PERFORMANCE MEASURE:

- List based approach: High
- Heuristic approach: High
- Search engine-based approach: Medium
- Visual similarity: High
- Crowdsourcing: Low

PAPER-18: Phishing Detection using Random Forest, SVM and Neural Network with Backpropagation

AUTHORS: Smita Sindhu, Sunil Parameshwar Patil, Arya Sreevalsan, Faiz Rahman

DATASET USED: UCI ML Repository for Phishing websites

ALGORITHMS USED:

- SVM
- Random Forest
- Neural Networks
- Backpropagation

TOOLS USED:

- Python
- Scikit learn
- Numpy
- Pandas

SUMMARY:

The paper discusses three methods to detect phishing websites have been implemented. The library sklearn was used to implement Random Forest, SVM and Neural Network with backpropagation classification algorithms. The library sys was used to extract information about constants, functions and methods. The features of URL are extracted using Lexical Feature Extraction and Random Forest classification method, SVM classification method and Neural Network with backpropagation classification algorithm are run to classify the websites as phishing website or legitimate website.

ADVANTAGES:

- 3 different methods of detecting phishing websites through the IP are discussed
- The paper has improved upon the accuracies of previous approaches when compared
- Generally the accuracy of the models used are very high

DISADVANTAGES:

- The approach discussed here could have been implemented as a website to help classify real world websites

METRICS:

The metrics of this paper are based on testing the trained model from the given dataset which is widely used for the purpose. The features of the various algorithms are tested and the best approach is obtained

PERFORMANCE MEASURE:

- Average accuracy: 97.4%
- Average False Positives: 0.18%

PAPER-19: Automatic Detection of Phishing Target from Phishing Webpage

AUTHORS: Gang Liu, Bite Qiu, Liu Wenyin

DATASET USED: Kaggle Phishing dataset

ALGORITHMS USED:

- Vector link relationship
- Ranking relationship
- Test similarity

TOOLS USED:

- MATLAB

SUMMARY:

In this paper, the approach to identifying the potential phishing target of a given webpage is implemented using mines its associated webpage set of the given webpage and systematically inspects those features of the link relation, ranking relation, similarity relation from the given webpage to its associated webpages. A DBSCAN clustering method is employed to find if there exists a cluster of webpages attacked by the given webpage potentially attacking.

ADVANTAGES:

- The paper involves a genuine attempt to capture the phishing websites using DBSCAN clustering technique
- Data analysis and visualization process explain the dataset in a better way for analysis.

DISADVANTAGES:

- Low accuracy and high false positive level

METRICS:

The metrics for this paper are based on EPS and MinPts. In order to calculate the value of accuracy and FP, it is needed to find the value of EPS and MinPts. The values of EPS and MinPts are found on the basis of relative comparison of results from existing papers. The results are then plotted and the values are obtained

PERFORMANCE MEASURE:

- Accuracy 92.4%
- False Positives 4.3%

PAPER-20: Phishing Detection from URLs Using Deep Learning Approach

AUTHORS: Shweta Singh, M.P. Singh, Ramprakash Pandey

DATASET USED: Real life data from Taiwan Anti Phishing group

ALGORITHMS USED:

- Deep learning
- CNN

TOOLS USED:

- Python
- Scikit Learn
- Numpy
- Pandas

SUMMARY:

This paper involves detection of the phishing websites using the image dataset that is associated with phishing websites and use a CNN approach to train a DL model that can recognize the images in phishing websites and can classify the websites as phishing websites. For building the model, firstly the CNN layers are defined with 5 layers, 2 max pooling and 1 ReLu activation function for smoothening. From the smoothened input a neural network of 500, 000 nodes is constructed that help in classifying the input image. The neural network is trained with over 100 epochs and the trained model is used for classification.

ADVANTAGES:

- CNN based approach which gives a higher accuracy and better training because of being neural networks
- Image based classification that helps classifying the given websites based on common image elements which are generally prevalent because of attractiveness

DISADVANTAGES:

- Doesn’t classify some of the websites that do not contain images.
- The project isn’t implemented to deal with real world phishing websites

METRICS:

The metrics are taken form of a confusion matrix which is drawn from 4 parameters ie. Legitimate/ phishing and Positive/Negative. And the leading diagonal is taken for the accuracy and the False positives and True Negatives are taken as inaccuracies in the system.

PERFORMANCE MEASURE:

- Accuracy: 98%
- False positives: 0.2%

PAPER-21: Certain Investigation on Web Application Security: Phishing Detection and Phishing Target Discovery

AUTHORS: R.Aravindhan, Dr.R.Shanmugalakshmi , K.Ramya, Dr. Selvan C

DATASET USED:

Open Phish, Phish Tank a public source for live data from phishing websites.

ALGORITHMS USED:

- Feature extraction
- Adaboost
- Named Entries Extraction

TOOLS USED:

- Python
- Scikit Learn
- Probability

SUMMARY:

This paper involves checking for the phishing websites and general phishing related texts (including mails) using a 3-step algorithmic approach for the improvement in methodology and the phases in detection. The various steps involved in this process are pre-processing of the data, mining for named entries, feature extraction, toxic feature detection, and then from the selected text, Phishing classifier algorithm classifies the phishing data from normal data. Stage 3 involves prediction of percentage probability that a given text is phishing text.

ADVANTAGES:

- The paper involves more robust algorithmic approach of getting the features extracted from the phishing websites using 3 stages of various algorithms.
- Multi-tier phishing detection also enables detection from mails
- Complexity of algorithm is low hence execution time is very fast.
- Robust metrics calculation methodology

DISADVANTAGES:

- Comparatively low accuracy and high number of false positives and true negatives.

METRICS:

Metrics for this project is based on 6 different parameters that need to be satisfied in order to arrive at the average accuracy and false positive measure. These 6 parameters are calculated based on the range of expected values and if the parameters are crossed the given range, it is taken as a valid parameter. Each parameter is assigned a value and the avg of values gives accuracy.

PERFORMANCE MEASURE:

Accuracy: 92% False Positives: 5% False Negatives: 3%

PAPER-22: Detection of Phishing URL using Bayesian Optimized SVM Classifier

AUTHORS: Shrishti Shukla, Pratyush Sharma

DATASET USED: Phistank OPENDNS dataset

ALGORITHMS USED:

- Benign
- Naïve bayes
- SVM
- KNN
- Bayesian optimization

TOOLS USED:

- Python
- Scikit Learn
- Numpy
- Pandas

SUMMARY:

The paper involves checking for phishing websites from the list of 5 different text classification algorithms. The models are built for each of these algorithms and the best model is taken for metrics and performance measure depending on the percentage accuracy of each algorithm. The models are collectively compared based on weighted average of importance and ease of finding from baseline algorithms and when classified correctly, every algorithm is given a +1 score mapped to it.

ADVANTAGES:

- Analysis of the various methods provide a good way of classifying the websites.
- Robust testing methodology and equally good technique for calculating the performance of the classification model.

DISADVANTAGES:

- Comparatively low amount of accuracy compared to the other classification techniques
- Is not deployed to test real world data using webapps or apps

METRICS:

After prediction of the test set from the given dataset, a distribution curve is drawn and the area under the curve is tested against the predicted area. The inverse of the ratio gives the accuracy and the ratio of the accuracy to that of the normalized score at 2% confidence level is taken as precision and the F score is also calculated for the same scenario.

PERFORMANCE MEASURE:

Accuracy: 95.2% F score: 94.8% Precision: 98.5%

PAPER-23: Phishing Detection Using Machine Learning Technique

AUTHORS: Junaid Rashid, Toqeer Mahmood, Muhammad Wasif Nisar, Tahira Nazir

DATASET USED: Web GNU widget for collection of HTML data, images, CSS and JS using a Py script

ALGORITHMS USED:

- FACA
- SVM
- Random Forest

TOOLS USED:

- Python
- Numpy
- Pandas
- Scikit Learn

SUMMARY:

The paper involves classification of the website content as phishing or non-phishing website using various ML algorithm and a comparative study of the same for the accuracy and sensitivity. The accuracy and sensitivity are calculated from the false positives, true negative values and the sensitivity is calculated from he false negative and the true positives values. Based on the sensitivity and accuracy values obtained the methodology is given a performance measure

ADVANTAGES:

- Usage of live data from websites using a web API for fetching as the dataset
- The various methodologies discussed here are standard ways to classify text based phishing websites.

DISADVANTAGES:

- Subpar or low accuracy level for text based phishing detection.

METRICS:

The performance metrics for this paper involve calculation of the accuracy and sensitivity from the false positive, false negative, true positive and true negative value while testing the dataset. Based on the sensitivity and accuracy values obtained the methodology is given a performance measure

PERFORMANCE MEASURE:

- FACA: 90.44%
- SVM: 94.27%
- Random forest: 95.66%

PAPER-24: On Effectiveness of Source Code and SSL Based Features for Phishing Website Detection

AUTHORS: Ayman Asaal, Shakir Baki , Avisha Das, Rakhesh Kahnna

DATASET USED:

Live data from 5000 websites collected using a web API

ALGORITHMS USED:

- Rapid Miner
- Phish Bench

TOOLS USED:

- Python
- Scikit Learn
- Weka

SUMMARY:

The paper discusses three methods to detect phishing websites have been implemented. The library sklearn was used to implement Random Forest, SVM and Neural Network with backpropagation classification algorithms. The library sys was used to extract information about constants, functions and methods. The features of URL are extracted using Lexical Feature Extraction and Random Forest classification method, SVM classification method and Neural Network with backpropagation classification algorithm are run to classify the websites as phishing website or legitimate website.

ADVANTAGES:

- The paper involves new tools to reduce the complexity involved and hence the running time for prediction
- Good accuracy and sensitivity values
- Good performance metrics

DISADVANTAGES:

- The paper is not deployed into a testing web app for the real-world data.

METRICS:

The metrics are calculated based on the similarities from the other dataset and the accuracy is calculated as number of matching entries vs the total area under the curve and the sensitivity values are calculated on the basis of the True positive, True Negative, False positive and False negatives.

PERFORMANCE MEASURE:

Accuracy: 97.8% Sensitivity: 95.5%

PAPER-25: On Effectiveness of Source Code and SSL Based Features for Phishing Website Detection

AUTHORS: Roopak.S, Athira P Vijayaraghavan, Tony Thomas
DATASET USED:

Kaggle Based Dataset for phishing classification

ALGORITHMS USED:

- Ripper Algorithm
- Association rule mining

TOOLS USED:

- Python
- Numpy
- Pandas
- Scikit Learn

SUMMARY:

The paper involves collection dataset that has records for legitimate and illegitimate websites based on the Kaggle dataset that has various entries linking to the phishing websites and their characteristics. Then a interesting feature mining is carried out in form of association rule mining and the websites are classified into N number of classes. From the rule-based association and extraction, the test pages are now labelled from ripper's algorithm. Then based on the labeling, the input dataset is tested for accuracy and sensitivity

ADVANTAGES:

- The paper has improved upon the accuracies of previous approaches when compared
- Data analysis and visualization process explain the dataset in a better way for analysis.

DISADVANTAGES:

- The paper does not build a website to put the analysis as an implementation and help in classifying the real-world data
- Low Accuracy and a proper performance measure is used to calculate the accuracy

METRICS:

The metrics of this paper are based on testing the trained model from the given dataset which is widely used for the purpose. The features of the various algorithms are tested and the best approach is obtained

PERFORMANCE MEASURE:

Accuracy: 92% Sensitivity: 92% F score: 96%

7 LIBRARIES USED

1) Numpy

   NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

2) Pandas

   Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

3) OS

   The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system dependent functionality.

4) Seaborn

   Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

5) Sklearn

   Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

6) Sklearn-genetic

   It features genetic algorithm's feature selection module for scikit learn.

7) 7 Pickle

   Used to save DL models as .sav files

8 CODE IMPLEMENTATION AND RESULTS:

Machine Learning

```
import numpy as np
import pandas as pd
import os

for dirname, _, filenames in os.walk('./'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
./DL part.ipynb
./ML part.ipynb
./MLP_Model.sav
./Multilayer_Perceptron_Model.sav
./Website Phishing.csv
./.ipynb_checkpoints\project-checkpoint.ipynb
./.ipynb_checkpoints\projectDL-checkpoint.ipynb
./.ipynb_checkpoints\projectML-checkpoint.ipynb
./.ipynb_checkpoints\Untitled-checkpoint.ipynb
```

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import math
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
%matplotlib inline

# Preprocessing
data = pd.read_csv("./Website Phishing.csv")
data.head(5)
```

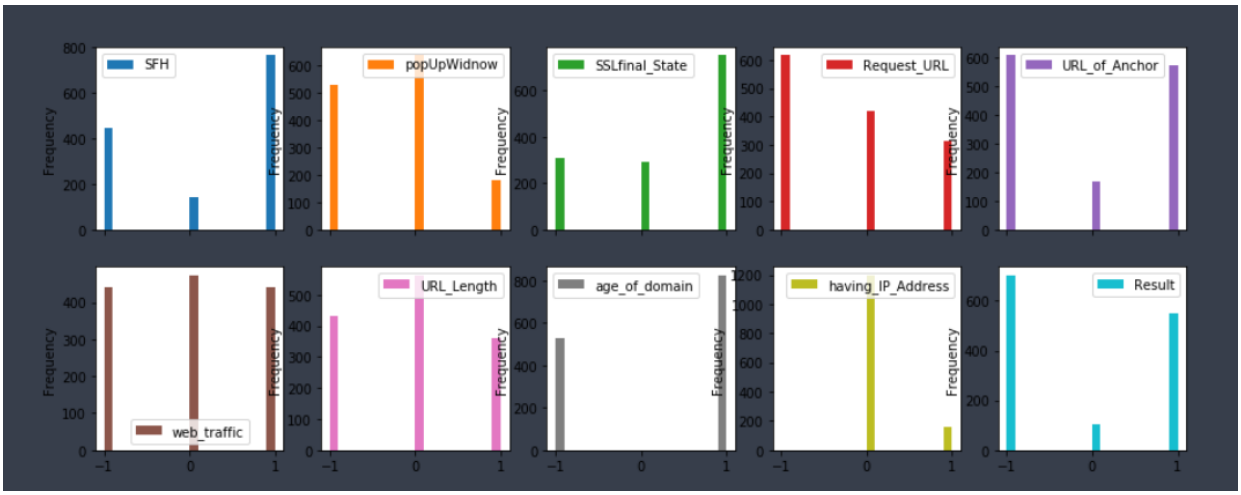
	SFH	popUpWidnow	SSLfinal_State	Request_URL	URL_of_Anchor	web_traffic	URL_Length	age_of_domain	having_IP_A
0	1	-1	1	-1	-1	1	1	1	0
1	-1	-1	-1	-1	-1	0	1	1	1
2	1	-1	0	0	-1	0	-1	1	0
3	1	0	1	-1	-1	0	1	1	0
4	-1	-1	1	-1	0	0	-1	1	0

```
a=len(data[data.Result==0])
b=len(data[data.Result==-1])
c=len(data[data.Result==1])
```

```
print("Count of Legitimate Websites = ", b)
print("Count of Suspicious Websites = ", a)
print("Count of Phishy Websites = ", c)
```

```
Count of Legitimate Websites =  702
Count of Suspicious Websites =  103
Count of Phishy Websites =  548
```

```
data.plot.hist(subplots=True, layout=(5,5), figsize=(15, 15), bins=20)
```

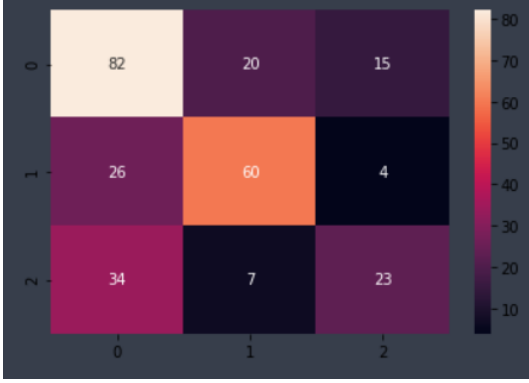



```
x=data.drop("Request_URL", axis=1)
y=data["Request_URL"]
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=60)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 6)
knn.fit(x_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='uniform')
```

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x26fe4d56648>
```




```
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
predic = knn.predict(x_test)
print("F1 score:",f1_score(y_test, predic,average='weighted'))
print("Accuracy:", accuracy_score(y_test, predic) * 100, "%")
```

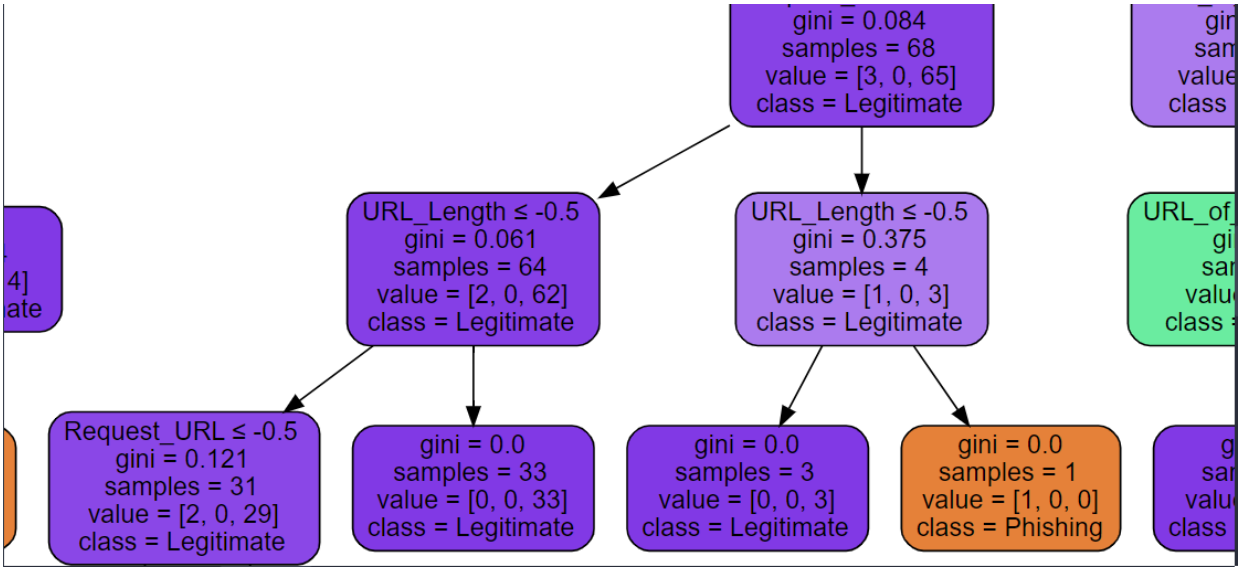
```
F1 score: 0.6010165171333759
Accuracy: 60.88560885608856 %
```

```
x = data.drop('Result',axis=1).values
y = data['Result'].values

from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=10)

print("Training set has {} samples.".format(x_train.shape[0]))
print("Testing set has {} samples.".format(x_test.shape[0]))
```

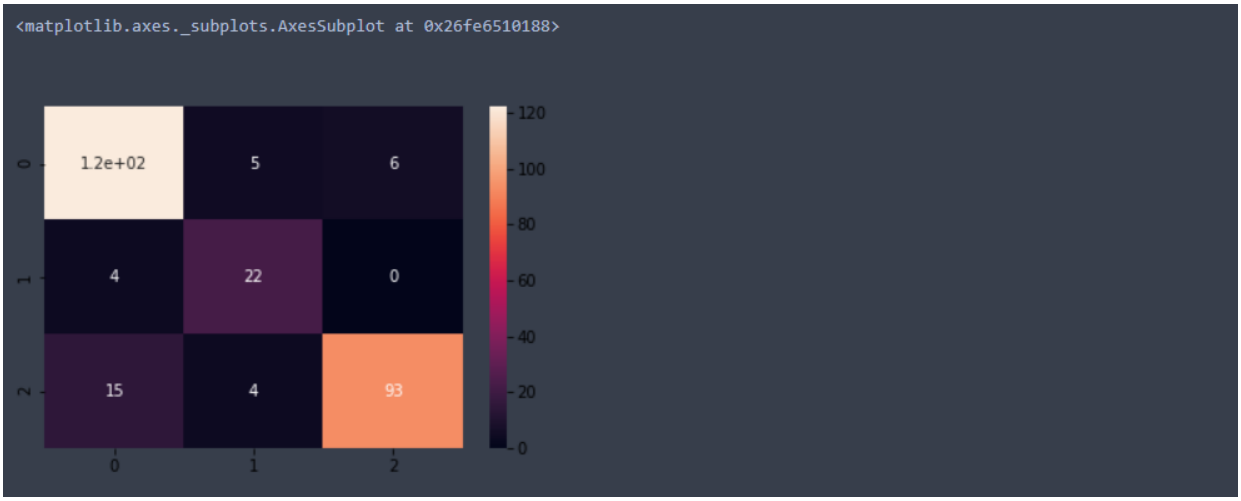
```
Training set has 1082 samples.
Testing set has 271 samples.
```

```
from sklearn import tree
model_tree = tree.DecisionTreeClassifier()
model = model_tree.fit(x_train, y_train)
features = ('SFH', 'popUpWidnow', 'SSLfinal_State', 'Request_URL', 'URL_of_Anchor',
        'web_traffic',   'URL_Length','age_of_domain',        'having_IP_Address')
name = ('Phishing', 'Suspicious', 'Legitimate')

import graphviz
dot_data = tree.export_graphviz(model, out_file=None,
                    feature_names = features, class_names =  name,
```

```
                filled=True, rounded=True,
                special_characters=True)
graph = graphviz.Source(dot_data)
graph
```




```
from sklearn.metrics import matthews_corrcoef
predictions = model.predict(x_test)
from sklearn.metrics import confusion_matrix,classification_report
c=confusion_matrix(y_test,predictions)

sns.heatmap(c, annot=True)
```




```
print("F1 score",f1_score(y_test,predictions,average='weighted'))
print("Accuracy:",100 *accuracy_score(y_test,predictions))
```

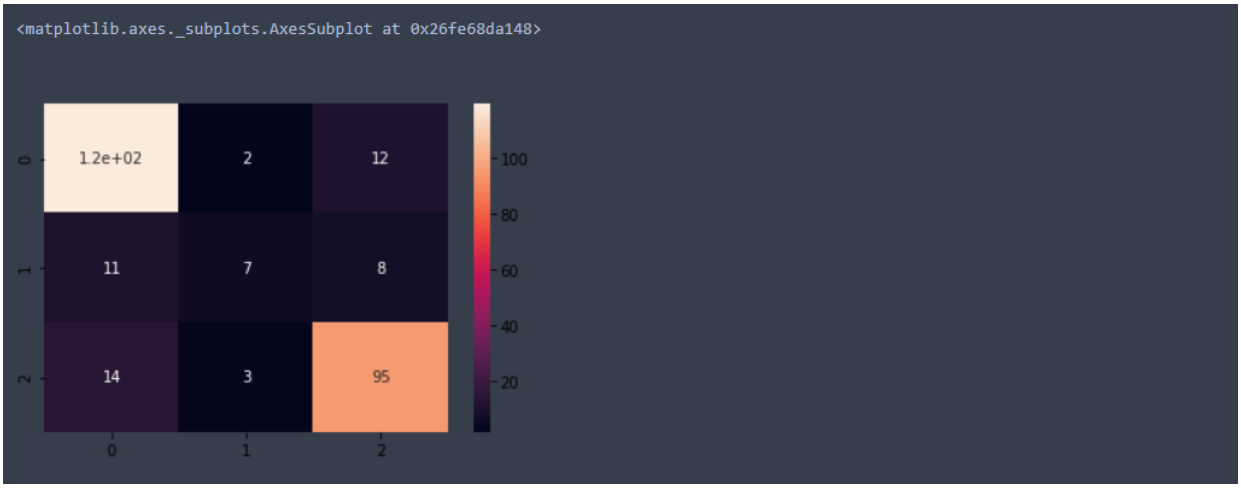
```
F1 score 0.8754169352873847
Accuracy: 87.4538745387454
```

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
y_pred = gnb.fit(x_train, y_train).predict(x_test)
print("Number of mislabeled points out of a total",x_test.shape[0],"points:", (y_test !=
    y_pred).sum())
```

```
Number of mislabeled points out of a total 271 points: 50
```

```
from sklearn.metrics import matthews_corrcoef
predictions = model.predict(x_test)
from sklearn.metrics import confusion_matrix,classification_report
c=confusion_matrix(y_test,y_pred)

sns.heatmap(c, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x26fe68da148>
```




```
print("F1 score",f1_score(y_test,y_pred,average='weighted'))
print("Accuracy:",100 *accuracy_score(y_test,y_pred))
```

```
F1 score 0.8029439446182232
Accuracy: 81.54981549815497
```

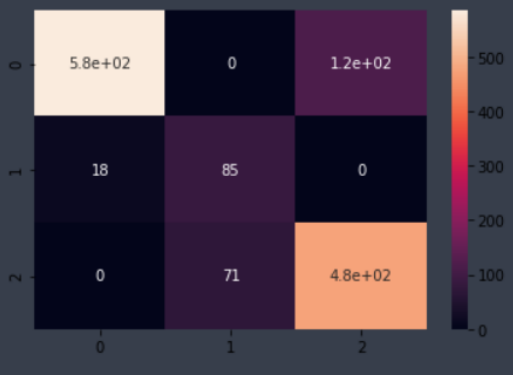
Deep learning

```
def calculate_precision(arr):
    precision=[]
    for i in range(len(arr)):
        sumVal=0
        for j in range(len(arr)):
```

```
            sumVal+=arr[i][j]
        precision.append(arr[i][i]/sumVal)
    return precision

def calculate_accuracy(arr):
    leading_diagonal=0
    for i in range(0, len(arr)):
        leading_diagonal+=arr[i][i]
    return leading_diagonal/sum(sum(np.array(arr)))

def calculate_recall(arr):
    recall=[]
    for i in range(len(arr)):
        sumVal=0
        for j in range(len(arr)):
            sumVal+=arr[j][i]
        recall.append(arr[i][i]/sumVal)
    return recall

import numpy as np

class Perceptron:
    def __init__(self, learning_rate=0.05, n_iters=1000):
        self.lr = learning_rate
        self.n_iters = n_iters
        self.activation_func = self._unit_step_func
        self.weights = None
        self.bias = None

    def fit(self, X, y):
        n_samples, n_features = X.shape
        self.weights = np.zeros(n_features)
        self.bias = 0
        y_ = np.array([1 if i > 0 else 0 for i in y])
        for _ in range(self.n_iters):
            print(_)
            for idx, x_i in enumerate(X):
                linear_output = np.dot(x_i, self.weights) + self.bias
                y_predicted = self.activation_func(linear_output)
                update = self.lr * (y_[idx] - y_predicted)
                self.weights += update * x_i
                self.bias += update
```

```
            print(idx, x_i)
 #
    def predict(self, X):
        linear_output = np.dot(X, self.weights) + self.bias
        y_predicted = self.activation_func(linear_output)
        return y_predicted

    def _unit_step_func(self, x):
        return np.where(x>=0, 1, 0)

 # Training the model
 data = pd.read_csv("./Website Phishing.csv")
 data.head(10)
 x=data.values.tolist()
 training_data=[i[:-1] for i in x]
 label_value=[i[-1] for i in x]
 perceptron_model=Perceptron()
perceptron_model.fit(np.array(training_data), label_value)

# Prediction
import seaborn as sns
from sklearn.metrics import confusion_matrix
perceptron_predictions=[]
for i in range(len(training_data)):
   res=perceptron_model.predict(training_data[i])
   perceptron_predictions.append(res)

c=confusion_matrix(label_value, perceptron_predictions)
sns.heatmap(c, annot=True)

perceptron_accuracy=calculate_accuracy(c)
perceptron_precision=calculate_precision(c)
perceptron_recall=calculate_recall(c)

print("accuracy:", perceptron_accuracy)
print("precision: ", perceptron_precision)
print("recall: ", perceptron_recall)
```

```
accuracy: 0.8477457501847746
precision:  [0.8333333333333334, 0.8252427184466019, 0.8704379562043796]
recall:  [0.9701492537313433, 0.5448717948717948, 0.803030303030303]
```



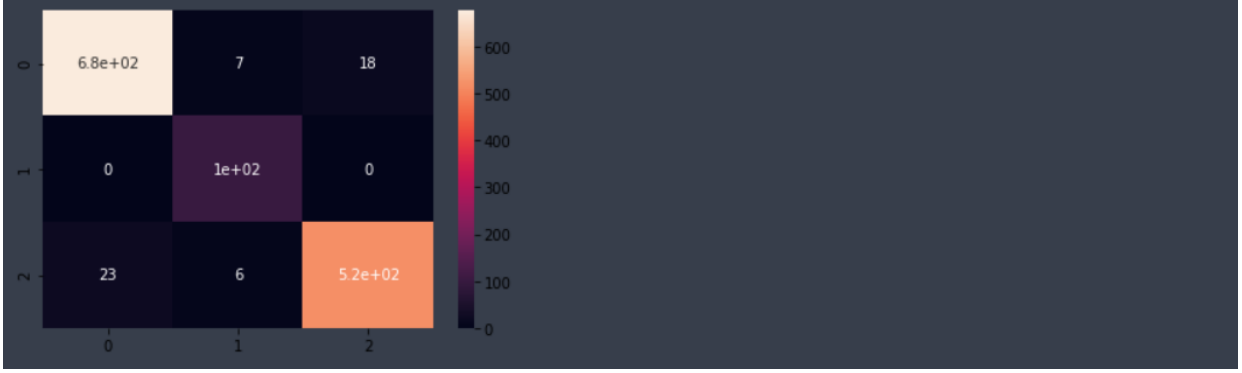

```
# multilayer perceptron with backpropagation
from sklearn.neural_network import MLPClassifier
X = [i[:-1] for i in x]
y = [i[-1] for i in x]
clf = MLPClassifier(solver='lbfgs', alpha=1e-6, hidden_layer_sizes=(15,7), random_state=1,
    max_iter=4000)
clf.fit(X, y)
```

```
MLPClassifier(activation='relu', alpha=1e-06, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(15, 7), learning_rate='constant',
              learning_rate_init=0.001, max_fun=15000, max_iter=4000,
              momentum=0.9, n_iter_no_change=10, nesterovs_momentum=True,
              power_t=0.5, random_state=1, shuffle=True, solver='lbfgs',
              tol=0.0001, validation_fraction=0.1, verbose=False,
              warm_start=False)
```

```
# multilayer perceptron prediction
X=[i[:-1] for i in x]
multilayer_perceptron_results=[]
for i in range(len(training_data)):
   res=clf.predict([X[i]])
   multilayer_perceptron_results.append(res[0])

c=confusion_matrix(label_value, multilayer_perceptron_results)
sns.heatmap(c, annot=True)

multilayer_perceptron_accuracy=calculate_accuracy(c)
multilayer_perceptron_precision=calculate_precision(c)
multilayer_perceptron_recall=calculate_recall(c)

print("accuracy:", multilayer_perceptron_accuracy)
print("precision: ", multilayer_perceptron_precision)
```

```
print("recall: ", multilayer_perceptron_recall)
```

```
accuracy: 0.9600886917960089
precision:  [0.9643874643874644, 1.0, 0.9470802919708029]
recall:  [0.9671428571428572, 0.8879310344827587, 0.9664804469273743]
```

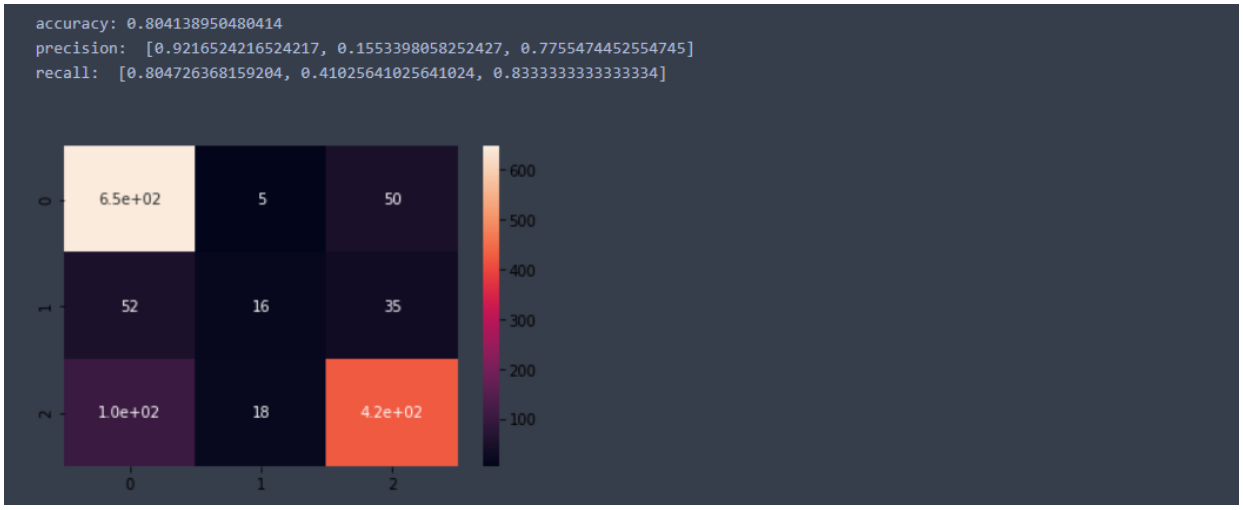



```
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

clf = make_pipeline(StandardScaler(),SGDClassifier(max_iter=10000, tol=1e-5))
clf.fit(X, y)
```

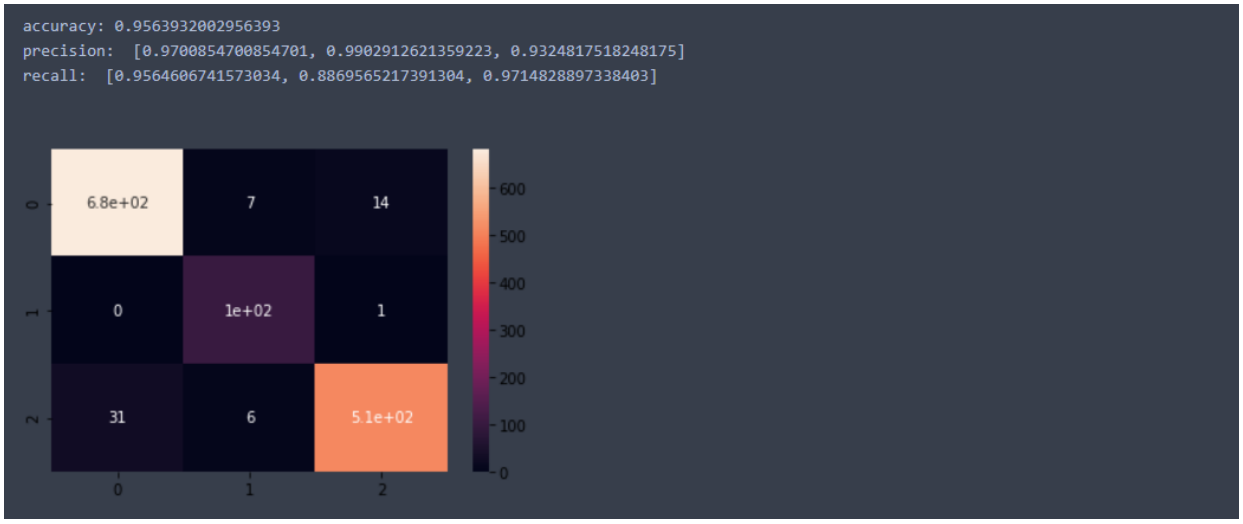
```
Pipeline(memory=None,
         steps=[('standardscaler',
                 StandardScaler(copy=True, with_mean=True, with_std=True)),
                ('sgdclassifier',
                 SGDClassifier(alpha=0.0001, average=False, class_weight=None,
                               early_stopping=False, epsilon=0.1, eta0=0.0,
                               fit_intercept=True, l1_ratio=0.15,
                               learning_rate='optimal', loss='hinge',
                               max_iter=10000, n_iter_no_change=5, n_jobs=None,
                               penalty='l2', power_t=0.5, random_state=None,
                               shuffle=True, tol=1e-05, validation_fraction=0.1,
                               verbose=0, warm_start=False))],
         verbose=False)
```

```
adaline_results=[]
X=[i[:-1] for i in x]
for i in range(len(training_data)):
    res=clf.predict([X[i]])
    adaline_results.append(res[0])

c=confusion_matrix(label_value, adaline_results)
sns.heatmap(c, annot=True)

adaline_accuracy=calculate_accuracy(c)
adaline_precison=calculate_precision(c)
adaline_recall=calculate_recall(c)
```

```
print("accuracy:", adaline_accuracy)
print("precision: ", adaline_precison)
print("recall: ", adaline_recall)
```




```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
clf = MLPClassifier(random_state=1, max_iter=1000).fit(training_data, label_value)

mlp_results=[]
X=[i[:-1] for i in x]
for i in range(len(training_data)):
    res=clf.predict([X[i]])
    mlp_results.append(res[0])

c=confusion_matrix(label_value, mlp_results)
sns.heatmap(c, annot=True)

mlp_accuracy=calculate_accuracy(c)
mlp_precision=calculate_precision(c)
mlp_recall=calculate_recall(c)

print("accuracy:", mlp_accuracy)
print("precision: ", mlp_precision)
print("recall: ", mlp_recall)
```

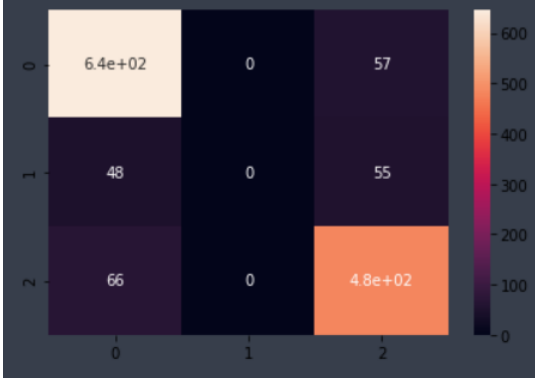



```
!pip install sklearn-genetic

from __future__ import print_function
import numpy as np
from sklearn import datasets, linear_model
from genetic_selection import GeneticSelectionCV

    estimator = linear_model.LogisticRegression(solver="liblinear", multi_class="ovr")
    selector = GeneticSelectionCV(estimator,
                                  cv=5,
                                  verbose=1,
                                  scoring="accuracy",
                                  max_features=5,
                                  n_population=50,
                                  crossover_proba=0.5,
                                  mutation_proba=0.2,
                                  n_generations=40,
                                  crossover_independent_proba=0.5,
                                  mutation_independent_proba=0.05,
                                  tournament_size=3,
                                  n_gen_no_change=10,
                                  caching=True,
                                  n_jobs=-1)
selector = selector.fit(training_data, label_value)
```

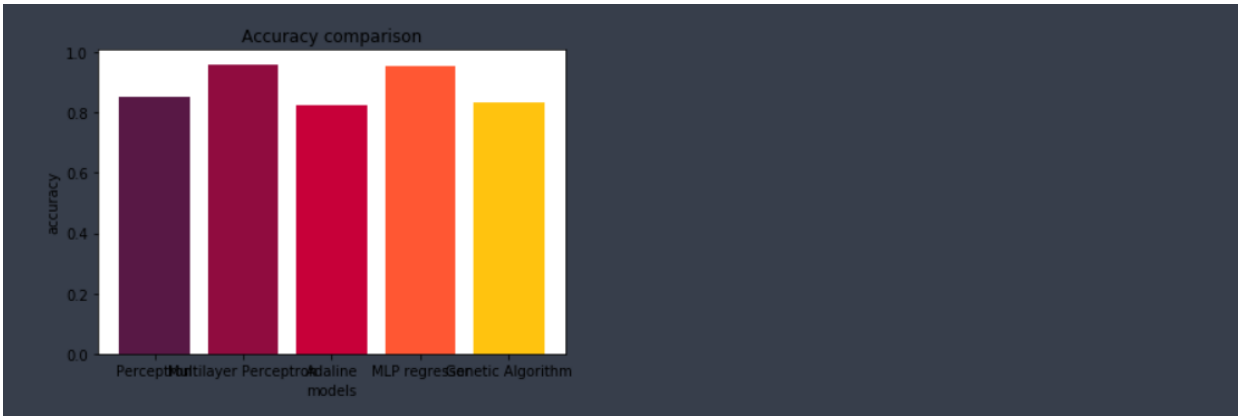
```
Selecting features with genetic algorithm.
gen     nevals  avg                     std                     min                     max
0       50      [0.71410327 2.8       ] [0.08486699 1.32664992] [0.51884652 1.        ] [0.81893126 5.        ]
1       28      [0.77912571 3.22      ] [0.04342309 1.25363472] [0.61121225 1.        ] [0.81893126 5.        ]
2       23      [-199.21147525    2.8       ]   [1.40011265e+03 1.32664992e+00] [-1.e+04  1.e+00]       [0.81893126 6.        ]
3       32      [0.81191007 2.1       ]         [0.01421853 0.83066239]         [0.75241219 1.        ] [0.81893126 4.        ]
4       30      [0.81776351 1.96      ]         [0.00558703 0.39799497]         [0.78048927 1.        ] [0.81893126 3.        ]
5       34      [0.81856187 2.02      ]         [0.0025857 0.14     ]           [0.80046194 2.        ] [0.81893126 3.        ]
6       31      [0.81198415 2.06      ]         [0.03887611 0.46518813]         [0.54472051 1.        ] [0.82856089 5.        ]
7       29      [0.81634533 2.12      ]         [0.01166579 0.51536395]         [0.75241219 2.        ] [0.82856089 5.        ]
8       31      [0.81671663 2.42      ]         [0.01237959 0.9610411 ]         [0.75241219 2.        ] [0.82856089 5.        ]
9       21      [0.81578806 2.88      ]         [0.01481564 1.2749902 ]         [0.75241219 1.        ] [0.82856089 5.        ]
10      31      [-199.20397048    3.54      ]   [1.40011372e+03 1.34476764e+00] [-1.e+04  2.e+00]       [0.82856089 6.        ]
11      30      [0.81823644 4.16      ]         [0.04044943 1.30169121]         [0.54472051 1.        ] [0.83594369 5.        ]
12      24      [-199.1876123    4.88     ]     [1.40011606e+03 5.15363949e-01] [-1.e+04  2.e+00]       [0.83594369 6.        ]
13      24      [0.82984529 4.78      ]         [0.00287408 0.46      ]         [0.82633593 3.        ] [0.83594369 5.        ]
14      31      [-199.18533798    4.64      ]   [1.40011638e+03 5.20000000e-01] [-1.e+04  4.e+00]       [0.83594369 6.        ]
15      42      [-399.19954741    4.24      ]   [1.95975519e+03 6.18384993e-01] [-1.e+04  3.e+00]       [0.83594369 7.        ]
16      29      [0.8347315 4.       ]           [0.00778029 0.2       ]         [0.78050567 3.        ] [0.83594369 5.        ]
17      23      [0.83547066 4.02      ]         [0.00291926 0.14      ]         [0.81524122 4.        ] [0.83594369 5.        ]
18      28      [0.83445133 4.02      ]         [0.00758566 0.31559468]         [0.78272789 3.        ] [0.83594369 5.        ]
19      31      [0.83468722 3.96      ]         [0.0075436  0.19595918]         [0.78272789 3.        ] [0.83594369 4.        ]
20      37      [0.83443614 4.        ]         [0.01013998 0.2       ]         [0.7635151 3.       ]   [0.83594369 5.        ]
21      30      [0.83434757 4.02      ]         [0.01014902 0.24413111]         [0.7635151 3.       ]   [0.83594369 5.        ]
```

```
genetic_algorithm=[]
X=[i[:-1] for i in x]
for i in range(len(training_data)):
    res=selector.predict([X[i]])
    genetic_algorithm.append(res[0])

c=confusion_matrix(label_value, genetic_algorithm)
sns.heatmap(c, annot=True)

genetic_algorithm_accuracy=calculate_accuracy(c)
genetic_algorithm_precision=calculate_precision(c)
genetic_algorithm_recall=calculate_recall(c)

print("accuracy:", genetic_algorithm_accuracy)
print("precision: ", genetic_algorithm_accuracy)
print("recall: ", genetic_algorithm_accuracy)
```

```
accuracy: 0.8329637841832964
precision:  0.8329637841832964
recall:  0.8329637841832964


C:\Users\JohnConda\lib\site-packages\ipykernel_launcher.py:13: RuntimeWarning: invalid value encountered in longlong_scalars
  del sys.path[0]
```




```
# accuracy histogram
import matplotlib.pyplot as plt
xAxis = [perceptron_accuracy, multilayer_perceptron_accuracy, adaline_accuracy,
    mlp_accuracy, genetic_algorithm_accuracy]
yAxis= [1,2,3,4,5]
left = [1, 2, 3, 4, 5]
height = xAxis
tick_label = ['Perceptron', 'Multilayer Perceptron', 'Adaline', 'MLP regressor', 'Genetic
    Algorithm']
plt.bar(left, height, tick_label = tick_label, width = 0.8, color = ['#581845', '#900c3f', '#c70039',
    '#ff5733', '#ffc30f' ])
plt.xlabel('models')
plt.ylabel('accuracy')
plt.title('Accuracy comparison')
plt.show()
```



9 PERFORMANCE ANALYSIS OF THE MODELS

SNO	CLASSIFIER MODEL	ACCURACY	PRECISION (AVG OF CLASSES)	RECALL (AVG OF CLASSES)
1	KNN	0.6885	0.60101	0.6785
2	Decision Tree	0.8753	0.87541	0.8943
3	Naïve Bayes	0.8154	0.8029	0.8098
4	Single Layer Perceptron	0.8523	0.8430	0.7748
5	Multi-Layer Perceptron	0.9600	0.9704	0.9405
6	ADALINE	0.8240	0.6129	0.6304
7	MLP Regressor	0.9563	0.9642	0.9383
8	Genetic algorithm	0.8329	0.8418	0.8296

10 FUTURE WORK:

Using the saved model, a new web application can be created that can classify the webpages based on the user input of the various fields like pop up window, request URL, age of domain, having IP address, etc.., The user fills a form with these factors and the server would respond to the clients request with the result from the trained model, in this case both multilayer perceptron and MLP and then send the result. Based on the results, the user can choose to either trust the website or to avoid it.

11 CONCLUSIONS:

Nowadays, information technology is entering the era of distributed systems, cloud computing and open-source networks. With the advent of world wide web people is facing more and more security problems in distinguishing legitimate sites from illegal sites. Security is a key issue in web browsing, because it seriously affects the various security protocols taken to prevent the

issue in the first place. Intrusion detection is an important way to ensure internet phishing prevention. Thus, by finding the right dataset and by using feature extraction on the model, we can build an ML, DL and evolutionary algorithm that best suit the cause of solving the issue.

Then we can have a comparative study of how much accuracy that we had from classifying these models. Now that we have obtained that Multilayer Percptron and MLP Regressor have accuracies >96, we can use them. From them, we can build and deploy an online model that helps users to identify legitimate sites from phishing sites andalso report the sites so that we arrive with a large repository of phishing sites that can be avoided at any given time

12 REFERENCES:

[1] WC-PAD: Web crawling based Phishing Attack Detection

https://ieeexplore.ieee.org/abstract/document/8888416/

[2] A Methodical Overview on Phishing Detection along with an Organized Way to Construct an Anti-Phishing Framework

https://ieeexplore.ieee.org/abstract/document/8728356/

[3] Data mining a way to solve Phishing Attacks

https://ieeexplore.ieee.org/abstract/document/8550910/

[4] Detection of phishing attacks

https://ieeexplore.ieee.org/abstract/document/8355389/

[5] RR Phish: Anti-Phishing via Mining Brand Resource
https://ieeexplore.ieee.org/abstract/document/8326085/

[6] Learning from the Ones that Got Away: Detecting New Forms of Phishing Attacks

https://ieeexplore.ieee.org/abstract/document/8440723/

[7] Detecting Phishing Attacks Using Natural Language Processing and Machine Learning

https://ieeexplore.ieee.org/abstract/document/9065490/

[8] Intelligent rule-based phishing websites classification

https://ieeexplore.ieee.org/abstract/document/6040723/

[9] A Hybrid System to Find & Fight Phishing Attacks Actively

https://digital-library.theiet.org/content/journals/10.1049/iet-ifs.2013.0202

[10] Fuzzy Rough Set Feature Selection to Enhance Phishing Attack Detection

https://ieeexplore.ieee.org/abstract/document/8858884/

[11] Systematization of Knowledge (SoK): A Systematic Review of Software-Based Web Phishing Detection

https://ieeexplore.ieee.org/abstract/document/8036198/

[12] Favicon – a Clue to Phishing Sites Detection

https://ieeexplore.ieee.org/abstract/document/6805775/

[13] A new method for Detection of Phishing Websites: URL Detection

https://ieeexplore.ieee.org/abstract/document/8473085/

[14] Phish Box: An Approach for Phishing Validation and Detection

https://ieeexplore.ieee.org/abstract/document/8328445/

[15] Phishing Detection in Websites using Parse Tree Validation

https://ieeexplore.ieee.org/abstract/document/8443961/

[16] AI Meta-Learners and Extra-Trees Algorithm for the Detection of Phishing Websites

https://ieeexplore.ieee.org/abstract/document/9154378/

[17] Towards the Detection of Phishing Attacks
https://ieeexplore.ieee.org/abstract/document/9142967/

[18] Phishing Detection using Random Forest, SVM and Neural Network with Back propagation

https://ieeexplore.ieee.org/abstract/document/9277256/

[19] Automatic Detection of Phishing Target from Phishing Webpage

https://ieeexplore.ieee.org/abstract/document/5597725/

[20] Phishing Detection from URLs Using Deep Learning Approach

https://ieeexplore.ieee.org/abstract/document/9277459/

[21] Certain Investigation on Web Application Security: Phishing Detection and Phishing Target Discovery

https://ieeexplore.ieee.org/abstract/document/7586405/

[22] Detection of Phishing URL using Bayesian Optimized SVM Classifier

https://ieeexplore.ieee.org/abstract/document/9297412/

[23] Phishing Detection Using Machine Learning Technique

https://ieeexplore.ieee.org/abstract/document/9283771/

[24] An In-Depth Benchmarking and Evaluation of Phishing Detection Research for Security Needs

https://ieeexplore.ieee.org/abstract/document/8970564/

[25] On Effectiveness of Source Code and SSL Based Features for Phishing Website Detection

https://ieeexplore.ieee.org/abstract/document/9063824/