

A DYNAMIC AND COMBINED FRAMEWORK FOR PREDICTING PHISHING ATTACK

Submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering Degree in

Computer Science and Engineering

By

SUBBARA GANESH (Roll. No. 17SCS0443)

SIRIGIRI MAHIPAL REDDY (Roll. No. 17SCS5186)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI – 600119, TAMILNADU.

MARCH - 2021



SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**



Accredited with Grade “A” by NAAC

JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119

www.sathyabamauniversity.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **SUBBARA GANESH (Roll. No. 17SCS0443)** and **SIRIGIRI MAHIPAL REDDY (Roll. No. 17SCS5186)** who carried out the project entitled “**A DYNAMIC AND COMBINE FRAMEWORK FOR PREDICTING PHISHING ATTACK**” under my supervision from **August 2020 to March 2021**.

Internal Guide

Dr.L.Mary Gladence, M.E.,Ph.D.,

Head of the Department

Dr. S. Vigneshwari, M.E., Ph.D.,

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

Submitted for Viva voce Examination held on_____

Internal Examiner

External Examiner

DECLARATION

I **SUBBARA GANESH** here by declare that the Project Report entitled “**A DYNAMIC AND COMBINE FRAMEWORK FOR PREDECTING PHISHING ATTACK**” is done by me under the guidance of Department of **Dr.L.Mary Gladence, M.E.,Ph.D.**,Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.L.Mary Gladence, M.E.,Ph.D.**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the department of COMPUTER SCIENCE AND ENGINEERING who were helpful in many ways for the completion of project.

ABSTRACT

With the rapid development of web applications, and with the comfort provided by these web applications, internet users utilize these benefits to a great extent that they make almost all their day to day activities such as news paper reading, shopping, electricity bill payment, ticket booking and entertainment with the help of the internet. This phenomenon forces the users of the internet to get connected with the internet for a prolonged time and hence it increases the chances of the users to get caught in the web of phishing – an attack crafted by hackers to steal sensitive information by tempting the users with lucrative offers initially and then redirecting them to a fraudulent website where they can deceive the user by asking them to submit their credentials usually users submit their credentials without knowing that these are fake offers created with a sole intention of stealing sensitive information.

In spite of the alert and awareness given by the web community in this regard, more and more phishing artists succeed in their attack. Also these phishing artists develop novel attacks such as tab nabbing, website impersonation etc that attracts more and more internet users to be caught in the web of phishing. However many tools and methodologies have been developed to prevent phishing and to alert users orally and visually. But still the success rates of the phishing attack remains high and also the approaches related to phishing detection suffers high false positive and false negative ratio.

	TABLE OF CONTENTS	
CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF ABEVERSACTIONS	ix
1	INTRODUCTION	1
	1.1 INSIGHT TO PROJECT	1
	1.2 SCOPE OF THE PROJECT	1
	1.3 DOMAIN OVERVIEW	1
	1.3.1 MACHINE LEARNING	2
2	LITERATURE REVIEW	6
3	SYSTEM ANALYSIS	13
	3.1 SYSTEM ANALYSIS	13
	3.2 EXISTING SYSTEM	18
	3.3 DRAWBACKS OF EXISTING SYSTEM	19
	3.4 PROPOSED SYSTEM	19
	3.5 ADVANTAGES OF PROPOSED SYSTEM	19
4	REQUIREMENT SPECIFICATION	20
	4.1 INTRO OF REQUIREMENT SPECIFICATION	20
	4.2 HARDWARE & SOFTWARE SPECIFICATION	20
	4.2.1 HARDWARE SPECIFICATION	20
	4.2.2 SOFTWARE SPECIFICATION	20
	4.3 SOFTWARE DESCRIPTION	21
	4.3.1 ABOUT PYTHON	21
	4.3.2 ABOUT IDE	25
5	SYSTEM DESIGN & SYSTEM IMPLEMENTATION	28
	5.1 INTRODUCTION OF SYSTEM DESIGN	28

	5.2 ARCHITECTURE DIAGRAM	28
	5.3 LIST OF MODULES	29
	5.3.1 DATA ASSESSMENT	29
	5.3.2 PRE-PROCESSING	30
	5.3.3 FEATURE SELECTION	31
	5.3.4 PREDICTION	31
6	RESULT AND DISCUSSION	33
	6.1 RESULT	33
7	CONCLUSION AND FEATURE ENHANCEMENT	37
	7.1 CONCLUSION	37
	7.2 FEATURE ENHANCEMENT	37
	REFERENCE	38
	APPENDIX	40
	A. SOURCE CODE	40

	LIST OF FIGURES	
FIGURE NO	FIGURE NAME	PAGE NO
1.1	MACHINE LEARNING	3
1.2	SUPERVISED LEARNING	4
3.1	WATERFALL MODEL	14
3.2	RAPID APPLICATION DEVELOPMENT	15
4.1	ABOUT PYTHON	21
5.1	ARCHITECTURE DIAGRAM	28
6.1	USING ADAM ALGORITHM	33
6.2	USING RMSPROP ALGORITHM	34
6.3	USING SGD ALGORITHM	35
6.4	USING ADAM, RMSPROP, SGD ALGORITHM	35
6.5	PREDICTION RESULT	36

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
AI	ARTIFICIAL INTELLIGENCE
ML	MACHINE LEARNING
DSDM	DYNAMIC SYSTEMS DEVELOPMENT MODEL
XP	ETREME PROGRAMMING
FDD	FEATURE DRIVEN DEVELOPMENT
JAD	JOINT APPLICATION DEVELOPMENT
LD	LEAN DEVELOPMENT
RAD	RAPID APPLICATION DEVELOPMENT
RUP	RATIONAL UNIFIED PROCESS
SDLC	SYSTEM DEVELOPMENT LIFE CYCLE
IDE	INTEGRATED DEVELOPMENT ENVINORNMENT
UML	UNIFIED MODELING LANGUAGE
NLP	NATURAL LANGUAGE PROCESSING SYSTEM
TDD	TEST DROVE DEVELOPMENT

	CH
	APT
1.1 INSIGHT TO PROJECT	ER 1
	INT
	ROD
	UCT
	ION

In this proposed system various tools and methodologies used to prevent phishing has been analyzed and an efficient mechanism has been proposed to prevent phishing. In the existing system, the GA population consists of chromosomes or individuals that represent all possible feature subsets. It is essential in GA to define several parameters, such as the population size, the number of generations, and the crossover and mutation probabilities. In GA-based feature selection, each gene in the chromosome is encoded with a binary value, either one if the feature is selected or zero if not selected. In GA-based feature weighting, the genes represent the features' weights and are encoded with real values, which are usually between zero and one.

1.2 SCOPE OF THE PROJECT

The main contributions of this project therefore are:

- Data Analysis
- Dataset Preprocessing
- Training the Model
- Testing of Dataset

1.3 DOMINE OVERVIEW

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyze the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions

about the future.

Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine learning algorithms. For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are

capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

1.3.1 MACHINE LEARNING

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms.

However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyperparameters to reach an optimum result.

The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry.

Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

With an exponential increase in data, there is a need for having a system that can handle this massive load of data. Machine Learning models like Deep Learning allow the vast majority of data to be handled with an accurate generation of predictions. Machine Learning has revolutionized the way we perceive information and the various insights we can gain out of it.

These machine learning algorithms use the patterns contained in the training data to perform classification and future predictions. Whenever any new input is introduced to the ML model, it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using

various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows –

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

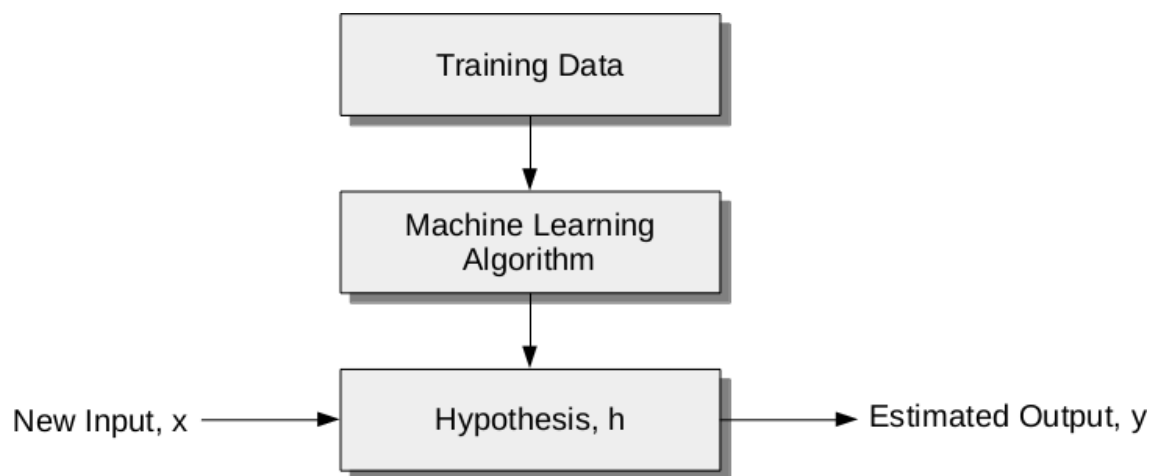


FIG 1.1: MACHINE LEARNING

Supervised Learning

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function $h(x)$ (sometimes called the “hypothesis”). “Learning” consists of using sophisticated mathematical algorithms to optimize this function so that, given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value $h(x)$ (say, market price for said house).

$$h(x_1, x_2, x_3, x_4) = \theta_0 + \theta_1 x_1 + \theta_2 x_3^2 + \theta_3 x_3 x_4 + \theta_4 x_1^3 x_2^2 + \theta_5 x_2 x_3^4 x_4^2$$

This function takes input in four dimensions and has a variety of polynomial terms. Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism’s genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”.
- Classification machine learning systems: Systems where we seek a yes-or-no prediction, such as “Is this tumor cancerous?”, “Does this cookie meet our quality standards?”, and so on.

In practice, x almost always represents multiple data points. So, for example, a housing price predictor might take not only square-footage (x_1) but also number of bedrooms (x_2), number of bathrooms (x_3), number of floors (x_4), year built (x_5), zip code (x_6), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a single input value is used.

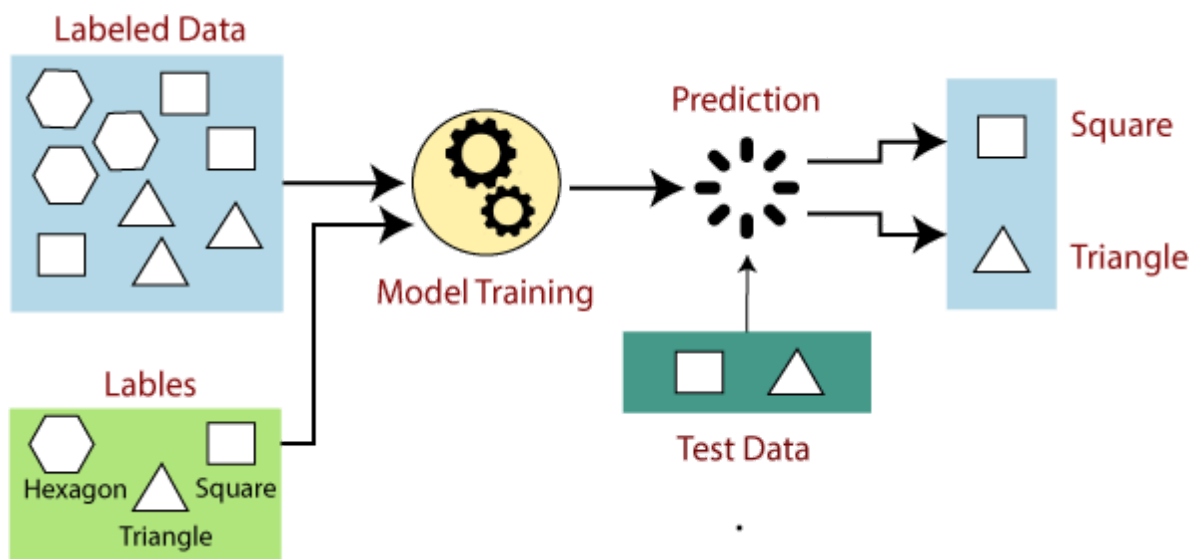


FIG 1.2 SUPERVISED LEARNING

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have

enough knowledge so that the model can accurately predict the output.

- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Regression

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

1. Linear Regression
2. Regression Trees
3. Non-Linear Regression
4. Bayesian Linear Regression
5. Polynomial Regression

Classification

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

LITERATURE SURVEY

C HAPTE R 2 LITERA TURE SURVE Y

Title: An Empirical Approach to Phishing Countermeasures Through Smart Glasses and Validation Agents.

Authors: JEMA DAVID NDIBWILE , EDITH TALINA LUHANGA , DOUDOU FALL , DAISUKE MIYAMOTO , GREGORY BLANC , AND YOUKI KADOBAYASHI.

Published Year- 2019

Efficiency

- ✓ Maximum Utilization of unstructured Data
- ✓ Elimination of Feature Engineering
- ✓ Ability to Deliver High Quality Results

Drawbacks

- ☐ Labor-intensive and time-consuming
- ☐ Establishing significant barriers for adapting to new domains
- ☐ Time-consuming and needs extra annotations

Description:

Phishing attacks have been persistent for more than two decades despite mitigation efforts from academia and industry. We believe that users fall victim to attacks not only because of lack of knowledge and awareness, but also because they are not attentive enough to security indicators and visual abnormalities on the webpages they visit. This is also probably why smart device users, who have more limited screen size and device capabilities compared to desktop users, are three times more likely to fall victim to phishing attacks. To assert our claim, we first investigated

general phishing awareness among different groups of smartphone users. We then used smart eyeglasses (electro-oculographic) to experimentally measure the mental effort and vigilance exhibited by users while surfing a website and while playing an Android phishing game that we developed. The results showed that knowledge and awareness about phishing do not seem to have a significant impact on security behaviours, as knowledgeable participants exhibited insecure

behaviours such as opening email attachments from unfamiliar senders. However, attentiveness was important as even participants with low cybersecurity knowledge could effectively identify attacks if they were reasonably attentive. Based on these results, we asserted that users are more likely to continue falling victim to phishing attacks due to insecure behaviours, unless tools to lessen the identification burden are provided. We thus recommended implementing a lightweight algorithm into a custom Android browser for detecting phishing sites deceptively without a user interaction. We used fake login credentials as validation agents and monitor the destination server HTTP responses to determine the authenticity of a webpage. We also presented initial evaluation results of this algorithm.

For the non-tech-savvy participants, we noticed two things: some participants with extremely higher mental effort did not necessarily get higher quiz scores in the game and some participants with lower mental effort did not necessarily get lower quiz scores. We believe that the participants who had lower quiz scores despite higher mental effort did not have even a slight idea of how the phishing sites look. On the other hand, the participants who got higher scores despite exhibiting less mental effort may have just guessed the correct answers or had a great knowledge of the subject. However, on average, non-tech-savvy participants who exhibited reasonable mental effort performed as well as much as most of the tech-savvy participants.

They found that participants who are intolerant of risk were more likely to regard legitimate interfaces as phishing. In contrast, participants who were more trusting and less curious performed better on a phishing security quiz. In our study, we found that women, who are usually more risk-averse in cyber and other behaviours than men were only less susceptible if they had higher education levels. This might be because higher education level means they are generally more computer-literate, and therefore understand or have had experiences with cyber-attacks such as computer viruses or spam messages. Statistically, our results show that the females whose education levels are at least bachelor degrees are less susceptible to phishing than men of the same education levels (21% vs. 38% respectively exhibited insecure behaviours). On the other hand, females with no bachelor degrees are more susceptible than men in the similar category (31% vs. 12.70%, respectively). Based on our conducted studies and experiments on smartphone

behaviours of users and the estimation of their mental efforts on identifying phishing attacks, we reckon that users require automated assistance for phishing attacks. Users are responsible for understanding how phishing attacks work and make efforts to learn about it through various awareness programs. However, leaving the entire burden of phishing identification to users may not be a plausible idea as we have seen that awareness alone is not enough. For instance, when a user is tired or exhausted she can make irrational decision despite being aware of the phishing attacks. Thus, we recommend an anti-phishing strategy that can independently and automatically detect phishing attacks and intervenes during the user login process. The key focus of the strategy is lowcomputational power requirement.

AUTHOR [2] Title: A Comprehensive Survey for Intelligent Spam Email Detection

Authors: Asif Karim, Sami Azam, Bharanidharan Shanmugam, Krishnan Kannoorpatti, Mamoun Alazab

Published Year- 2019

Efficiency

- ✓ Improved baseline performance
- ✓ Less training data
- ✓ Model generalize better

Drawbacks

- ☐ Can be computationally expensive.
- ☐ If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- ☐ It does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation

Description

The tremendously growing problem of phishing e-mail, also known as spam including spear phishing or spam borne malware, has demanded a need for reliable intelligent anti-spam e-mail filters. This survey paper describes a focused literature survey of Artificial Intelligence (AI) and Machine Learning (ML) methods for intelligent spam email detection, which we believe can help in developing appropriate countermeasures. In this paper, we considered 4 parts in the email's

structure that can be used for intelligent analysis: (A) Headers Provide Routing Information, contain mail transfer agents (MTA) that provide information like email and IP address of each sender and recipient of where the email originated and what stopovers, and final destination. (B) The SMTP Envelope, containing mail exchangers' identification, originating source and destination domains\users. (C) First part of SMTP Data, containing information like from, to, date, subject – appearing in most email clients (D) Second part of SMTP Data, containing email body including text content, and attachment. Based on the number the relevance of an emerging intelligent method, papers representing each method were identified, read, and summarized. Insightful findings, challenges and research problems are disclosed in this paper. This comprehensive survey paves the way for future research endeavors addressing theoretical and empirical aspects related to intelligent spam email detection

State that hackers even spoof the 'Date' field in SMTP header to keep their spam emails on top of receiver's Inbox, so that immediate attention can be gained. The authors have also suggested that ESP's should employ a dedicated 'Time Stamping Server' to authenticate the sending date for every email. Number of other researchers have proposed alteration (for instance, modifying some of the SMTP transactional steps) in the blueprint of SMTP to make it more secure and robust as the preferred choice for mail transfer protocol. However, such steps are not always practicable for multiple reasons as discussed later. This section will highlight few of such commendable research undertakings and a discussion on the hindering bottlenecks.

A highly successful method in earlier days of email communication, where large email vendors (Hotmail, Yahoo!, AOL etc.) mathematically calculated an alphanumeric strings of 32 to 128 characters, known as signature of the email (the Hash value), to store it in a database . The vendors work on the idea that spammers will send out a burst of spam emails to achieve their target and some of these spam emails will reach to their honeypot accounts, that is, account that had been set up specifically to catch spam emails. These vendors also rely on the fact that the generated signature will be largely different for spam to that of non-spam emails . Therefore, soon as the signature pattern matches to that of the spam pattern, it is added to a database for spam signature, and as other emails arrive at any of the

other customer accounts, those are instantly discarded if found spam - by matching the signature (calculated using the exact same method, from header and body, thus the method works in both part C and D to that of the one stored in the database, provided the record is found. Vendors supply the database to other Email Service Providers (ESP) and thus once an email is identified as spam, it is updated in several of these databases positioned throughout the globe.

The issues with Blacklists are that the spammers can frequently alter source address . Also the source address itself can be spoofed as mentioned earlier. In case of blacklists composed of URIs, spammers continuously set up cheap new domains before starting a fresh cycle of mass spamming, leaving the blacklists very little time to react instantly. Additionally, the list is often quite slow to be updated and thus rather ineffectual against phishing email threats that banks on user-visits at short-lived phishing websites.

AUTHOR [3]Title: Learning from the Ones that Got Away: Detecting New Forms of Phishing Attacks

Authors: Christopher N. Gutierrez , Taegyu Kim , Raffaele Della Corte , Jeffrey Avery , Dan Goldwasser , Marcello Cinque , Saurabh Bagchi

Published Year 2018

Efficiency

- ✓ Makes deep learning more accessible
- ✓ Helps solve complex real-world problems with several constraints
- ✓ Tackle problems like having little or almost no labeled data

avaliabliity Drawbacks

- ☐ Sensitive to noisy and missing data
- ☐ Does not work well with high
- ☐ dimensionality

Does not work well with large dataset

Description

Phishing attacks continue to pose a major threat for computer system defenders, often forming the first step in a multi-stage attack. There have been great strides

made in phishing detection; however, some phishing emails appear to pass through filters by making simple structural and semantic changes to the messages. We tackle this

problem through the use of a machine learning classifier operating on a large corpus of phishing and legitimate emails. We design SAFEPC (Semi-Automated Feature generation for Phish Classification), a system to extract features, elevating some to higher level features, that are meant to defeat common phishing email detection strategies. To evaluate SAFE-PC, we collect a large corpus of phishing emails from the central IT organization at a tier- 1 university. The execution of SAFE-PC on the dataset exposes hitherto unknown insights on phishing campaigns directed at university users. SAFE-PC detects more than 70% of the emails that had eluded our production deployment of Sophos, a state-of-the-art email filtering tool. It also outperforms SpamAssassin, a commonly used email filtering tool. We also developed an online version of SAFE-PC, that can be incrementally retrained with new samples. Its detection performance improves with time as new samples are collected, while the time to retrain the classifier stays constant.

The Legitimate or Benign dataset (Dataset 3) is composed of 158,444 benign emails, each with a header, subject line, and message body. The number of unique benign emails is 137,684 . However, we use a subset of the total legitimate emails for our evaluation. The final set consists 106,182 emails comprised from several public sources. About half of the legitimate emails were collected from public listserv hosted on .edu domains from universities in North America. The motivation for using these is that they are close to the spam emails sent to a university audience. The content includes university seminar announcements, academic listservs dealing with research, social activities, and a Mozilla bug reporting list. Unfortunately, we were unable to use samples of live university emails during the same period of the phishing dataset because of privacy reasons. The other half of legitimate emails are from Jeb Bush's publicly released emails during his time as governor of Florida, with only the subset related to financial transactions. The motivation for using these emails is they consist of real financial discussions that fit with our university context and can be considered close to phishing emails related to financial information. Prior work in phishing classification has used the Enron email set [13] as benign set. However, we wished to have emails that are closer to the phishing ones sent to our target university audience, and so we did not use this email set. The final mix of the unique legitimate emails composing dataset 3 is Jeb Bush Finance (53%), Listservs on Cytometry (33%), Mozilla (12%), Vision list (4%), and CompSci Colloquium (1%).

Phishing attackers continuously change topics to deceive users. An example is to send emails about Blackboard toward the beginning/end of semesters. Blackboard is used by students to check course-related information. details popular phishing emails each month, along with the number of phishing emails collected. These values are further separated into the number of phishing emails that were caught and not caught. Low observations in December and July could be due to an incomplete dataset or a genuinely low phishing activity. Some interesting (subjective) observations can be made from the data . The number of phishing emails significantly drops when students are on breaks (Winter/Summer sessions). The three super-categories that we create are Financial Institution, University, and Technology Provider. We find the financial institution and university features frequently appear in both the caught and uncaught datasets. The presence of a financial institution's name is an expected observation because attackers are motivated by monetary gains. The most popular financial institutions in our dataset are RHB, RBS, Western Union, Among university phishing campaigns, Blackboard was by far the most prevalent. The most popular IT providers are AOL and Google, the latter through Google Docs.

Several phishing emails that were classified as legitimate contain ambiguous messages. For example, some of the observed emails contained a link with a short greeting (2-4 words). These types of messages are quite challenging to classify because, without context, one cannot safely determine if the email is truly mischievous in nature. For example, receiving an email with only a URL from a colleague after a meeting could be considered acceptable, but receiving an unsolicited email with a URL from a stranger is suspicious. An automated technique will need to collect more external context to perform well with these challenging phishing emails. Other observations include emails that contain gibberish with a URL. Some of the other emails that were misclassified also appeared to be spam or "clickbait"

CHAPTE

R 3

3.1 SYSTEM ANALYSIS

SYSTEM ANALYSIS

One of the primary choices we face for our project implementations is “Which development methodology should we use?” There are various software development methodologies available in the ICT industry. They are

- Agile Software Development
- Crystal Methods
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Joint Application Development (JAD)
- Lean Development (LD)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum
- Spiral
- Systems Development Life Cycle (SDLC)
- Waterfall (a.k.a. Traditional)

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

Waterfall Model

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated

into a sequence of pre -defined phases.

1. WaterfallModel

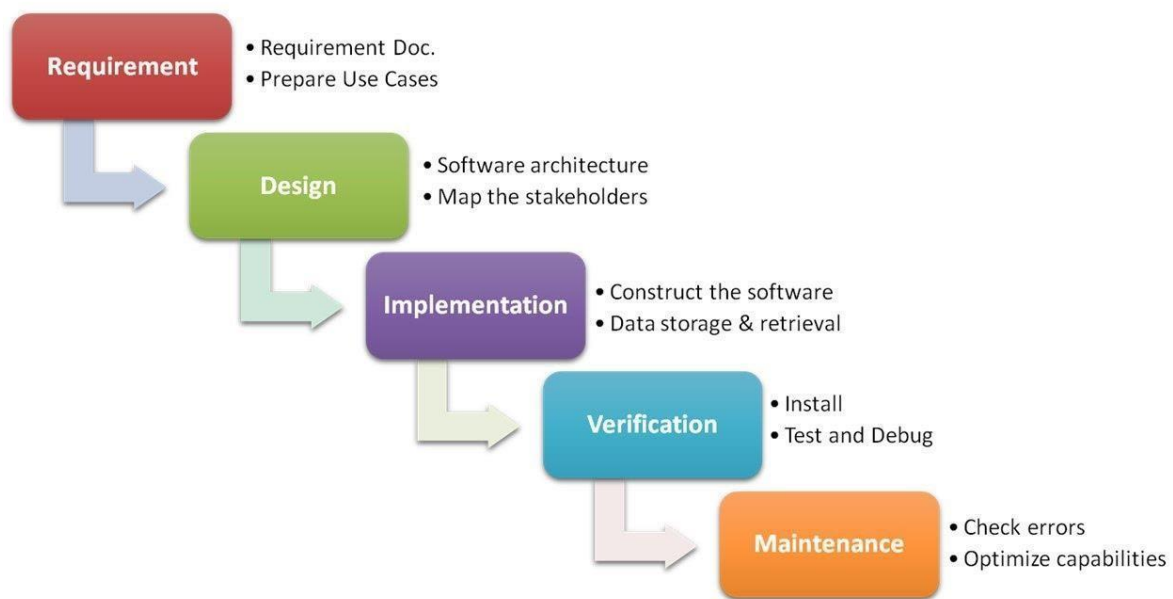


FIG 3.1 WATERFALL MODEL

In this stage, you should gather comprehensive data approximately what this challenge requires. You may gather this data in a diffusion of ways, from interviews to questionnaires to interactive brainstorming. By means of the stop of this section, the task requirements have to be clean, and also you have to have a necessities file that has been allotted on your team.

2. System design

Using the well-known requirements, your team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

3. Implementation

During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which are integrated at the end of this phase or the beginning of the next.

4. Testing

Once all coding is done, testing of the product can begin. Testers methodically find and report any problems. If serious issues arise, your project may need to return to phase one for reevaluation.

5. Delivery/deployment

In this phase, the solution is complete, and your project team submits the deliverables to be deployed or released.

6. Maintenance

The final solution has been implemented to the client and is being used. As troubles arises, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

RAD Model

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of an software application in comprehensive functional components.

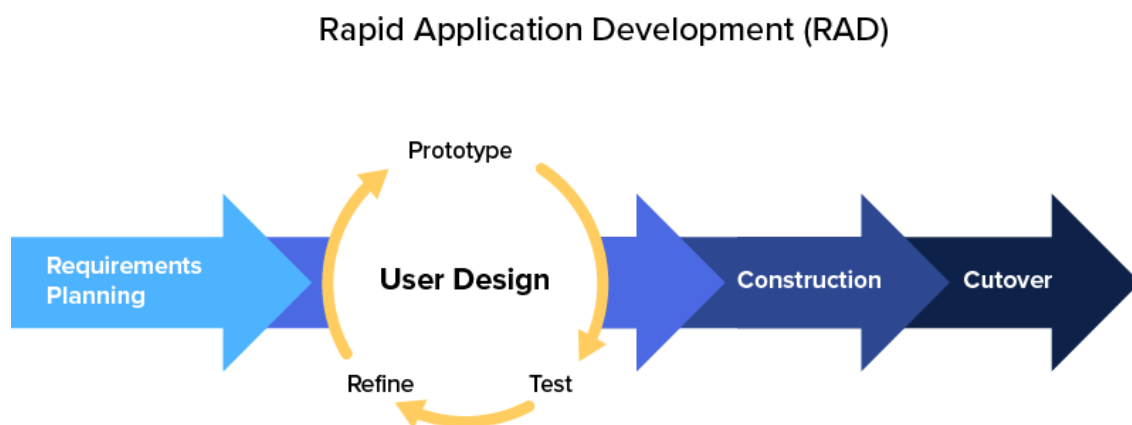


FIG 3.2 RAPID APPLICATION DEVELOPMENT

Phase 1: Requirements planning

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project.

During this stage, web application programmers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

Researching the current problem

Defining the requirements for the project

Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

Phase 2: User design

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until they reach a satisfactory design.

Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

Phase 3: Rapid construction

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.

Because the majority of the problems and changes were addressed during the thorough iterative design phase, web application programmers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

Preparation for rapid construction

Program and application development

Coding

Unit, integration, and system testing

The software development team of programmers, coders, testers, and web application programmers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

This third phase is important because the client still gets to give input throughout the process. They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

Phase 4: Cutover

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training.

All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

Here are just a few advantages:

RAD lets you break the project down into smaller, more manageable tasks.

The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.

Clients get a working product delivered in a shorter time frame.

Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

3.2 EXISTING SYSTEM

A phishing attack is initially conducted through web browsing or receiving a link within an e-mail, which appears to be from an authentic organisation, asking users to update their data. By clicking the link, the users will be directed to a website that could be phishing or legitimate.

In the existing system, the original phishing and legitimate data sets are collected, the raw data set must be prepared properly into an appropriate form with feature selection and machine learning.

In the existing system, the feature extraction aims at extracting the popular website features, which may help and contribute to the classification decision of the website's type, either phishing or legitimate. Several recent studies on phishing website detection have been carried out to investigate and identify the relevant and significant features that can be used to differentiate legitimate from phishing

websites.

3.3 DRAWBACKS OF EXISTING SYSTEM

- ✓ Cannot predict the new phishing attack
- ✓ Less improvement and minimal performance
- ✓ Does not uses effective feature selection method
- ✓ It fails when a specific attribute is not present,Not easy to implement

3.4 PROPOSED SYSTEM

The proposed system applies Fuzzy Rough Set (FRS) theory as a tool to select most effective features from three benchmarked data sets. The selected features are fed into three often used classifiers for phishing detection. a multilayer perceptron, a random forest, and sequential minimal optimization SMO.

The proposed system train each classifier on a separate out-of-sample dataset to examine the ability of FRS feature selection in developing a generalizable phishing detection. This training set which is composed by random extraction of websites from an online repository. All hyper-parameters are also set based on the values used in these previous works. To evaluate the ability for FRS feature selection to develop a generalizable phishing detector, we train each classifier on a separate out-of-sample dataset composed by random extraction of phishing and non-phishing websites from an online repository.

The proposed system reduces the false positive ratio of the phishing detectors. The main idea of this approach is to extract the possible domain name from the victim URL and then compare the page rank of the extracted domain names with the actual domain. If, there is a considerable difference in the ranks then the extracted domain name will be reported as phishing. This approach based on the domain rank can effectively detect phishing, because the phishing campaigns live only for a short period of time

3.5 ADVANTAGES OF PROPOSED SYSTEM

- ✓ Maximize the classification performance
- ✓ Higher classification sensitivity
- ✓ Increases the accuracy of the prediction
- ✓ Removes the irrelevant and redundant features
- ✓ High accuracy in complex application

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 INTRODUCTION OF REQUIREMENT SPECIFICATION

Clearly defined requirements are essential signs on the road that leads to a successful project. They establish a formal agreement between a client and a provider that they are both working to reach the same goal. High-quality, detailed requirements also help mitigate financial risks and keep the project on a schedule. According to the Business Analysis Body of Knowledge definition, requirements are a usable representation of a need.

Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management. In this article, we'll discuss the main types of requirements for software products and provide a number of recommendations for their use.

4.2 HARDWARE & SOFTWARE SPECIFICATION

4.2.1 HARDWARE SPECIFICATIONS

Hardware Requirement

Hardware	Minimum Requirement
----------	---------------------

Disk Space	32 GB or more, 10 GB or more for Foundation Edition
------------	---

Processor	1.4 GHz 64 bit
-----------	----------------

Memory	512 MB
--------	--------

Display	(800 × 600) Capable video adapter and monitor
---------	---

4.2.2 SOFTWARE SPECIFICATION

Backend Technologies

- ✓ Python
- ✓ Numpy
- ✓ Sci-learn

- ✓ Eclipse IDE
- ✓ Frontend Technologies
- ✓ Web Technologies
- ✓ Bootstrap

4.3 SOFTWARE DESCRIPTION

4.3.1 ABOUT PYTHON

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems.

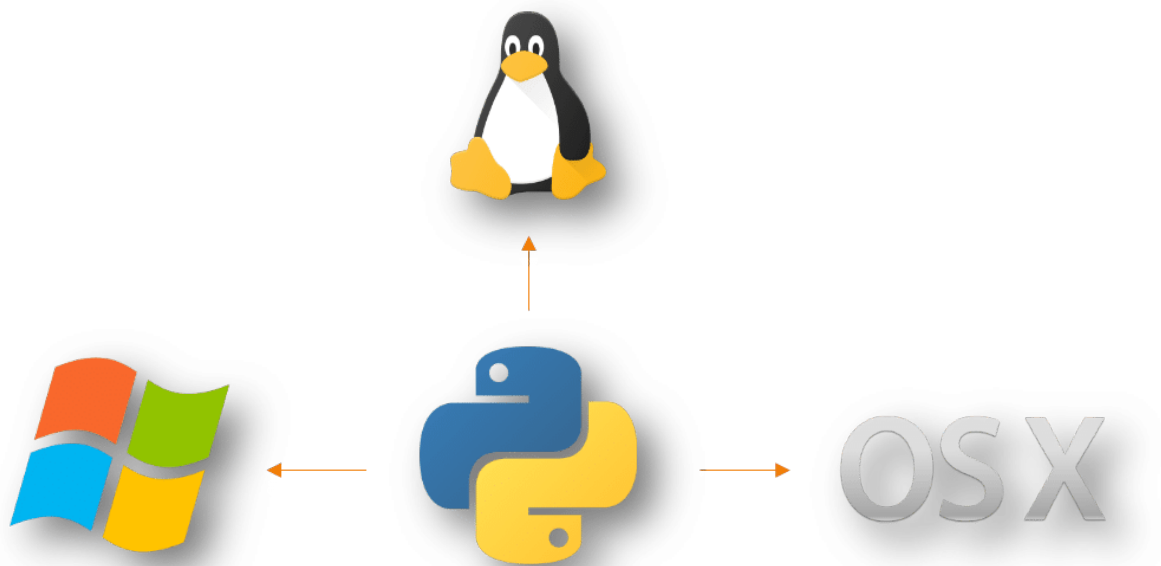


FIG 4.1: ABOUT PYTHON

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.

Python Data Types

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

- Universal Language Construct
- Support both High Level and Low Level Programming
- Language Interoperability
- Fastest Development life cycle therefore more productive coding environment
- Less memory used because a single container hold multiple data types and each type doesn't require its own function
- Learning Ease and open source development
- Speed and user-friendly data structure
- Extensive and extensible libraries.
- Simple & support IoT
- And many more

Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

- Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax
- Interactive mode
- Large and comprehensive standard libraries

- Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a difficulty when compared to C++ Java, Perl, and so forth

About Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure ("a one dimensional ndarray") that can store values — and for every value it holds a unique index, too.
2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and

the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

4.3.2 ABOUT IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications.

Programming Web application software engineers utilize wide scope of devices habitually all through application source code creation, building and application testing. Programming Development apparatuses frequently incorporate word

processors for source code creation, code libraries for reuse, compilers and test stages.

An Integrated Development Environment (IDE) is a product application that gives a programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE. The major reasons for using an IDE are given below:

We can easily write our code in its text editor window.

To store the resource files of a project, we have a project editor window.

It makes debugging of the code easier.

An IDE's autosuggest feature helps us in writing efficient code.

We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration. The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

- ✓ Komodo
- ✓ NetBeans
- ✓ PyCharm
- ✓ Python for Visual Studio.NET
- ✓ PyDev
- ✓ PyStudio
- ✓ LiClipse (Eclipse)

LiClipse is at the core for all Python based application development activities. It's been available for many years. LiClipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of LiClipse

- Perfect Debugger
- Error checking features
- An enhanced ability to analyze the code for quality and security concerns
- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Tooling
- Powerful Debugging options

IDE is a program that is used to write applications (other programs) in a variety of programming languages. It allows you to complete many distinct program development steps in one place (program window). A program which allows you to do this is known as an integrated development environment, or IDE. The program is free and easy to import onto your home computer.

CHAPTER 5

SYSTEM DESIGN AND SYSTEM IMPLEMENTATION

5.1 INTRODUCTION OF SYSTEM DESIGN

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

5.2 ARCHITECTURE DIAGRAM

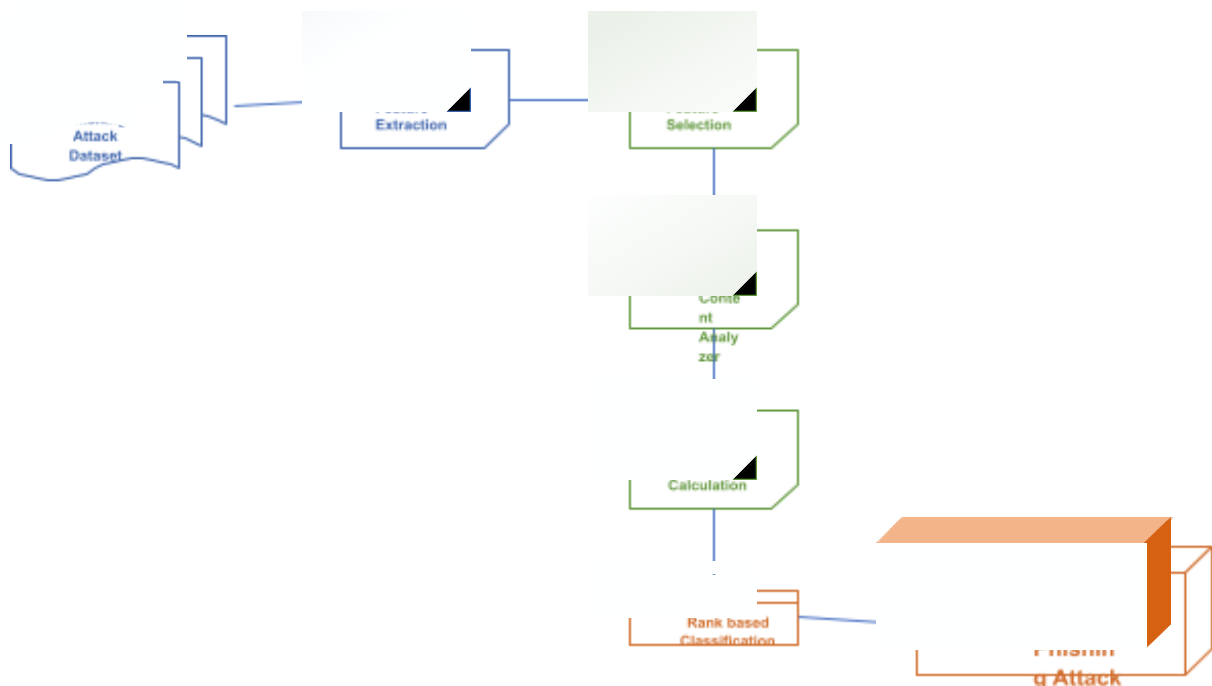


FIG 5.1: ARCHITECTURE DIAGRAM

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly.

5.3 LIST OF MODULES

Module 1 : Data Assessment

Module 2 : Pre-processing

Module 3 : Feature Selection

Module 4 : Prediction

5.3.1 DATA ASSESSMENT

At this stage, we explore variables one by one. Method to perform uni-variate analysis will depend on whether the variable type is categorical or continuous. Let's look at these methods and statistical measures for categorical and continuous variables individually:

Bi-variate Analysis finds out the relationship between two variables. Here, we look for association and disassociation between variables at a pre-defined significance level. We can perform bi-variate analysis for any combination of categorical and continuous variables. The combination can be: Categorical & Categorical, Categorical & Continuous and Continuous & Continuous. Different methods are used to tackle these combinations during analysis process.

Continuous & Continuous: While doing bi-variate analysis between two continuous variables, we should look at scatter plot. It is a nifty way to find out the relationship between two variables. The pattern of scatter plot indicates the relationship between variables. The relationship can be linear or non-linear.

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

5.3.2 PRE-PROCESSING

Text preprocessing is often the first step in the pipeline of a Natural Language Processing (NLP) system, with potential impact in its final performance. Despite its importance, text preprocessing has not received much attention in the deep learning literature. In this paper we investigate the impact of simple text preprocessing decisions (particularly tokenizing, lemmatizing, lowercasing and multiword grouping) on the performance of a standard neural text classifier. We perform an extensive evaluation on standard benchmarks from text categorization and sentiment analysis. While our experiments show that a simple tokenization of input text is generally adequate, they also highlight significant degrees of variability across preprocessing techniques.

The comparison of two web pages is done based on the similar contents on the web page. This technique makes use of Term Frequency/Inverse document Frequency (TF-IDF). TF-IDF compares the terms in the original website to the phishy one. Another approach is to capture the screen shots of a website and then process it to compare. This information retrieved from screen shots after processing can be given to a search engine to acquire its page rank and check the legitimacy of the website by comparing the content on it. Website logo can be used in this method to analyze

the webpage using the Google image database. 3. Blacklist Based Approach- A blacklist includes list of websites that are declared as spam. Such blacklists are maintained by organizations like Google. Spam URLs are added to this blacklist. The disadvantage of this approach is that a newly created phishing URL may not be present in the blacklist. Thus, such URLs will be left undetected. URLs present in the blacklist have denied access. This means a user cannot surf this webpage.

5.3.3 FEATURE SELECTION

Heuristic analysis takes three features Web content feature, URL feature, Web traffic feature all these features are extracted by the web crawler. URL Analyzer checks for the occurrence of '@' and '-' in URL, since '@' in a URL mean its left side can be discarded and only 59 characters are taken from the right side. and Legal sites does not use '-' very often. The URL of legitimate site does not contain many dots, only the phishing websites contains many dots. So the URL is also scanned for number of dots. The URL analyzer also checks the dictionary words and reports on the finding of misspelled words with levenshtein distance(ld). Levenshtein distance is used to calculate the difference in two strings sequence. If the distance is less, there is a possibility of phishing attack. URL Analyzer will also check whether the given URL contains the IP and whether that IP is the IP of its domain. Based on these Heuristics, the website will be classified as legal or phished website.

5.3.4 PREDICTION

When the training set for the current tree is drawn by sampling with replacement, about one-third of the cases are left out of the sample. This oob (out-of-bag) data is used to get a running unbiased estimate of the classification error as trees are added to the forest. It is also used to get estimates of variable importance.

After each tree is built, all of the data are run down the tree, and proximities are computed for each pair of cases. If two cases occupy the same terminal node, their proximity is increased by one. At the end of the run, the proximities are normalized by dividing by the number of trees. Proximities are used in replacing missing data, locating outliers, and producing illuminating low-dimensional views of the data.

Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the k th tree.

Put each case left out in the construction of the k th tree down the k th tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests.

In every tree grown in the forest, put down the oob cases and count the number of votes cast for the correct class. Now randomly permute the values of variable m in the oob cases and put these cases down the tree. Subtract the number of votes for the correct class in the variable- m -permuted oob data from the number of votes for the correct class in the untouched oob data. The average of this number over all trees in the forest is the raw importance score for variable m .

If the values of this score from tree to tree are independent, then the standard error can be computed by a standard computation. The correlations of these scores between trees have been computed for a number of data sets and proved to be quite low, therefore we compute standard errors in the classical way, divide the raw score by its standard error to get a z -score, and assign a significance level to the z -score assuming normality.

C

HAPT

ER 6

RESU

LT

AND

DISC

USSI

ON

6.1 RESULT

The reliability of the proposed framework lies in preventing users from being victims of phishing. Clearly, we can't guarantee that each web clients are specialists in web security and henceforth a tool to detect phishing is required to protect the web clients. The rundown based and heuristic-based methodologies can detect phishing but there is a bogus caution rate associated with these methodologies which are unsuitable. Then again the Machine-learning and multi-level characterization approaches improve the authenticity of the prediction, yet it fails to detect the image-based phishing content. Likewise, when on one side, the web client endures privacy and financial misfortune, because of unintended exposure of delicate data, on the opposite side the phishing sites generally mirror the site of a well-reputed association or an organization and consequently they will lose important clients and also the economic status. Nevertheless, only some have taken the required steps to contribute to the detection of phishing. As of late, a progressed form of phishing based on flash content, which bypasses the techniques with which detection occurs.

```
import tensorflow as tf

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=5)
model.evaluate(X_test, y_test)
```

Epoch 1/5
232/232 [=====] - 1s 5ms/step - loss: 0.1970 - accuracy: 0.9233
Epoch 2/5
232/232 [=====] - 1s 4ms/step - loss: 0.1389 - accuracy: 0.9436
Epoch 3/5
232/232 [=====] - 1s 5ms/step - loss: 0.1165 - accuracy: 0.9518
Epoch 4/5

FIG 6.1: USING ADAM ALGORITHM

So to fix these we propose a framework that can identify the phished content that has embedded objects and helps in target disclosure. The proposed phishing detection framework will be designed in a way that it will have an ideal exchange off between time uncertainty and detection precision

```
import tensorflow as tf

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=5)
model.evaluate(X_test, y_test)
```

```
Epoch 1/5
232/232 [=====] - 2s 8ms/step - loss: 0.2063 - accuracy: 0.9203
Epoch 2/5
232/232 [=====] - 2s 9ms/step - loss: 0.1418 - accuracy: 0.9437
Epoch 3/5
232/232 [=====] - 2s 8ms/step - loss: 0.1180 - accuracy: 0.9529
Epoch 4/5
232/232 [=====] - 2s 7ms/step - loss: 0.1073 - accuracy: 0.9552
Epoch 5/5
232/232 [=====] - 2s 7ms/step - loss: 0.0941 - accuracy: 0.9618
115/115 [=====] - 0s 2ms/step - loss: 0.1445 - accuracy: 0.9482

[0.14453916251659393, 0.9482049942016602]
```

FIG 6.2: USING RMSPROP ALGORITHM

Then again the Machine-learning and multi-level characterization approaches improve the authenticity of the prediction, yet it fails to detect the image-based phishing content. Likewise, when on one side, the web client endures privacy and financial misfortune, because of unintended exposure of delicate data, on the opposite side the phishing sites generally mirror the site of a well-reputed association or an organization and consequently they will lose important clients and also the economic status.

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

```
import tensorflow as tf

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=5)
model.evaluate(X_test, y_test)

Epoch 1/5
232/232 [=====] - 1s 5ms/step - loss: 0.4858 - accuracy: 0.8154
Epoch 2/5
232/232 [=====] - 1s 5ms/step - loss: 0.2501 - accuracy: 0.9139
Epoch 3/5
232/232 [=====] - 1s 4ms/step - loss: 0.2025 - accuracy: 0.9259
Epoch 4/5
232/232 [=====] - 1s 5ms/step - loss: 0.1846 - accuracy: 0.9284
Epoch 5/5
232/232 [=====] - 1s 4ms/step - loss: 0.1749 - accuracy: 0.9332
115/115 [=====] - 0s 2ms/step - loss: 0.1818 - accuracy: 0.9279

[0.181759312748909, 0.9279254674911499]
```

FIG 6.3: USING SGD ALGORITHM

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

```
l=[0.9559,0.9512,0.9282]
l1 = ['Adam','RMSProp','SGD']
import matplotlib.pyplot as plt
plt.ylabel('Test Accuracy of Model2')
plt.xlabel('Optimizers')
plt.plot(l1,l)

[<matplotlib.lines.Line2D at 0x2788c61ed60>]
```

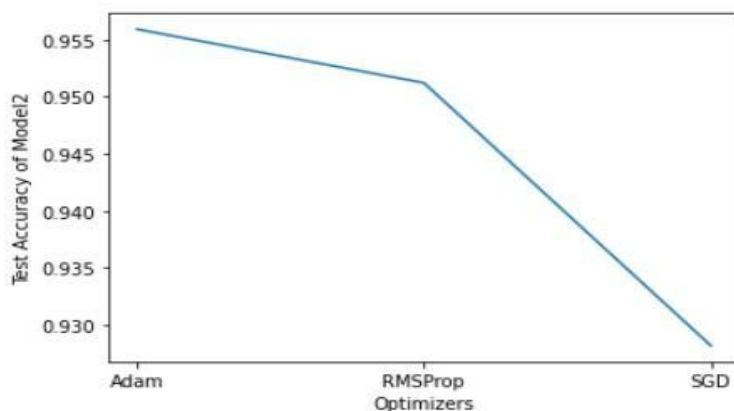


FIG 6.4: USING ADAM, RMSPROP, SGD ALGORITHM

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

```
import numpy as np
l = np.array([[1,0,1,1,1,-1,-1,-1,-1,1,1,-1,1,0,-1,-1,-1,-1,0,1,1,1,1,-1,1,-1,1,0,-1]])
p= model.predict(l)
```

```
round(p[0][0])
```

0

```
round(pred[0][0])
```

0

```
y_test[0]
```

0

```
round(pred[1][0])
```

1

```
print(y_test[9])
pred = model.predict(X_test)
```

0

```
y_test[20]
```

0

```
round(pred[20][0])
```

0

FIG 6.5:PREDECTION RESULTS

CHAPTER 7

CONCLUSION & FEATURE ENHANCEMENT

7.1 CONCLUSION

The credibility of a phishing detector lies in protecting the individuals from falling prey to phishing. Obviously we cannot claim that every web users are security experts and hence a sophisticated phishing tool is required to protect the web users. The list based and heuristic based approaches, though can detect the phishing website the observed false alarm rate with this approach is quite unacceptable. On the other hand the machine learning and multi-tier classification approaches improves the detection accuracy, but it is inefficient in detecting image based phishing content. Also, when on one side ,the web users suffers privacy and economic loss ,due to the unintended disclosure of sensitive information, on the other side the phishing sites jeopardize the impersonated site which is usually a reputed organization and hence they tend to lose valuable customer and money. However, only a few researches have taken effort to discover the targeted phishing site. Recently, a more sophisticated form of phishing based on flash content is prevalent, which by passes the strategy of most phishing. So, we propose a phishing detector, which can detect phishing content containing embedded objects and aids in target discovery. The proposed phishing tool will be designed in such a way that it will have a perfect trade-off between time complexity and detection accuracy.

7.2 FEATURE ENHANCEMENT

For future work, we intend to create a prototype implementation of an email account management tool, that can be used to evaluate the real-world applicability of the proposed address separation method, whether a positive effect in user awareness can be observed, and whether the additional context information given by address separation can be used to improve automated detection of phishing emails.

REFERENCES

- N. Shrestha, R. K. Kharel, J. Britt and R. Hasan, "High- performance Classification of Phishing URLs Using a Multi-modal Approach with MapReduce", 2015 IEEE World Congress on Services, New York, NY, 2015, pp. 206-212. [6]
- E. H. Chang, K. L. Chiew, S. N. Sze and W. K. Tiong, "Phishing Detection via Identification of Website Identity", 2013 International Conference on IT Convergence and Security (ICITCS), Macao, 2013, pp. 1-4.
- M. Aydin and N. Baykal, "Feature Extraction and Classification Phishing Websites Based on URL" , 2015 IEEE Conference on Communications and Network Security (CNS), Florence, 2015, pp. 769-770.
- P. Singh, N. Jain and A. Maini, "Investigating the Effect Of Feature Selection and Dimensionality Reduction On Phishing Website Classification Problem", 2015 1st International Conference on Next Generation Computing Technologies (NGCT), Dehradun, 2015, pp. 388-393.
- M. A. U. H. Tahir, S. Asghar, A. Zafar and S. Gillani "A Hybrid Model to Detect Phishing-Sites using Supervised Learning Algorithms" , 2016 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, 2016, pp. 1126-1133.
- E. Buber, . Demir and O. K. Sahingoz "Feature selections for the machine learning based detection of phishing websites" , 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, 2017, pp. 1-5.
- Z. Dong, A. Kapadia, J. Blythe and L. J. Camp "Beyond the Lock Icon: Real-time Detection of Phishing Websites Using Public Key Certificates" , 2015 APWG Symposium on Electronic Crime Research (eCrime), Barcelona, 2015, pp. 1-12.
- Anti-Phishing Working Group. Accessed: Sep. 2016. [Online]. Available <http://www.antiphishing.org>
- Naga Venkata Sunil and A. Sardana, "A PageRank based detection technique for phishing web sites," 2012 IEEE Symposium on Computers & Informatics (ISCI), Penang, 2012, pp. 58-63. doi: 10.1109/ISCI.2012.6222667
- N. Gutierrez et al., "Learning from the Ones that Got Away: Detecting New Forms of Phishing Attacks," in IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 6, pp. 988-1001, 1 Nov.-Dec. 2018. doi: 10.1109/TDSC.2018.2864993

R. Dhamija and J.D. Tygar, "The Battle against Phishing: Dynamic SecuritySkins", Proc. Symp. Usable Privacy and Security, 2005, pp 77-88. Mobile Marketing Statistics. Accessed: Mar. 2017.

Chandrashekar, G., Sahin, F.: 'A survey on feature selection methods', Comput. Electr. Eng., 2014, 40, (1), pp. 16–28

Sun, X., Liu, Y., Xu, M., et al.: 'Feature selection using dynamic weights for classification', Knowl.-Based Syst., 2013, 37, pp. 541–549

Zhao, M., Fu, C., Ji, L., et al.: 'Feature selection and parameter optimization for support vector machines: A new approach based on genetic algorithm with feature chromosomes', Expert Syst. Appl., 2011, 38, (5), pp. 5197–5204

Kohavi, R., John, G.H.: 'Wrappers for feature subset selection', Artif. Intell., 1997, 97, (1–2), pp. 273–324

Das, A., Das, S.: 'Feature weighting and selection with a pareto-optimal trade-off between relevancy and redundancy', Pattern Recognit. Lett., 2017, 88, pp. 12–19

Varpa, K., Iltanen, K., Juhola, M.: 'Genetic algorithm based approach in attribute weighting for a medical data set', J. Comput. Med., 2014, 2014, pp. 1–11

Pérez-Rodríguez, J., Arroyo-Peña, A.G., García-Pedrajas, N.: 'Simultaneous instance and feature selection and weighting using evolutionary computation', Appl. Soft Comput., 2015, 37, pp. 416–443

Paul, S., Das, S.: 'Simultaneous feature selection and weighting – An evolutionary multi-objective optimization approach', Pattern Recogn. Lett., 2015, 65, pp. 51–59

APPENDIX

A.SOURCE CODE

```
import pandas as pd
import io
df = pd.read_csv('csv_result-Phishing_features.csv')
df.head()
y = df['Result']
y=y.replace(-1,0)
y
X = df.iloc[:,1:31]
X.head()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
random_state=42)
X_train,X_test = X_train.values,X_test.values
y_train,y_test = y_train.values,y_test.values
import tensorflow as tf
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5)
model.evaluate(X_test, y_test)
l=[0.9221,0.9450,0.9548,0.9575,0.9641]
```

```

l1 = [1,2,3,4,5]

import matplotlib.pyplot as plt
%matplotlib inline

plt.ylabel('Train Accuracy of Model2-Adam')
plt.xlabel('Epochs')
plt.plot(l1,l)

import tensorflow as tf

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=5)
model.evaluate(X_test, y_test)

l=[0.9184,0.9432,0.9514,0.9584,0.9610]

l1 = [1,2,3,4,5]

import matplotlib.pyplot as plt

plt.ylabel('Train Accuracy of Model2-RMSProp')
plt.xlabel('Epochs')
plt.plot(l1,l)

import tensorflow as tf

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),

```

```

tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5)
model.evaluate(X_test, y_test)
l=[0.8415,0.9237,0.9295,0.9328,0.9341]
l1 = [1,2,3,4,5]
import matplotlib.pyplot as plt
plt.ylabel('Train Accuracy of Model2-SGD')
plt.xlabel('Epochs')
plt.plot(l1,l)
l=[0.9559,0.9512,0.9282]
l1 = ['Adam','RMSProp','SGD']
import matplotlib.pyplot as plt
plt.ylabel('Test Accuracy of Model2')
plt.xlabel('Optimizers')
plt.plot(l1,l)
import tensorflow as tf
model1 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(1024, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model1.compile(optimizer='adam',
              loss='binary_crossentropy',

```

```

        metrics=['accuracy'])
model1.fit(X_train, y_train, epochs=5)
model1.evaluate(X_test, y_test)
l=[0.9194,0.9430,0.9537,0.9599,0.9644]
l1 = [1,2,3,4,5]
import matplotlib.pyplot as plt
plt.ylabel('Train Accuracy of Model1-Adam')
plt.xlabel('Epochs')
plt.plot(l1,l)
import tensorflow as tf
model1 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(1024, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model1.compile(optimizer='rmsprop',
               loss='binary_crossentropy',
               metrics=['accuracy'])
model1.fit(X_train, y_train, epochs=5)
model1.evaluate(X_test, y_test)
l=[0.9191,0.9442,0.9511,0.9584,0.9598]
l1 = [1,2,3,4,5]
import matplotlib.pyplot as plt
plt.ylabel('Train Accuracy of Model1-RMSProp')
plt.xlabel('Epochs')
plt.plot(l1,l)
import tensorflow as tf

```

```

model1 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(1024, activation=tf.nn.relu),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])

```

```

model1.compile(optimizer='sgd',
               loss='binary_crossentropy',
               metrics=['accuracy'])

```

```

model1.fit(X_train, y_train, epochs=5)

```

```

model1.evaluate(X_test, y_test)

```

```

l=[0.7918,0.9160,0.9241,0.9291,0.9337]

```

```

l1 = [1,2,3,4,5]

```

```

import matplotlib.pyplot as plt

```

```

plt.ylabel('Train Accuracy of Model1-SGD')

```

```

plt.xlabel('Epochs')

```

```

plt.plot(l1,l)

```

```

l=[0.9479,0.9529,0.9282]

```

```

l1 = ['Adam','RMSPProp','SGD']

```

```

import matplotlib.pyplot as plt

```

```

plt.ylabel('Test Accuracy of Model1')

```

```

plt.xlabel('Optimizers')

```

```

plt.plot(l1,l)

```

```

import tensorflow as tf

```

```

model1 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(1024,

```

```
activation=tf.nn.relu),
```

```

tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(2, activation=tf.nn.sigmoid)
])
model1.compile(optimizer='sgd',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])
model1.fit(X_train, y_train, epochs=5)
model1.evaluate(X_test, y_test)
import numpy as np
l = np.array([[1,0,1,1,1,-1,-1,-1,-1,1,1,-1,1,0,-1,-1,-1,-1,0,1,1,1,1,1,-1,1,-1,1,0,-1]])
p= model.predict(l)
l=[0.9182,0.9372,0.9534,0.9550,0.9577]
l3=[0.8079,0.9041,0.9164,0.9222,0.9287]
l2=[0.9125,0.9430,0.9498,0.9552,0.9602]
l1 = [1,2,3,4,5]
import matplotlib.pyplot as plt
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.plot(l1,l2,l)
plt.plot(l1,l3,l)
import tensorflow as tf
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',

```

```

        metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5)
model.evaluate(X_test, y_test)
import tensorflow as tf
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(256, activation=tf.nn.relu),
    tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model.compile(optimizer='sgd',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5)
model.evaluate(X_test, y_test)
adam=[0.9182,0.9372,0.9534,0.9550,0.9577]
rms = [0.9020,0.9369,0.9491,0.9522,0.9571]
sgd = [0.8327,0.9126,0.9179,0.9249,0.9283]
l = [1,2,3,4,5]
plt.plot(adam,l)
plt.plot(sgd,l)
plt.plot(rms,l)
plt.xlabel("Epochs")
plt.ylabel("Train Accuracy")
import tensorflow as tf
model1 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(512, activation=tf.nn.relu),
    tf.keras.layers.Dense(1024,

```

```
activation=tf.nn.relu),
```

```
tf.keras.layers.Dropout(0.2),
tf.keras.layers.Dense(1, activation=tf.nn.sigmoid)
])
model1.compile(optimizer='adam',
               loss='binary_crossentropy',
               metrics=['accuracy'])
```

```
model1.fit(X_train, y_train, epochs=5)
pred = model1.evaluate(X_test, y_test)
adam=[0.9132, 0.9415,0.9544,0.9606,0.9607]
rms = [0.9035,0.9376,0.9449,0.9529,0.9557]
sgd = [0.8342,0.9068,0.9159,0.9210,0.9270]
l = [1,2,3,4,5]
plt.plot(adam,l)
plt.plot(sgd,l)
plt.plot(rms,l)
plt.ylabel("Epochs")
plt.xlabel("Train Accuracy")
pred
```