

**DETECTION OF PHISHING WEBSITE USING NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING**

**A PROJECT REPORT**

***Submitted by***

**GIRISWARNA.R [REGISTER NO:211417104067]**

**HARINI.P [REGISTER NO:211417104078]**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE**

**ANNA UNIVERSITY: CHENNAI 600 025**

**AUGUST 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report **“DETECTION OF PHISHING WEBSITE USING NATURAL LANGUAGE PROCESSING AND MACHINE LEARNING”** is the bonafide work of **“GIRISWARNA R(211417104067)** **and** **HARINI P (211417104078)”** who carried out the project work under my supervision.



| **SIGNATURE** | **SIGNATURE** |
| --- | --- |
| **Dr. S. MURUGAVALLI, M.E., Ph.D.,**  **HEAD OF DEPARTMENT**      DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI,  POONAMALLEE,  CHENNAI-600 123. | **M. SANGEETHA, M.Tech.,**  **ASSOCIATE PROFESSOR**  DEPARTMENT OF CSE,  PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI,  POONAMALLEE,  CHENNAI-600 123 |

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on **05.08.2021**.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to express our heartfelt and sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHIKUMAR, M.E., Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E.,M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of the CSE Department, **Dr. S.MURUGAVALLI , M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Mrs.M. SANGEETHA , M.Tech.,** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

**GIRISWARNA R(211417104067)**

**HARINI P(211417104078)**

**ABSTRACT**

Malicious Web sites largely promote the growth of Internet criminal activities and constrain the development of Web services. As a result, there has been strong motivation to develop systemic solutions to stopping the user from visiting such Websites. We propose a learning based approach to classifying Web sites into 3 classes: Benign, Spam and Malicious. Our mechanism only analyzes the Uniform Resource Locator (URL) itself without accessing the content of Web sites. Thus, it eliminates the run-time latency and the possibility of exposing users to the browser based vulnerabilities. By employing learning algorithms, our scheme achieves better performance on generality and coverage compared with blacklisting service.

URLs of the websites are separated into 3 classes:

* Benign: Safe websites with normal services
* Spam: Website performs the act of attempting to flood the user with advertising or sites such as fake surveys and online dating etc.
* Malware: Website created by attackers to disrupt computer operation, gather sensitive information, or gain access to private computer systems.

| **CHAPTER**  **NO** | **TITLE** | **PAGE NO** |
| --- | --- | --- |
|  | **ABSTRACT** | iii |
|  | **LIST OF TABLES** | vi |
|  | **LIST OF FIGURES** | vii |
|  | **LIST OF SYMBOLS, ABBREVIATIONS** | ix |
| **1.** | **INTRODUCTION** | 01 |
|  | 1.1 OVERVIEW | 02 |
|  | 1.2 PROBLEM DEFINITION | 03 |
| **2.** | **LITERATURE SURVEY** | 04 |
| **3.** | **SYSTEM ANALYSIS** | 13 |
|  | 3.1 EXISTING SYSTEM | 14 |
|  | 3.2 PROPOSED SYSTEM | 15 |
|  | 3.3 REQUIREMENT ANALYSIS AND SPECIFICATION | 16 |
|  | 3.3.1 Input Requirements | 16 |
|  | 3.3.2 Output Requirements | 16 |
|  | 3.4 THE PYTHON PROGRAMMING LANGUAGE | 16 |
|  | 3.4.1 History of Python | 17 |
|  | 3.4.2 Why the name Python? | 18 |
|  | 3.4.3 Python Features | 19 |
|  |  |  |
|  | 3.5 MACHINE LEARNING | 21 |
|  | 3.5.1 Machine learning vs Traditional programming | 22 |
|  | 3.5.2 How does Machine Learning work? | 22 |
|  | 3.5.3 Machine Learning Algorithms and Where they are used? | 25 |
|  | 3.5.4 Application of Machine Learning | 30 |
|  | 3.5.5 Example of Application of Machine Learning in Supply chain | 31 |
|  | 3.5.6 Deep Learning | 32 |
|  | 3.5.7 Difference between Machine learning and Deep learning | 35 |
|  | 3.5.8 When to use ML or DL? | 36 |
|  | 3.6 ALGORITHM | 37 |
|  | 3.7 TECHNOLOGY STACK | 39 |
|  | 3.7.1 Software Requirements | 39 |
|  | 3.7.2 Hardware Requirements | 39 |
| **4.** | **SYSTEM DESIGN** | 45 |
|  | 4.1 DATA FLOW DIAGRAMS | 46 |
|  | 4.2 UML DIAGRAM | 49 |
|  | 4.2.1 Use Case Diagram | 49 |
|  | 4.2.2 Class Diagram | 51 |
|  | 4.2.3 Sequence Diagram | 52 |
|  | 4.2.4 Activity Diagram | 53 |
|  |  |  |
| **5.** | **SYSTEM ARCHITECTURE** | 54 |
|  | 5.1 ARCHITECTURE OVERVIEW | 55 |
|  | 5.2 MODULE DESCRIPTION | 56 |
|  | 5.2.1 Phishing Website Features | 56 |
|  | 5.2.2 Using Pop-Up Window | 58 |
|  | 5.2.3 Classification | 58 |
| **6.** | **SYSTEM IMPLEMENTATION** | 59 |
|  | 6.1 CODING | 60 |
| **7** | **SYSTEM TESTING** | 87 |
|  | 7.1 SYSTEM TESTING | 88 |
|  | 7.1.1 Unit Testing | 88 |
|  | 7.1.2 Integration Testing | 88 |
|  | 7.1.3 Regression Testing | 89 |
|  | 7.1.4 Smoke Testing | 89 |
|  | 7.1.5 Alpha Testing | 89 |
|  | 7.1.6 Beta Testing | 89 |
|  | 7.1.7 System Testing | 90 |
|  | 7.1.8 Stress Testing | 90 |
|  | 7.1.9 Performance Testing | 90 |
|  | 7.1.10 Object Oriented Testing | 90 |
|  | 7.2 TEST CASES & REPORTS / PERFORMANCE ANALYSIS | 91 |
|  | 7.3 EVALUATION METRICS | 92 |
| **8** | **CONCLUSION AND FUTURE ENHANCEMENTS** | 93 |
|  | 8.1 CONCLUSION  8.2 FUTURE WORK | 94  94 |
|  | **APPENDICES** | 95 |
|  | A.1 SAMPLE SCREENSHOT  A.2 PUBLICATION | 96  101 |
|  | **REFERENCES** | 102 |

**LIST OF TABLES**

| **TABLE NO** | **TABLE** | **PAGE NO** |
| --- | --- | --- |
| 3.5.7 | DIFFERENCE BETWEEN MACHINE LEARNING AND DEEP LEARNING | 35 |
| 7.2 | TEST CASES AND REPORT | 91 |

**LIST OF FIGURES**

| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO** |
| --- | --- | --- |
| 3.5.1 | ML vs TP | 22 |
| 3.5.2 | Machine Learning Works | 23 |
| 3.5.3 | Interface from Model | 24 |
| 3.5.8 | Comparison Graph | 37 |
| 4.1.1 | Data Flow Diagram | 48 |
| 4.2.1 | Use case Diagram | 50 |
| 4.2.2 | Class Diagram | 51 |
| 4.2.3 | Sequence Diagram | 52 |
| 4.2.4  5.1.1  A.1.1  A.1.2  A.1.3  A.1.4  A.1.5 | Activity Diagram  System Architecture  Decision Tree  RF Classifier  SVM Classifier  Benign Sample  Malware Sample | 53  55  96  97  98  99  100 |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF ABBREVIATION**

| ML  TP  RF  SVM | Machine learning  Traditional Programming  Random Forest Algorithm  Support Vector Machine |
| --- | --- |

**CHAPTER 1**

**INTRODUCTION**

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

We propose a learning based approach to classifying Web sites into 3 classes: Benign, Spam and Malicious. Our mechanism only analyzes the Uniform Resource Locator (URL) itself without accessing the content of Web sites. Thus, it eliminates the run-time latency and the possibility of exposing users to the browser based vulnerabilities. URLs of the websites are separated into 3 classes:Benign: Safe websites with normal services, Spam: Website performs the act of attempting to flood the user with advertising or sites such as fake surveys and online dating etc, Malware: Website created by attackers to disrupt computer operation, gather sensitive information, or gain access to private computer systems.

**1.2 PROBLEM DEFINITION**

While the Internet has brought unprecedented convenience to many people for managing their finances and investments, it also provides opportunities for conducting fraud on a massive scale with little cost to the fraudsters. Fraudsters can manipulate users instead of hardware/software systems, where barriers to technological compromise have increased significantly. Phishing is one of the most widely practised Internet frauds. It focuses on the theft of sensitive personal information such as passwords and credit card details. Phishing attacks take two forms:

* attempts to deceive victims to cause them to reveal their secrets by pretending to be trustworthy entities with a real need for such information
* attempts to obtain secrets by planting malware onto victims’ machines.

The specific malware used in phishing attacks is subject of research by the virus and malware community and is not addressed in this thesis. Phishing attacks that proceed by deceiving users are the research focus of this thesis and the term ‘phishing attack’ will be used to refer to this type of attack.

**CHAPTER 2**

**LITERATURE SURVEY**

**CHAPTER 2**

**LITERATURE SURVEY**

**Paper 1 : Learning to Detect Phishing URLs**

**Ram B. Basnet ,Andrew H. Sung -IJRET , 03(06)**

Phishing websites, fraudulent sites that impersonate atrusted third party to gain access to private data, continueto cost Internet users over a billion dollars each year. Inthis paper, we describe the design and performance characteristics of a scalable machine learning classifier we developed to detect phishing websites. We use this classifierto maintain Google’s phishing blacklist automatically. Ourclassifier analyzes millions of pages a day, examining theURL and the contents of a page to determine whether ornot a page is phishing. Unlike previous work in this field,we train the classifier on a noisy dataset consisting of millions of samples from previously collected live classificationdata. Despite the noise in the training data, our classifierlearns a robust model for identifying phishing pages whichcorrectly classifies more than 90% of phishing pages several weeks after training concludes.

**Paper 2 :Malicious web Page detection based on on-line learning algorithm**

**Wen Zhang, Yu-Xin Ding, Yan Tang, Bin Zhao - IEEE ,10(11)**

The Internet has become an indispensable tool in peoples' daily life. It also bring us serious computer security problem. One big security threat comes from malicious webpages. In this paper we study how to detect malicious pages. Since malicious webpages are generated inconstantly, we use on line learning methods to detect malicious webpages. To keep the client side as safe as possible, we do not download the webpages, and analysis webpages' content. We only use URL information to determine if the URL links to a malicious pages. The feature selection methods for URL are discussed, and the performances of different on line learning methods are compared. To improve the performance of on line learning classifiers, an improved on line learning method is proposed, experiments show that this method is effective.

**Paper 3 :Real Time Detection of Phishing Website**

**Ahmed,Abdullah - IEEE ,7(11)**

Web Spoofing lures the user to interact with the fake websites rather than the real ones. The main objective of this attack is to steal the sensitive information from the users. The attacker creates a `shadow' website that looks similar to the legitimate website. This fraudulent act allows the attacker to observe and modify any information from the user. This paper proposes a detection technique of phishing websites based on checking Uniform Resources Locators (URLs) of web pages. The proposed solution is able to distinguish between the legitimate web page and fake web page by checking the Uniform Resources Locators (URLs) of suspected web pages. URLs are inspected based on particular characteristics to check the phishing web pages. The detected attacks are reported for prevention. The performance of the proposed solution is evaluated using Phistank and Yahoo directory datasets. The obtained results show that the detection mechanism is deployable and capable to detect various types of phishing attacks maintaining a low rate of false alarms.

**Paper 4 : Holistic Analysis and Detection of Malicious Web pages**

**Birhanu Eshete, Adolfo Villafiorita , and Komminist Weldemariam - International Conference on Security and Privacy in Communication System, volume:106**

Malicious web pages are among the major security threats on the Web. Most of the existing techniques for detecting malicious web pages focus on specific attacks. Unfortunately, attacks are getting more complex whereby attackers use blended techniques to evade existing countermeasures. In this paper, we present a holistic and at the same time lightweight approach, called BINSPECT, that leverages a combination of static analysis and minimalistic emulation to apply supervised learning techniques in detecting malicious web pages pertinent to drive-by-download, phishing, injection, and malware distribution by introducing new features that can effectively discriminate malicious and benign web pages. Large scale experimental evaluation of BINSPECT achieved above 97% accuracy with low false signals. Moreover, the performance overhead of BINSPECT is in the range 3-5 seconds to analyze a single web page, suggesting the effectiveness of our approach for real-life deployment.

**Paper 5 : Using Syntactic Features for Phishing Detection**

**Gilchan Park, Julia M. Taylor- IEEE, 10(4)**

This paper reports on the comparison of the subject and object of verbs in their usage between phishing emails and legitimate emails. The purpose of this research is to explore whether the syntactic structures and subjects and objects of verbs can be distinguishable features for phishing detection. To achieve the objective, we have conducted two series of experiments: the syntactic similarity for sentences, and the subject and object of verb comparison. The results of the experiments indicated that both features can be used for some verbs, but more work has to be done for others.

**Paper 6 :Phish Score: Hacking phishers’ minds**

**S. Marchal, J. Francois, R. State, and T. Engel - IEEE, 11(9)**

Despite the growth of prevention techniques, phishing remains an important threat since the principal countermeasures in use are still based on reactive URL blacklisting. This technique is inefficient due to the short lifetime of phishing Web sites, making recent approaches relying on real-time or proactive phishing URLs detection techniques more appropriate. In this paper we introduce PhishScore, an automated real-time phishing detection system. We observed that phishing URLs usually have few relationships between the part of the URL that must be registered (upper level domain) and the remaining part of the URL (low level domain, path, query). Hence, we define this concept as intra-URL relatedness and evaluate it using features extracted from words that compose a URL based on query data from Google and Yahoo search engines. These features are then used in machine learning based classification to detect phishing URLs from a real dataset.

**Paper 7 : iTrustPage: A User-Assisted Anti-Phishing Tool**

**Troy Ronda, Stefan Saroiu, Alec Wolman- ACM, 42(4)**

Despite the many solutions proposed by industry and the research community to address phishing attacks, this problem continues to cause enormous damage. Because of our inability to deter phishing attacks, the research community needs to develop new approaches to anti-phishing solutions. Most of today’s anti-phishing technologies focus on automatically detecting and preventing phishing attacks. While automation makes anti-phishing tools user-friendly, automation also makes them suffer from false positives, false negatives, and various practical hurdles. As a result, attackers often find simple ways to escape automatic detection. This paper presents iTrust Page – an anti-phishing tool that does not rely completely on automation to detect phishing. Instead, iTrust Page relies on user input and external repositories of information to prevent users from filling out phishing Web forms. With iTrust Page, users help to decide whether or not a Web page is legitimate. Because iTrust Page is user-assisted, iTrust Page avoids the false positives and the false negatives associated with automatic phishing detection. We implemented iTrust Page as a downloadable extension to FireFox. After being featured on the Mozilla website for FireFox extensions, iTrust Page was downloaded by more than 5,000 users in a two week period. We present an analysis ofour tool’s effectiveness and ease of use based on our examination of usage logs collected from the 2,050 users who used iTrust Page for more than two weeks. Based on these logs, we find that iTrust Page disrupts users on fewer than 2% of the pages they visit, and the number of disruptions decreases over time.

**Paper 8 :Performance comparison of classifiers on reduced phishing website dataset**

**M. Karabatak and T. Mustafa- IEEE, 18(7)**

The Internet is becoming a necessary and important tool in everyday life. However, Internet users might have poor security for different kinds of web threats, which may lead to financial loss or clients lacking trust in online trading and banking. Phishing is described as a skill of impersonating a trusted website aiming to obtain private and secret information such as a username and password or social security and credit card number. In this paper, phishing website dataset taken from UCI was investigated. Its dimension was reduced and the performance comparison of classification algorithms is studied on reduced phishing website dataset. Phishing website dataset was taken from UCI machine learning repository. This dataset consists of 11055 records and 31 features. Feature selection algorithms were applied to reduce the dimension of phishing website dataset and to obtain higher classification performance. Then, the performance of classification algorithms is compared to other data mining classification algorithms. Finally, a comparative classification performance on the reduced dataset by using the common classification algorithms is given.

**Paper 9 : Phishing Detection: A Literature Survey**

**Mahmoud Khonji, Youssef Iraqi- IEEE, 13(4)**

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 believe is critical to present where the phishing detection techniques fit in the overall mitigation process.

**Paper 10 : Online Phishing Classification Using Adversarial Data Mining and Signaling Games**

**Gaston L’Huillier, Richard Weber, Nicolas Figueroa- ACM, 28(9)**

In adversarial systems, the performance of a classifier decreases after it is deployed, as the adversary learns to defeat it. Recently, adversarial data mining was introduced, where the classification problem is viewed as a game mechanism between an adversary and an intelligent and adaptive classifier .Over the last years, phishing fraud through malicious email messages has been a serious threat that affects global security and economy, where traditional spam filtering technique shave shown to be ineffective. In this domain, using dynamic games of incomplete information, a game theoretic data mining framework is proposed in order to build an adversary-aware classifier for phishing fraud detection. To build the classifier, an online version of the Weighted Margin Support Vector Machines with a game theoretic prior knowledge function is proposed. In this paper, a new content based feature extraction technique for phishing filtering is described. Experiments show that the proposed classifier is highly competitive compared with previously proposed online classification algorithms in this adversarial environment , machine learning techniques over extracted features.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

A poorly structured NN model may cause the model to underfit the training dataset . On the other hand, exaggeration in restructuring the system to suit every single item in the training dataset may cause the system to be overfitted . One possible solution to avoid the Overfitting problem is by restructuring the NN model in terms of tuning some parameters, adding new neurons to the hidden layer or sometimes adding a new layer to the network. A NN with a small number of hidden neurons may not have a satisfactory representational power to model the complexity and diversity inherent in the data. On the other hand, networks with too many hidden neurons could overfit the data. However, at a certain stage the model can no longer be improved, therefore, the structuring process should be terminated. Hence, an acceptable error rate should be specified when creating any NN model, which itself is considered a problem since it is difficult to determine the acceptable error rate a prior . For instance, the model designer may set the acceptable error rate to a value that is unreachable which causes the model to stick in local minima or sometimes the model designer may set the acceptable error rate to a value that can further be improved.

**DISADVANTAGES**

* It will take time to load all the dataset.
* Process is not accurate.
* It will analyze slowly

**3.2 PROPOSED SYSTEM**

Lexical features are based on the observation that the URLs of many illegal sites look different, compared with legitimate sites. Analyzing lexical features enables us to capture the property for classification purposes. We first distinguish the two parts of a URL: the host name and the path, from which we extract bag-of-words (strings delimited by ‘/’, ‘?’, ‘.’, ‘=’, ‘-’ and ‘’).

We find that phishing websites prefer to have longer URLs, more levels (delimited by dot), more tokens in domain and path, and longer tokens. Besides, phishing and malware websites could pretend to be a benign one by containing popular brand names as tokens other than those in second-level domain. Considering phishing websites and malware websites may use IP addresses directly so as to cover the suspicious URL, which is very rare in a benign case. Also, phishing URLs are found to contain several suggestive word tokens (confirm, account, banking, secure, ebayisapi, webscr, login, sign in), we check the presence of these security sensitive words and include the binary value in our features. Intuitively, malicious sites are always less popular than benign ones. For this reason, site popularity can be considered as an important feature. Traffic rank feature is acquired from Alexa.com. Host-based features are based on the observation that malicious sites are always registered in less reputable hosting centres or regions.

**Advantage:**

* All of the URLs in the dataset are labelled.
* We used two supervised learning algorithms random forest and support vector machine to train using scikit-learn library

**3.3 REQUIREMENT ANALYSIS AND SPECIFICATION**

**3.3.1 INPUT REQUIREMENTS**

1. **JUPYTER NOTEBOOK**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

1. **PYTHON 2.7**

It is used to compile python programs.

**3.3.2 OUTPUT REQUIREMENTS**

System with 64 bit distribution capable of running 32 bit application and 1200\*800 minimum screen resolution with stable internet connections.

**3.4 THE PYTHON PROGRAMMING LANGUAGE:**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactical constructions than other languages.

**Python is Interpreted:** Python is processed at runtime by the interpreter.You do

not need to compile your program before executing it. This is similar to PERL and PHP.

**Python is Interactive:** You can actually sit at a Python prompt and interactwith the interpreter directly to write your programs.

**Python is Object-Oriented:** Python supports Object-Oriented style ortechnique of

programming that encapsulates code within objects.

**Python is a Beginner's Language:** Python is a great language for thebeginner-level

programmers and supports the development of a wide range of applications from

simple text processing to WWW browsers to games.

**3.4.1 HISTORY OF PYTHON:**

Python laid its foundation in the late 1980s.

* The implementation of Python was started in December 1989 by Guido Van Rossum at CWI in the Netherlands.
* In February 1991, Guido Van Rossum published the code (labeled version 0.9.0) to alt.sources.
* In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce.
* Python 2.0 added new features such as list comprehensions, garbage collection systems.
* On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify the fundamental flaw of the language.
* *ABC programming language* is said to be the predecessor of Python language, which was capable of Exception Handling and interfacing with the Amoeba Operating System.
* The following programming languages influence Python:
  + ABC language.
  + Modula-3

**3.4.2 WHY THE NAME PYTHON?**

There is a fact behind choosing the name [Python](https://www.javatpoint.com/python-tutorial). Guido van Rossum was reading the script of a popular BBC comedy series "Monty Python's Flying Circus". It was late on-air 1970s.

Van Rossum wanted to select a name which unique, sort, and little-bit mysterious. So he decided to select naming Python after the "Monty Python's Flying Circus" for their newly created programming language.

The comedy series was creative and well random. It talks about everything. Thus it is slow and unpredictable, which made it very interesting.

Python is also versatile and widely used in every technical field, such as [Machine Learning](https://www.javatpoint.com/machine-learning), Artificial intelligence, Web Development, [Mobile Application](https://www.javatpoint.com/javatpoint.com/mobile-application-testing), Desktop Application, Scientific Calculation, etc.

**3.4.3 PYTHON FEATURES:**

Python's features include:

**Easy-to-learn:** Python has few keywords, simple structure, and a clearlydefined

syntax. This allows the student to pick up the language quickly.

**Easy-to-read:** Python code is more clearly defined and visible to the eyes.

**Easy-to-maintain:** Python's source code is fairly easy-to-maintain.

**A broad standard library:** Python's bulk of the library is very portable and

cross-platform compatible on UNIX, Windows, and Macintosh.

**Interactive Mode:** Python has support for an interactive mode which allows

interactive testing and debugging of snippets of code.

* **Portable:** Python can run on a wide variety of hardware platforms and has thesame interface on all platforms.
* **Extendable:** You can add low-level modules to the Python interpreter. Thesemodules enable programmers to add to or customize their tools to be more efficient.

**Databases:** Python provides interfaces to all major commercial databases.

**GUI Programming:** Python supports GUI applications that can be created andported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

* **Scalable:** Python provides a better structure and support for large programsthan shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

It supports functional and structured programming methods as well as OOP.

It can be used as a scripting language or can be compiled to byte-code for buildin large applications.

It provides very high-level dynamic data types and supports dynamic type checking.

It supports automatic garbage collection.

It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**3.5 MACHINE LEARNING**

Machine Learning is a system that can learn from example through self-improvement and without being explicitly coded by programmers. The breakthrough comes with the idea that a machine can singularly learn from the data (i.e., example) to produce accurate results.

Machine learning combines data with statistical tools to predict an output. This output is then used by corporations to make actionable insights. Machine learning is closely related to data mining and Bayesian predictive modeling. The machine receives data as input, and uses an algorithm to formulate answers.

A typical machine learning task is to provide a recommendation. For those who have a Netflix account, all recommendations of movies or series are based on the user's historical data. Tech companies are using unsupervised learning to improve the user experience with personalizing recommendations.

Machine learning is also used for a variety of tasks like fraud detection, predictive maintenance, portfolio optimization, automatizing tasks and so on.

**3.5.1 MACHINE LEARNING VS TRADITIONAL PROGRAMMING**

Traditional programming differs significantly from machine learning. In traditional programming, a programmer codes all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.



**DATA RULES**

**OUTPUT**

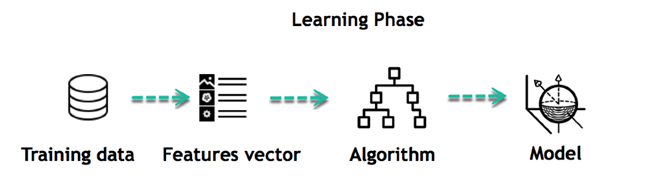
**Fig 3.5.1. ML vs TP**

## 3.5.2 HOW DOES MACHINE LEARNING WORKS?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being. Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the **learning** and **inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector.** You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.

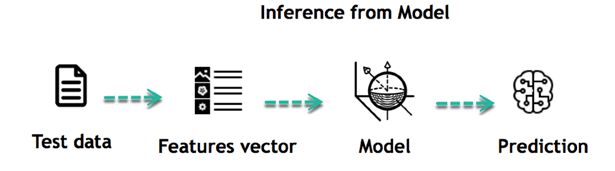


**Fig 3.5.2. Machine Learning Works**

For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

#### *Inferring*

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.



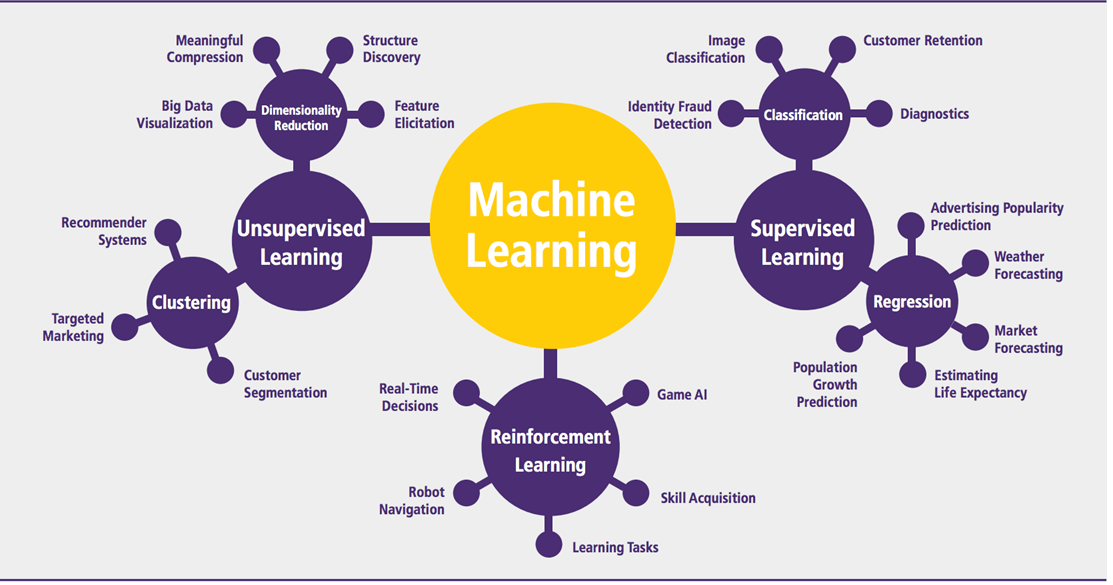
**Fig 3.5.3. Inference From Model**

The life of Machine Learning programs is straightforward and can be summarized in the following points:

1. Define a question
2. Collect data
3. Visualize data
4. Train algorithm
5. Test the Algorithm
6. Collect feedback
7. Refine the algorithm
8. Loop 4-7 until the results are satisfying
9. Use the model to make a prediction

Once the algorithm gets good at drawing the right conclusions, it applies that knowledge to new sets of data.

## 3.5.3 MACHINE LEARNING ALGORITHM AND WHERE THEY ARE USED?



Machine learning can be grouped into two broad learning tasks: Supervised and Unsupervised. There are many other algorithms

#### *Supervised learning*

An algorithm uses training data and feedback from humans to learn the relationship of given inputs to a given output. For instance, a practitioner can use marketing expense and weather forecast as input data to predict the sales of cans.

You can use supervised learning when the output data is known. The algorithm will predict new data.

There are two categories of supervised learning:

* Classification task
* Regression task

#### *Classification*

Imagine you want to predict the gender of a customer for a commercial. You will start gathering data on the height, weight, job, salary, purchasing basket, etc. from your customer database. You know the gender of each of your customer, it can only be male or female. The objective of the classifier will be to assign a probability of being a male or a female (i.e., the label) based on the information (i.e., features you have collected). When the model learned how to recognize male or female, you can use new data to make a prediction. For instance, you just got new information from an unknown customer, and you want to know if it is a male or female. If the classifier predicts male = 70%, it means the algorithm is sure at 70% that this customer is a male, and 30% it is a female.

The label can be of two or more classes. The above example has only two classes, but if a classifier needs to predict object, it has dozens of classes (e.g., glass, table, shoes, etc. each object represents a class)

#### *Regression*

When the output is a continuous value, the task is a regression. For instance, a financial analyst may need to forecast the value of a stock based on a range of feature like equity, previous stock performances, macroeconomics index. The system will be trained to estimate the price of the stocks with the lowest possible error.

| **Algorithm Name** | **Description** | **Type** |
| --- | --- | --- |
| **Linear regression** | Finds a way to correlate each feature to the output to help predict future values. | Regression |
| **Logistic regression** | Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors) | Classification |
| **Decision tree** | Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a final decision output is made | Regression Classification |
| **Naive Bayes** | The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event. | Regression Classification |
| **Support vector machine** | Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divides the classes. It is best used with a non-linear solver. | Regression (not very common) Classification |
| **Random forest** | The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most votes; while for the regression task, the average prediction of all the trees is the final prediction. | Regression Classification |
| **AdaBoost** | Classification or regression technique that uses a multitude of models to come up with a decision but weighs them based on their accuracy in predicting the outcome | Regression Classification |
| **Gradient-boosting trees** | Gradient-boosting trees is a state-of-the-art classification/regression technique. It is focusing on the error committed by the previous trees and tries to correct it. | Regression Classification |

#### *Unsupervised learning*

In unsupervised learning, an algorithm explores input data without being given an explicit output variable (e.g., explores customer demographic data to identify patterns)

You can use it when you do not know how to classify the data, and you want the algorithm to find patterns and classify the data for you

| **Algorithm** | **Description** | **Type** |
| --- | --- | --- |
| **K-means clustering** | Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans) | Clustering |
| **Gaussian mixture model** | A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters | Clustering |
| **Hierarchical clustering** | Splits clusters along a hierarchical tree to form a classification system.  Can be used for Cluster loyalty-card customer | Clustering |
| **Recommender system** | Help to define the relevant data for making a recommendation. | Clustering |
| **PCA/T-SNE** | Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances. | Dimension Reduction |

**3.5.4 APPLICATION OF MACHINE LEARNING**

**Augmentation**:

* Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

**Automation**:

* Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

**Finance Industry**

* Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

**Government organization**

* The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalkers.

**Healthcare industry**

* Healthcare was one of the first industry to use machine learning with image detection.

**Marketing**

* Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers developed advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, the marketing department relies on AI to optimize the customer relationship and marketing campaign.

**3.5.5 EXAMPLE OF APPLICATION OF MACHINE LEARNING IN SUPPLY CHAIN**

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network.

Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time.

In the past year stock managers relied extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In terms of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

**Example of Machine Learning Google Car**

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

**3.5.6 DEEP LEARNING**

Deep learning is a computer software that mimics the network of neurons in the brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. The machine uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. Deep learning is the new state of the art in terms of AI. In deep learning, the learning phase is done through a neural network.

**Reinforcement Learning**

Reinforcement learningis a subfield of machine learning in which systems are trained by receiving virtual "rewards" or "punishments," essentially learning by trial and error. Google's DeepMind has used reinforcement learning to beat a human champion in the Go games. Reinforcement learning is also used in video games to improve the gaming experience by providing smarter bot.

One of the most famous algorithms are:

* Q-learning
* Deep Q network
* State-Action-Reward-State-Action (SARSA)
* Deep Deterministic Policy Gradient (DDPG)

**Applications/ Examples of deep learning applications**

**AI in Finance:**The financial technology sector has already started using AI to save time, reduce costs, and add value. Deep learning is changing the lending industry by using more robust credit scoring. Credit decision-makers can use AI for robust credit lending applications to achieve faster, more accurate risk assessment, using machine intelligence to factor in the character and capacity of applicants.

Underwrite is a Fintech company providing an AI solution for credit makers company. underwrite.ai uses AI to detect which applicant is more likely to pay back a loan. Their approach radically outperforms traditional methods.

**AI in HR:**Under Armour, a sportswear company revolutionizes hiring and modernizes the candidate experience with the help of AI. In fact, Under Armour Reduces hiring time for its retail stores by 35%. Under Armour faced a growing popularity interest back in 2012. They had, on average, 30000 resumes a month. Reading all of those applications and begin to start the screening and interview process was taking too long. The lengthy process to get people hired and on-boarded impacted Under Armour's ability to have their retail stores fully staffed, ramped and ready to operate.

At that time, Under Armour had all of the 'must have' HR technology in place such as transactional solutions for sourcing, applying, tracking and onboarding but those tools weren't useful enough. Under armour choose **HireVue**, an AI provider for HR solution, for both on-demand and live interviews. The results were bluffing; they managed to decrease by 35% the time to fill. In return, they hired higher quality staff.

**AI in Marketing:**AI is a valuable tool for customer service management and personalization challenges. Improved speech recognition in call-center management and call routing as a result of the application of AI techniques allows a more seamless experience for customers.

For example, deep-learning analysis of audio allows systems to assess a customer's emotional tone. If the customer is responding poorly to the AI chatbot, the system can reroute the conversation to real, human operators that take over the issue.

Apart from the three examples above, AI is widely used in other sectors/industries.

**Artificial Intelligence**



## 3.5.7 DIFFERENCE BETWEEN MACHINE LEARNING AND DEEP LEARNING

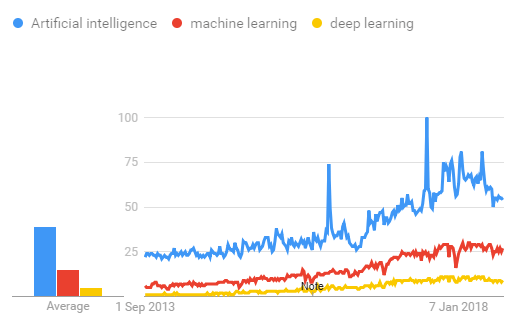
|  | **Machine Learning** | **Deep Learning** |
| --- | --- | --- |
| **Data Dependencies** | Excellent performances on a small/medium dataset | Excellent performance on a big dataset |
| **Hardware dependencies** | Work on a low-end machine. | Requires powerful machine, preferably with GPU: DL performs a significant amount of matrix multiplication |
| **Feature engineering** | Need to understand the features that represent the data | No need to understand the best feature that represents the data |
| **Execution time** | From few minutes to hours | Up to weeks. Neural Network needs to compute a significant number of weights |
| **Interpretability** | Some algorithms are easy to interpret (logistic, decision tree), some are almost impossible (SVM, XGBoost) | Difficult to impossible |

## 3.5.8 WHEN TO USE ML OR DL?

In the table below, we summarize the difference between machine learning and deep learning.

|  | **Machine learning** | **Deep learning** |
| --- | --- | --- |
| **Training dataset** | Small | Large |
| **Choose features** | Yes | No |
| **Number of algorithms** | Many | Few |
| **Training time** | Short | Long |

With machine learning, you need fewer data to train the algorithm than deep learning. Deep learning requires an extensive and diverse set of data to identify the underlying structure. Besides, machine learning provides a faster-trained model. Most advanced deep learning architecture can take days to a week to train. The advantage of deep learning over machine learning is it is highly accurate. You do not need to understand what features are the best representation of the data; the neural network learned how to select critical features. In machine learning, you need to choose for yourself what features to include in the model.



**Fig 3.5.8. Comparison Graph**

**3.6 ALGORITHM**

**RANDOM FOREST**

Random forest is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where you join different types of algorithms or same algorithm multiple times to form a more powerful prediction model. The random forest algorithm combines multiple algorithm of the same type i.e. multiple decision *trees*, resulting in a *forest of trees*, hence the name "Random Forest". The random forest algorithm can be used for both regression and classification tasks.

**HOW RANDOM FOREST WORKS**

The following are the basic steps involved in performing the random forest algorithm

1. Pick N random records from the dataset.
2. Build a decision tree based on these N records.
3. Choose the number of trees you want in your algorithm and repeat steps 1 and 2.
4. For classification problems, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote.

**ADVANTAGES OF USING RANDOM FOREST**

pros of using random forest for classification and regression.

1. The random forest algorithm is not biased, since there are multiple trees and each tree is trained on a subset of data. Basically, the random forest algorithm relies on the power of "the crowd"; therefore, the overall biasedness of the algorithm is reduced.
2. This algorithm is very stable. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, but it is very hard for it to impact all the trees.
3. The random forest algorithm works well when you have both categorical and numerical features.
4. The random forest algorithm also works well when data has missing values or it has not been scaled.

**3.7 TECHNOLOGY STACK**

**3.7.1 SOFTWARE REQUIREMENTS**

* Windows 7 or higher
* Python 2.7
* Anaconda Navigator
* Python’s standard library
* Pandas
* Numpy
* Sklearn
* tkMessageBox
* Dataset of Phishing websites

**3.7.2 HARDWARE REQUIREMENTS**

* Intel 1.66 Ghz processor pentium 4
* RAM 4GB(minimum)
* 100GB HDD
* Internet Connectivity

**ANACONDA NAVIGATOR**

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, Mac OS and Linux.

## Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

## What applications can I access using navigator ?

The following applications are available by default in Navigator:

* JupyterLab
* Jupyter Notebook
* QTConsole
* Spyder
* VSCode
* Glueviz
* Orange 3 App
* Rodeo
* RStudio

Advanced conda users can also build your own Navigator applications

## How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

**What’s new in 1.9?**

* Add support for **Offline Mode** for all environment related actions.
* Add support for custom configuration of main windows links.

**NUMPY**

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

* NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
* The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
* NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
* A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

**SKLEARN**

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy.Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

**PANDAS**

Pandas is an open-source library that is built on top of NumPy library. It is a Python package that offers various data structures and operations for manipulating numerical data and time series. It is mainly popular for importing and analyzing data much easier. Pandas is fast and it has high-performance & productivity for users.

**PYTHON STANDARD LIBRARY**

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

**TK MESSAGEBOX**

The tkMessageBox module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.

Some of these functions are show info, show warning, show error, ask question, ask ok cancel, ask yes no, and ask retry ignore.

**CHAPTER 4**

**SYSTEM DESIGN**

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 DATA FLOW DIAGRAMS**

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

**LEVEL 0**















**LEVEL 1**















**LEVEL 2**















**Fig 4.1.1. Data Flow diagram**

**4.2 UML DIAGRAM**

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.

**4.2.1 USE CASE DIAGRAM**

A use case is a set of scenarios that describe an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. A Use case Diagram is used to present a graphical overview of the functionality provided by a system in terms of actors, their goals and any dependencies between those use cases.

**Use case diagram consists of two parts:**

**Use case:** A use case describes a sequence of actions that provided something of

measurable value to an actor and is drawn as a horizontal ellipse.

**Actor:** An actor is a person, organization or external system that plays a role in one or

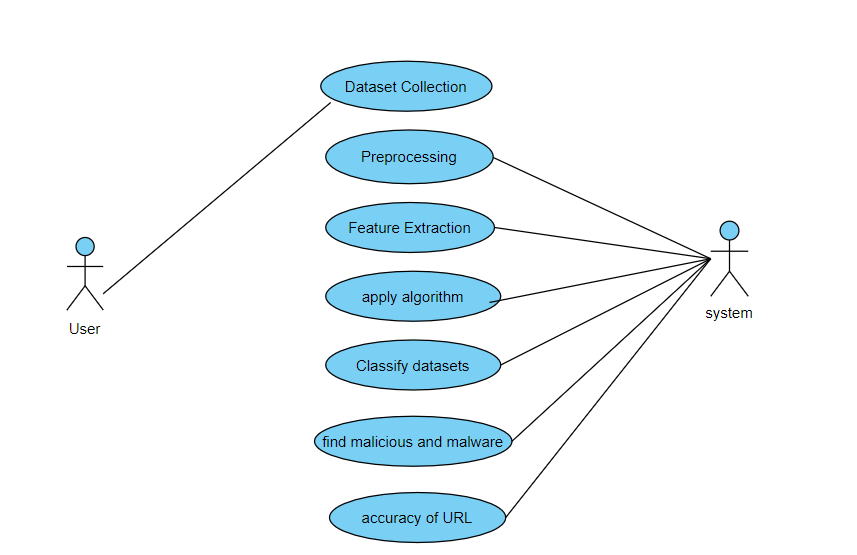
more interaction with the system.

**Communication Link:** The participation of an actor in a use case is shown by

connecting an actor to a use case by a solid link. Actors may be connected to use

cases by associations, indicating that the actor and the use case communicate with

one another using message.

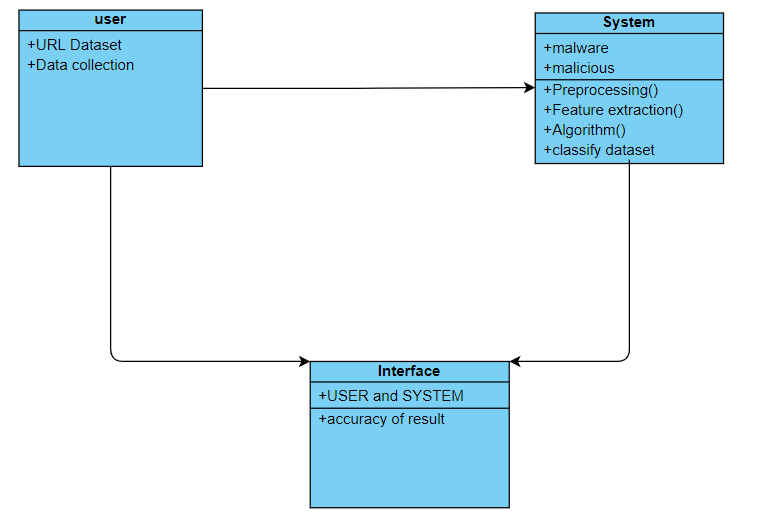
****

**Fig 4.2.1. Use case diagram**

**4.2.2 CLASS DIAGRAM**

A Class diagram in the Unified Modeling Language is a type of static structure

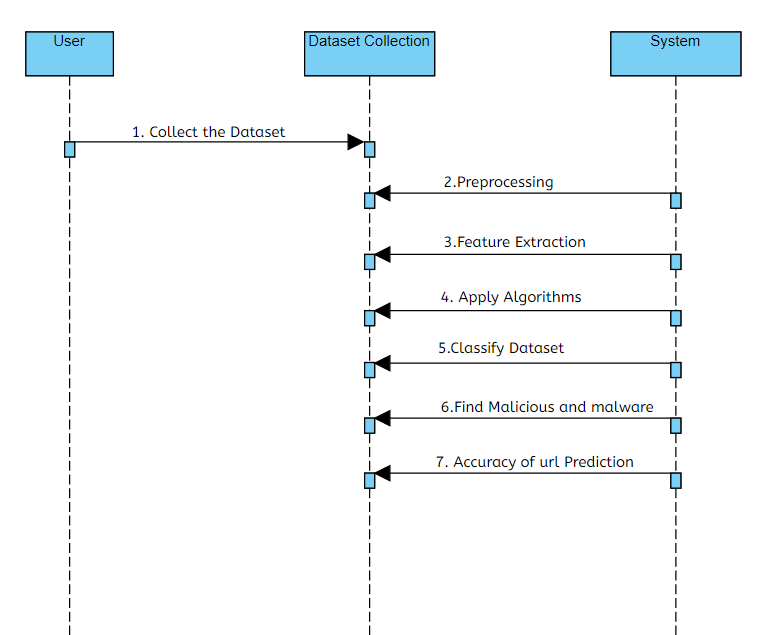
diagram that describes the structure of a system by showing the system&#39;s classes, their attributes, operations (or methods), and the relationships among objects.



**Fig 4.2.2. Class diagram**

**4.2.3 SEQUENCE DIAGRAM**

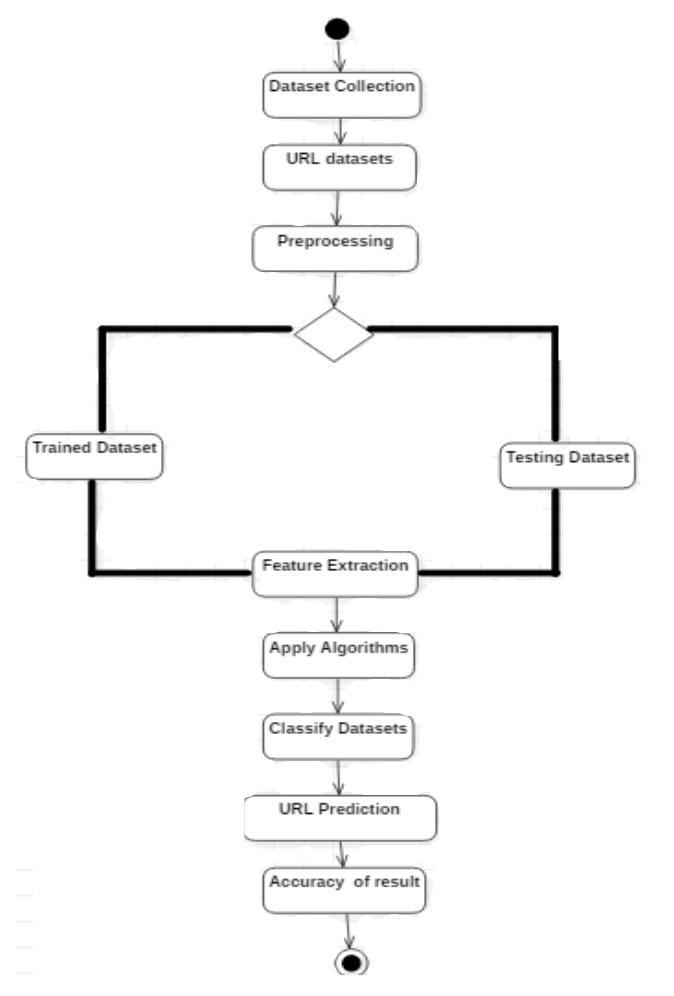
Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur.

****

**Fig 4.2.3. Sequence diagram**

**4.2.4 ACTIVITY DIAGRAM**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another.



**Fig 4.2.4. Activity diagram**

**CHAPTER 5**

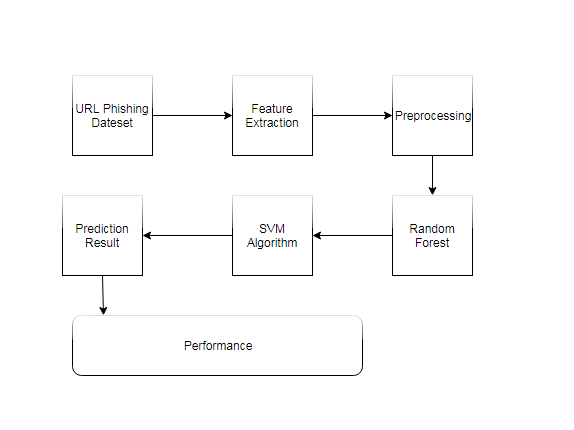
**SYSTEM ARCHITECTURE**

**CHAPTER 5**

**SYSTEM ARCHITECTURE**

**5.1 ARCHITECTURE OVERVIEW**

First ,we need to collect the URL Phishing dataset from kaggle and feature extraction is done for that dataset where it classifies the website as Phishing or not.Then, preprocessing is done with Random Forest and SVM Algorithm. From this, result will be predicted and performance is done

****

**Fig 5.1.1 Architecture Diagram**

**5.2 MODULE DESCRIPTION**

**5.2.1. PHISHING WEBSITE FEATURES**

One of the challenges faced by our research was the unavailability of reliable training datasets. In fact, this challenge faces any researcher in the field. However, although plenty of articles about predicting phishing websites using data mining techniques have been disseminated these days, no reliable training dataset has been published publicly, maybe because there is no agreement in literature on the definitive features that characterize phishing websites, hence it is difficult to shape a dataset that covers all possible features.

In this article, we shed light on the important features that have proved to be sound and effective in predicting phishing websites. In addition, we proposed some new features, experimentally assigned new rules to some well-known features and updated some other features.

**Address Bar based Features**

1. Using the IP Address

If an IP address is used as an alternative to the domain name in the URL, such as “http://125.98.3.123/fake.html”, users can be sure that someone is trying to steal their personal information. Sometimes, the IP address is even transformed into hexadecimal code as shown in the following link “[http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html](about:blank)”

Rule: IF{ If The Domain Part has an IP Address → Phishing

Otherwise → Legitimate

2)Long URL to Hide the Suspicious Part

Rule: IF{ 𝑈𝑅𝐿 𝑙𝑒𝑛𝑔𝑡ℎ < 54 → 𝑓𝑒𝑎𝑡𝑢𝑟𝑒 = Legitimate

𝑒𝑙𝑠𝑒 𝑖𝑓 𝑈𝑅𝐿 𝑙𝑒𝑛𝑔𝑡ℎ ≥ 54 𝑎𝑛𝑑 ≤ 75 → 𝑓𝑒𝑎𝑡𝑢𝑟𝑒 = 𝑆𝑢𝑠𝑝𝑖𝑐𝑖𝑜𝑢𝑠

𝑜𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒 → 𝑓𝑒𝑎𝑡𝑢𝑟𝑒 = Phishing

3)Using URL Shortening Services “TinyURL”

Rule: IF{ TinyURL → Phishing

Otherwise → Legitimate

4) URL’s having “@” Symbol

Using “@” symbol in the URL leads the browser to ignore everything preceding the “@” symbol and the real address often follows the “@” symbol.

Rule: IF { Url Having @ Symbol → Phishing

Otherwise → Legitimate

5)Redirecting using “//”

The existence of “//” within the URL path means that the user will be redirected to another website. An example of such URL’s is: “http://www.legitimate.com//http://www.phishing.com”. We examine the location where the “//” appears. We find that if the URL starts with “HTTP”, that means the “//” should appear in the sixth position. However, if the URL employs “HTTPS” then the “//” should appear in seventh position.

Rule: IF { The Position of the Last Occurrence of "//" in the URL > 7 → Phishing

Otherwise → Legitimate

6)Adding Prefix or Suffix Separated by (-) to the Domain

For example http://www.Confirme-paypal.com/.

Rule: IF { Domain Name Part Includes (−) Symbol → Phishing

Otherwise → Legitimate

7)Domain Registration Length

Based on the fact that a phishing website lives for a short period of time, we believe that trustworthy domains are regularly paid for several years in advance. In our dataset, we find that the longest fraudulent domains have been used for one year only.

Rule: IF{ Domains Expires on ≤ 1 years → Phishing

Otherwise → Legitimate

**5.2.2. USING POP-UP WINDOW**

It is unusual to find a legitimate website asking users to submit their personal information through a pop-up window. On the other hand, this feature has been used in some legitimate websites and its main goal is to warn users about fraudulent activities or broadcast a welcome announcement, though no personal information was asked to be filled in through these pop-up windows.

Rule: IF Popup Window Contains Text Fields→ Phishing

Otherwise → Legitimate

**5.2.3. CLASSIFICATION**

To ensure that our approach works well irrespective of the underlying classiﬁer chosen for the task, we performed the experiments using two different classiﬁers: Random Forest and Support vector machine, as these are some of the most commonly used classiﬁers for the task of text-data classiﬁcation. Scikit-learn implementation of these classiﬁers with their default parameter settings are used for our experiments. The tf-idf feature is used to represent each URL in the database.

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**6.1 CODING**

**feature extraction.py**

from urlparse import urlparse

#from urllib.parse import urlparse

import re

import urllib2

#import urllib.request

import urllib

from xml.dom import minidom

import csv

import pygeoip

#from urllib.request import urlopen#shayd karana pde ye

opener = urllib2.build\_opener()

#opener = urllib.request.build\_opener()

opener.addheaders = [('User-agent', 'Mozilla/5.0')]

nf=-1

def Tokenise(url):

if url=='':

return [0,0,0]

token\_word=re.split('\W+',url)

#print token\_word

no\_ele=sum\_len=largest=0

for ele in token\_word:

l=len(ele)

sum\_len+=l

if l>0: ## for empty element exclusion in average length

no\_ele+=1

if largest<l:

largest=l

try:

return [float(sum\_len)/no\_ele,no\_ele,largest]

except:

return [0,no\_ele,largest]

def find\_ele\_with\_attribute(dom,ele,attribute): # used in sitepopularity fxn

for subelement in dom.getElementsByTagName(ele):

if subelement.hasAttribute(attribute):

return subelement.attributes[attribute].value

return nf

def sitepopularity(host):

xmlpath='http://data.alexa.com/data?cli=10&dat=snbamz&url='+host

#print xmlpath

try:

xml= urllib2.urlopen(xmlpath)

#xml= urllib.request.urlopen(xmlpath)

dom =minidom.parse(xml)

rank\_host=find\_ele\_with\_attribute(dom,'REACH','RANK')

#country=find\_ele\_with\_attribute(dom,'REACH','RANK')

rank\_country=find\_ele\_with\_attribute(dom,'COUNTRY','RANK')

return [rank\_host,rank\_country]

except:

return [nf,nf]

def Security\_sensitive(tokens\_words):

sec\_sen\_words=['confirm', 'account', 'banking', 'secure', 'ebayisapi', 'webscr', 'login', 'signin']

cnt=0

for ele in sec\_sen\_words:

if(ele in tokens\_words):

cnt+=1;

return cnt

def exe\_in\_url(url):

if url.find('.exe')!=-1:

return 1

return 0

def Check\_IPaddress(tokens\_words):

cnt=0;

for ele in tokens\_words:

if unicode(ele).isnumeric():

cnt+=1

else:

if cnt>=4 :

return 1

else:

cnt=0;

if cnt>=4:

return 1

return 0

def getASN(host):

try:

g = pygeoip.GeoIP('GeoIPASNum.dat')

asn=int(g.org\_by\_name(host).split()[0][2:])

return asn

except:

return nf

def web\_content\_features(url):

wfeatures={}

total\_cnt=0

try:

source\_code = str(opener.open(url))

#print source\_code[:500]

wfeatures['src\_html\_cnt']=source\_code.count('<html')

wfeatures['src\_hlink\_cnt']=source\_code.count('<a href=')

wfeatures['src\_iframe\_cnt']=source\_code.count('<iframe')

#suspicioussrc\_ javascript functions count

wfeatures['src\_eval\_cnt']=source\_code.count('eval(')

wfeatures['src\_escape\_cnt']=source\_code.count('escape(')

wfeatures['src\_link\_cnt']=source\_code.count('link(')

wfeatures['src\_underescape\_cnt']=source\_code.count('underescape(')

wfeatures['src\_exec\_cnt']=source\_code.count('exec(')

wfeatures['src\_search\_cnt']=source\_code.count('search(')

for key in wfeatures:

if(key!='src\_html\_cnt' and key!='src\_hlink\_cnt' and key!='src\_iframe\_cnt'):

total\_cnt+=wfeatures[key]

wfeatures['src\_total\_jfun\_cnt']=total\_cnt

except Exception ,e:

print ("Error"+str(e)+" in downloading page "+url)

default\_val=nf

wfeatures['src\_html\_cnt']=default\_val

wfeatures['src\_hlink\_cnt']=default\_val

wfeatures['src\_iframe\_cnt']=default\_val

wfeatures['src\_eval\_cnt']=default\_val

wfeatures['src\_escape\_cnt']=default\_val

wfeatures['src\_link\_cnt']=default\_val

wfeatures['src\_underescape\_cnt']=default\_val

wfeatures['src\_exec\_cnt']=default\_val

wfeatures['src\_search\_cnt']=default\_val

wfeatures['src\_total\_jfun\_cnt']=default\_val

return wfeatures

def safebrowsing(url):

api\_key = "ABQIAAAA8C6Tfr7tocAe04vXo5uYqRTEYoRzLFR0-nQ3fRl5qJUqcubbrw"

name = "URL\_check"

ver = "1.0"

req = {}

req["client"] = name

req["apikey"] = api\_key

req["appver"] = ver

req["pver"] = "3.0"

req["url"] = url #change to check type of url

try:

params = urllib.urlencode(req)

req\_url = "https://sb-ssl.google.com/safebrowsing/api/lookup?"+params

res = urllib.request.urlopen(req\_url)

#res = urllib.request.urlopen(req\_url)

# print res.code

# print res.read()

if res.code==204:

# print "safe"

return 0

elif res.code==200:

# print "The queried URL is either phishing, malware or both, see the response body for the specific type."

return 1

elif res.code==204:

print ("The requested URL is legitimate, no response body returned.")

elif res.code==400:

print ("Bad Request The HTTP request was not correctly formed.")

elif res.code==401:

print ("Not Authorized The apikey is not authorized")

else:

print ("Service Unavailable The server cannot handle the request. Besides the normal server failures, it could also indicate that the client has been throttled by sending too many requests")

except:

return -1

def feature\_extract(url\_input):

Feature={}

tokens\_words=re.split('\W+',url\_input) #Extract bag of words stings delimited by (.,/,?,,=,-,\_)

#print tokens\_words,len(tokens\_words)

#token\_delimit1=re.split('[./?=-\_]',url\_input)

#print token\_delimit1,len(token\_delimit1)

obj=urlparse(url\_input)

host=obj.netloc

path=obj.path

Feature['URL']=url\_input

Feature['rank\_host'],Feature['rank\_country'] =sitepopularity(host)

Feature['host']=obj.netloc

Feature['path']=obj.path

Feature['Length\_of\_url']=len(url\_input)

Feature['Length\_of\_host']=len(host)

Feature['No\_of\_dots']=url\_input.count('.')

Feature['avg\_token\_length'],Feature['token\_count'],Feature['largest\_token'] = Tokenise(url\_input)

Feature['avg\_domain\_token\_length'],Feature['domain\_token\_count'],Feature['largest\_domain'] = Tokenise(host)

Feature['avg\_path\_token'],Feature['path\_token\_count'],Feature['largest\_path'] = Tokenise(path)

Feature['sec\_sen\_word\_cnt'] = Security\_sensitive(tokens\_words)

Feature['IPaddress\_presence'] = Check\_IPaddress(tokens\_words)

# print host

# print getASN(host)

# Feature['exe\_in\_url']=exe\_in\_url(url\_input)

Feature['ASNno']=getASN(host)

Feature['safebrowsing']=safebrowsing(url\_input)

"""wfeatures=web\_content\_features(url\_input)

for key in wfeatures:

Feature[key]=wfeatures[key]

"""

#debug

# for key in Feature:

#print key +':'+str(Feature[key])

return Feature

**gui.py**

from Tkinter import \*

import tkMessageBox

import trainer as tr

import pandas

import main

from PIL import ImageTk, Image

import os

root = Tk()

root.geometry('1100x600+500+800')

root.configure(background = "#001a4d")

root.attributes("-fullscreen", True)

frame = Frame(root)

frame.pack()

bottomframe = Frame(root)

bottomframe.pack(side = BOTTOM)

im = Image.open('image.png').resize((1100,500))#width,height

#size= width,height = im.size

#im.resize((5000,128))

img = ImageTk.PhotoImage(im)

panel = Label(root, image = img)

#panel.pack(side = "bottom", fill = "both", expand = "yes")

panel.pack()

L1 = Label(frame, text="Enter the URL: ",fg="MidnightBlue",font = 'times 17 bold underline')# for text enter the url

L1.pack( side = LEFT)

E1 = Entry(frame,bd =35, width=180,fg="#001a4d" ,bg="AliceBlue")# for text box

#E1.insert(0, 'Enter your URL')

E1.pack(side = RIGHT)

def submitCallBack():

url=E1.get()

main.process\_test\_url(url,'gui\_url\_features.csv')

return\_ans = tr.gui\_caller('url\_features.csv','gui\_url\_features.csv')

a=str(return\_ans).split()

if int(a[1])==0:

tkMessageBox.showinfo( "URL Checker Result","The URL "+url+" is Benign")

elif int(a[1])==1:

tkMessageBox.showinfo( "URL Checker Result","The URL "+url+" is Malicious")

else:

tkMessageBox.showinfo( "URL Checker Result","The URL "+url+" is Malware")

B1 = Button(bottomframe, text ="Submit", command = submitCallBack,bg="LightSeaGreen",height=3,width=10)

B1.pack()

root.mainloop()

**main.py**

import csv

import Feature\_extraction as urlfeature # this will take file feature\_extraction as urlfeature

import trainer as tr # this will take file train.py as tr

def resultwriter(feature,output\_dest): # this will write all the features iin a csv file

flag=True

with open(output\_dest,'wb') as f:

for item in feature:

w = csv.DictWriter(f, item[1].keys())

if flag:

w.writeheader()

flag=False

w.writerow(item[1])

def process\_URL\_list(file\_dest,output\_dest):# i think this takes whole file of urls with given malicious to extract their feature and provide malicious column also like this will take url.txt

feature=[]

with open(file\_dest) as file:

for line in file:

url=line.split(',')[0].strip()

malicious\_bool=line.split(',')[1].strip()

if url!='':

print ('working on: '+url) #showoff

ret\_dict=urlfeature.feature\_extract(url)

ret\_dict['malicious']=malicious\_bool

feature.append([url,ret\_dict]);

resultwriter(feature,output\_dest)

def process\_test\_list(file\_dest,output\_dest): # i think this takes whole file of urls without given malicious to extract their feature and doest not provide malicious column like this will take query.txt

feature=[]

with open(file\_dest) as file:

for line in file:

url=line.strip()

if url!='':

print ('working on: '+url) #showoff

ret\_dict=urlfeature.feature\_extract(url)

feature.append([url,ret\_dict]);

resultwriter(feature,output\_dest)

#change

def process\_test\_url(url,output\_dest): # i think this takes a single url to extract feature, this is used in gui.py file only

feature=[]

url=url.strip()

if url!='':

print ('working on: '+url) #showoff

ret\_dict=urlfeature.feature\_extract(url)

feature.append([url,ret\_dict]);

resultwriter(feature,output\_dest)

def main():# i think 1,2,4 lines are appropriate to for creating extracted features file of train and test data then apply model on them

#process\_URL\_list('URL.txt','url\_features.csv')

#process\_test\_list("query.txt",'query\_features.csv')

#tr.train('url\_features.csv','url\_features.csv') #arguments:(input\_training feature,test/query training features)

tr.train('train\_features.csv','test\_features.csv') #testing with urls in query.txt

**trainer.py**

import pandas

from sklearn import preprocessing

from sklearn.ensemble import RandomForestClassifier

import numpy

from sklearn import svm

from sklearn.model\_selection import cross\_validate

import matplotlib.pylab as plt

import warnings

from sklearn.ensemble import BaggingClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

warnings.filterwarnings("ignore", category=DeprecationWarning,

module="pandas", lineno=570)

from sklearn.ensemble import GradientBoostingClassifier

#from xgboost import XGBClassifier

def return\_nonstring\_col(data\_cols): # giving columns that are not string in nature like url , host, path

cols\_to\_keep=[]

train\_cols=[]

for col in data\_cols:

if col!='URL' and col!='host' and col!='path':

cols\_to\_keep.append(col)

if col!='malicious' and col!='result':

train\_cols.append(col)

return [cols\_to\_keep,train\_cols]

def svm\_classifier(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

clf = svm.SVC()

print (clf.fit(train[train\_cols], train['malicious']))

scores = cv.cross\_val\_score(clf, train[train\_cols], train['malicious'], cv=30)

print('Estimated score SVM: %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

query['result']=clf.predict(query[train\_cols])

print (query[['URL','result']])

query[['URL','result']].to\_csv("C:/Users/hp/phishing/test\_predicted\_target\_svm.csv")

# Called from gui

def forest\_classifier\_gui(train,query,train\_cols):# train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

rf = RandomForestClassifier(n\_estimators=150)

print (rf.fit(train[train\_cols], train['malicious']))

query['result']=rf.predict(query[train\_cols])

print (query[['URL','result']].head(2))

return query['result']

def svm\_classifier\_gui(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

clf = svm.SVC()

train[train\_cols] = preprocessing.scale(train[train\_cols])

query[train\_cols] = preprocessing.scale(query[train\_cols])

print (clf.fit(train[train\_cols], train['malicious']))

query['result']=clf.predict(query[train\_cols])

print (query[['URL','result']].head(2))

return query['result']

def GradientBoosting\_Classifier\_gui(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

grad = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1.0, max\_depth=1, random\_state=0)

print (grad.fit(train[train\_cols], train['malicious']))

query['result']=grad.predict(query[train\_cols])

print (query[['URL','result']].head(2))

return query['result']

def forest\_classifier(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

rf = RandomForestClassifier(n\_estimators=150)

print (rf.fit(train[train\_cols], train['malicious']))

scores = cv.cross\_val\_score(rf, train[train\_cols], train['malicious'], cv=30)

print('Estimated score RandomForestClassifier: %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

query['result']=rf.predict(query[train\_cols])

print (query[['URL','result']])

query[['URL','result']].to\_csv("C:/Users/hp/phishing/test\_predicted\_target\_rf.csv")

def Bagging\_Classifier(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

bag = BaggingClassifier(n\_estimators=150)

print (bag.fit(train[train\_cols], train['malicious']))

scores = cv.cross\_val\_score(bag, train[train\_cols], train['malicious'], cv=30)

print('Estimated score BaggingClassifier : %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

query['result']=bag.predict(query[train\_cols])

print (query[['URL','result']])

query[['URL','result']].to\_csv("C:/Users/hp/phishing/test\_predicted\_target\_bag.csv")

def logistic\_regression(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

logis = LogisticRegression()

print (logis.fit(train[train\_cols], train['malicious']))

scores = cv.cross\_val\_score(logis, train[train\_cols], train['malicious'], cv=30)

print('Estimated score logisticregression : %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

query['result']=logis.predict(query[train\_cols])

print (query[['URL','result']])

query[['URL','result']].to\_csv("C:/Users/hp/phishing/test\_predicted\_target\_logis.csv")

def DecisionTree\_Classifier(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

deci = DecisionTreeClassifier(random\_state = 100,max\_depth=3, min\_samples\_leaf=5)

print (deci.fit(train[train\_cols], train['malicious']))

scores = cv.cross\_val\_score(deci, train[train\_cols], train['malicious'], cv=30)

print('Estimated score decisiontreeclassifier : %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

query['result']=deci.predict(query[train\_cols])

print (query[['URL','result']])

query[['URL','result']].to\_csv("C:/Users/hp/phishing/test\_predicted\_target\_deci.csv")

def KNeighbors\_Classifier(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

Kneigh = KNeighborsClassifier()

print (Kneigh.fit(train[train\_cols], train['malicious']))

scores = cv.cross\_val\_score(Kneigh, train[train\_cols], train['malicious'], cv=30)

print('Estimated score KNeighborsClassifier : %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

query['result']=Kneigh.predict(query[train\_cols])

print (query[['URL','result']])

query[['URL','result']].to\_csv("C:/Users/hp/phishing/test\_predicted\_target\_Kneigh.csv")

def GradientBoosting\_Classifier(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

grad = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1.0, max\_depth=1, random\_state=0)

print (grad.fit(train[train\_cols], train['malicious']))

scores = cv.cross\_val\_score(grad, train[train\_cols], train['malicious'], cv=30)

print('Estimated score GradientBoostingClassifier : %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

query['result']=grad.predict(query[train\_cols])

print (query[['URL','result']])

query[['URL','result']].to\_csv("C:/Users/hp/phishing/test\_predicted\_target\_grad.csv")

def Xg\_boost(train,query,train\_cols): # train is train dataset and query is test dataset and train\_cols is are the columns of train dataset exclude malicious

xg = XGBClassifier()

print (xg.fit(train[train\_cols], train['malicious']))

scores = cv.cross\_val\_score(xg, train[train\_cols], train['malicious'], cv=30)

print('Estimated score Xgboost : %0.5f (+/- %0.5f)' % (scores.mean(), scores.std() / 2))

query['result']=xg.predict(query[train\_cols])

print (query[['URL','result']])

query[['URL','result']].to\_csv("C:/Users/hp/phishing/test\_predicted\_target\_xg.csv")

def train(db,test\_db):

query\_csv = pandas.read\_csv(test\_db)

cols\_to\_keep,train\_cols=return\_nonstring\_col(query\_csv.columns)

#query=query\_csv[cols\_to\_keep]

train\_csv = pandas.read\_csv(db)

cols\_to\_keep,train\_cols=return\_nonstring\_col(train\_csv.columns)

train=train\_csv[cols\_to\_keep]

#svm\_classifier(train\_csv,query\_csv,train\_cols)

#forest\_classifier(train\_csv,query\_csv,train\_cols)

**CHAPTER 7**

**SYSTEM TESTING**

**CHAPTER 7**

**SYSTEM TESTING**

**7.1 SYSTEM TESTING**

Testing is the process of executing a program with the aim of finding errors. To make our software perform well it should be error-free. If testing is done successfully it will remove all the errors from the software.

#### 7.1.1 Unit Testing

It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by using sample input and observing its corresponding outputs.

#### 7.1.2 Integration Testing

#### The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is combined to produce output.

Integration testing is of four types: (i) Top-down (ii) Bottom-up (iii) Sandwich (iv) Big-Bang

Example

**(a)** Black Box testing:- It is used for validation.

In this we ignore internal working mechanism and

focuse on what is the output?.

**(b)** White Box testing:- It is used for verification.

In this we focus on internal mechanism i.e.

how the output is achieved?

**7.1.3 Regression Testing**

Every time a new module is added leads to changes in the program. This type of testing makes sure that the whole component works properly even after adding components to the complete program.

#### 7.1.4 Smoke Testing

This test is done to make sure that software under testing is ready or stable for further testing.It is called a smoke test as the testing an initial pass is done to check if it did not catch the fire or smoke in the initial switch on.

#### 7.1.5 Alpha Testing

This is a type of validation testing. It is a type of *acceptance testing* which is done before the product is released to customers. It is typically done by QA people.

**7.1.6 Beta Testing**

The beta test is conducted at one or more customer sites by the end-user of the software. This version is released for a limited number of users for testing in a real-time environment

**7.1.7 System Testing**

This software is tested such that it works fine for the different operating systems. It is covered under the black box testing technique. In this, we just focus on the required input and output without focusing on internal working.

In this, we have security testing, recovery testing, stress testing, and performance testing

**7.1.8 Stress Testing**

In this, we give unfavorable conditions to the system and check how they perform in those conditions.

#### 7.1.9 Performance Testing

It is designed to test the run-time performance of software within the context of an integrated system. It is used to test the speed and effectiveness of the program. It is also called load testing. In it we check, what is the performance of the system in the given load.

#### 7.1.10 Object-Oriented Testing

This testing is a combination of various testing techniques that help to verify and validate object-oriented software. This testing is done in the following manner:

* Testing of Requirements,
* Design and Analysis of Testing,
* Testing of Code,
* Integration testing,
* System testing,
* User Testing.

**7.2 TEST CASES AND REPORT**

| **S.NO** | **INPUT** | **EXPECTED OUTPUT** | **ACTUAL OUTPUT** | **STATUS** |
| --- | --- | --- | --- | --- |
| **1** | http://hdfcbank.com | The URL <http://hdfcbank.com>  is benign | The URL <http://hdfcbank.com>  is benign | PASSED |
| **2** | http://stackoverflow.com/questions/8551735/how-do-i-run-python-code-from-sublime-text-2 | The URL <http://stackoverflow.com/questions/8551735/how-do-i-run-python-code-from-sublime-text-2> is benign | The URL <http://stackoverflow.com/questions/8551735/how-do-i-run-python-code-from-sublime-text-2> is benign | PASSED |
| **3** | http://j-heaven.tripod.com/library.htm | TheURL <http://j-heaven.tripod.com/library.htm> is Malware | TheURL <http://j-heaven.tripod.com/library.htm> is Malware | PASSED |
| **4** | http://jam.canoe.ca/movies/artists/g/gottfried\_gilbert/1997/02/21/758674.html | The URL <http://jam.canoe.ca/movies/artists/g/gottfried_gilbert/1997/02/21/758674.html>  is Malicious | The URL <http://jam.canoe.ca/movies/artists/g/gottfried_gilbert/1997/02/21/758674.html>  is Malicious | PASSED |
| **5** | http://www.lushmedia.co.uk/ | The URL <http://www.lushmedia.co.uk/> is Malware | The URL <http://www.lushmedia.co.uk/> is Malware | PASSED |
| **6** | http://timesofindia.com | The URL <http://timesofindia.com> is benign | The URL <http://timesofindia.com> is benign | PASSED |

**7.3 EVALUATION METRICS**

The overall accuracy can be detected from the confusion matrix.The *overall accuracy* is calculated by summing the number of correctly classified values and dividing by the total number of values. The correctly classified values are located along the upper-left of the confusion matrix. The total number of values is the number of values in either the truth or predicted-value arrays.

****

Here,NL→L be the number of legitimate websites classified as legitimate

NL→P be the number of legitimate websites misclassified as phishing

NP→L be the number of phishing misclassified as legitimate

NP→P be the number of phishing websites classified as phishing.

**CHAPTER 8**

**CONCLUSION**

**CHAPTER 8**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**8.1. CONCLUSION**

Finally, phishing attacks are a major problem. It is important that they are countered. The work reported in this thesis indicates how understanding of the nature of phishing may be increased and provides a method to identify phishing problems in systems. It also contains a prototype of a system that catches those phishing attacks that evaded other defences, i.e. those attacks that have “slipped through the net”. An original contribution has been made in this important field, and the work reported here has the potential to make the internet world a safer place for a significant number of people.

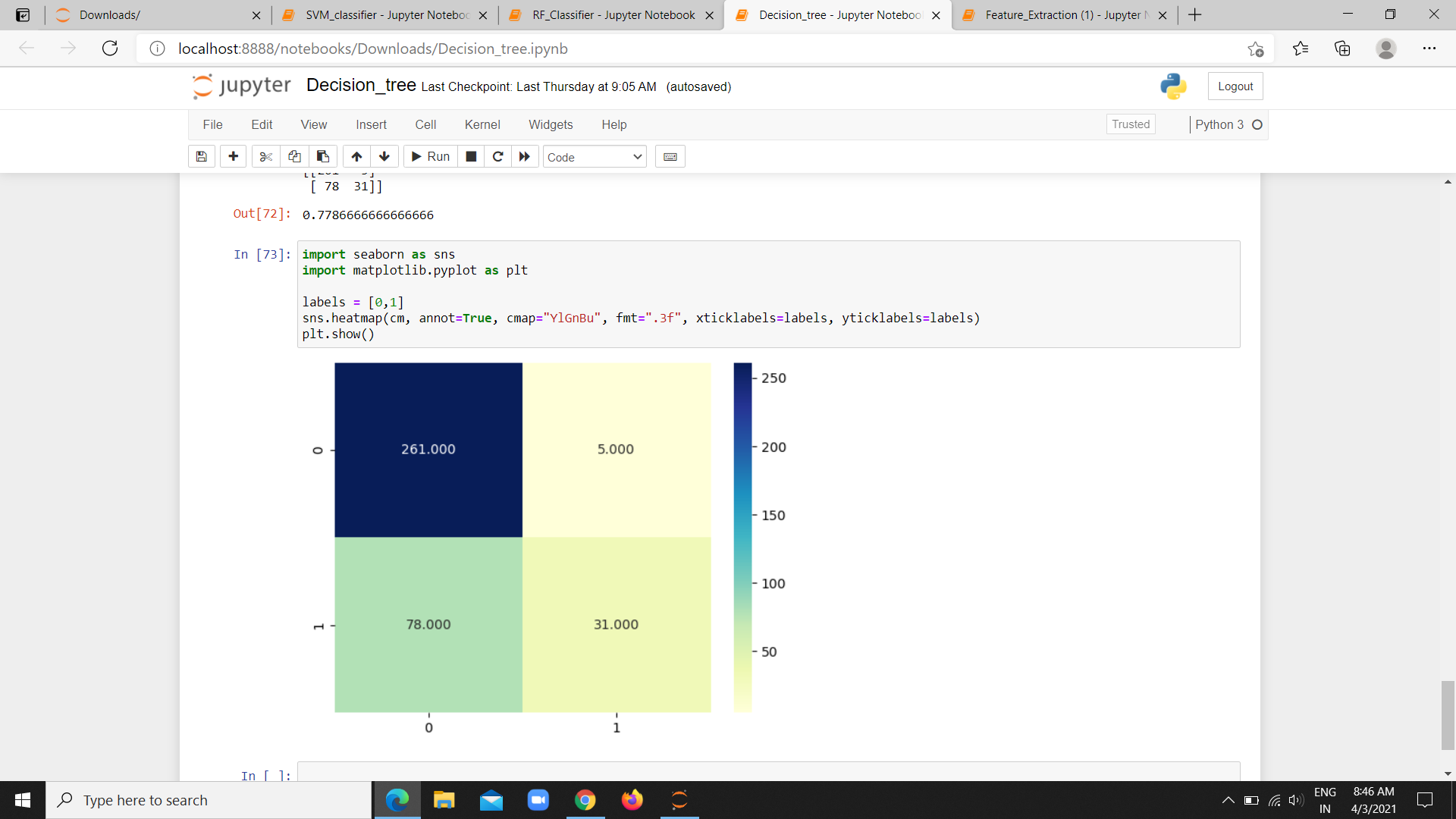
**8.2. Future Work**

In the future we will provide some technical solutions by improving the efficiency of spam filters. By which too many mails are classified correctly and properly. By this legitimate user can surf the internet with less fear.Although the use of URL lexical features alone has been shown to result in high accuracy (97%), phishers have learned how to make predicting a URL destination difficult by carefully manipulating the URL to evade detection. Therefore, combining these features with others, such as host, is the most effective approach .For future enhancements, we intend to build the phishing detection system as a scalable web service which will incorporate online learning so that new phishing attack patterns can easily be learned and improve the accuracy of our models with better feature extraction.

**APPENDICES**

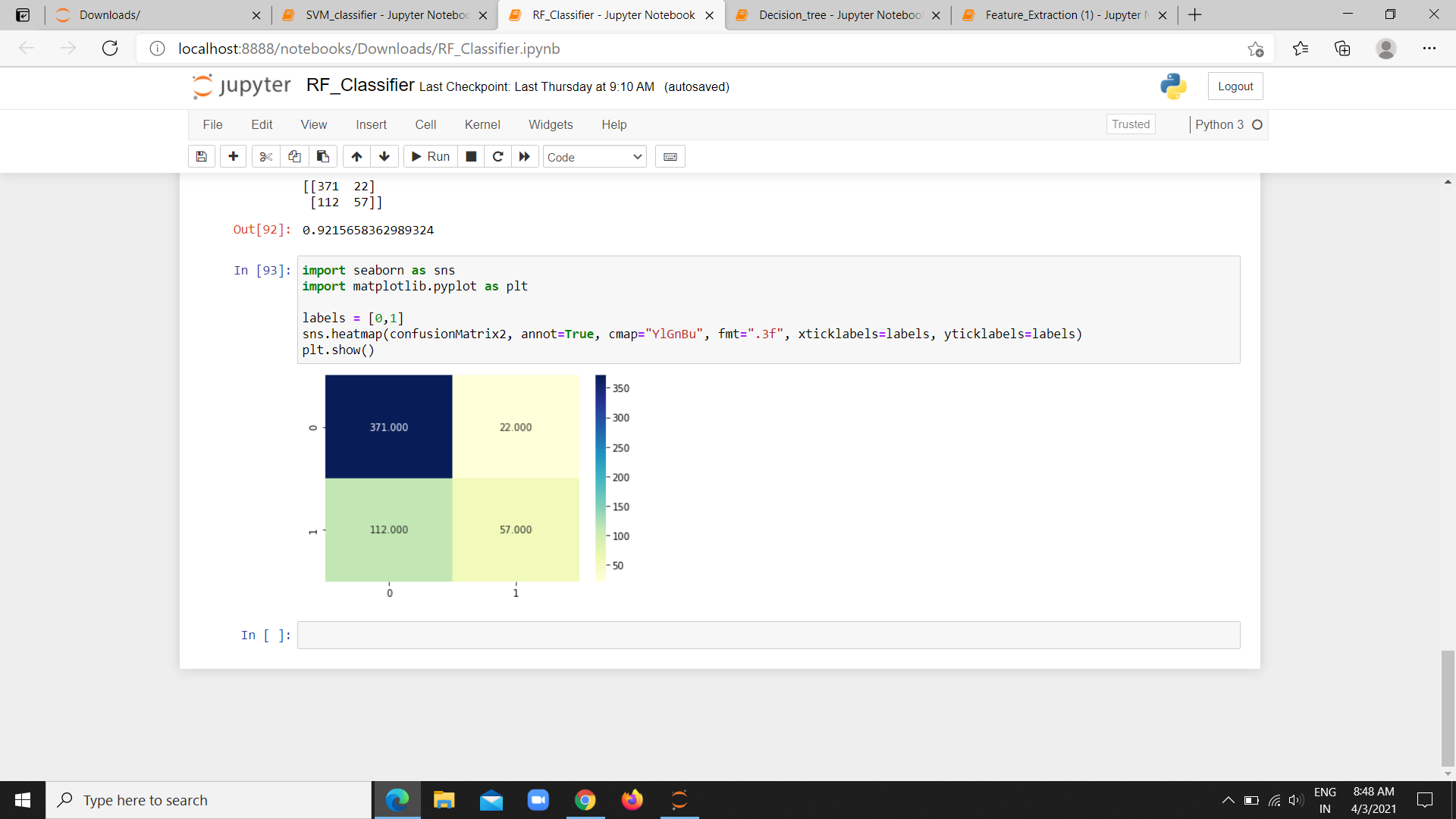
**A.1 SAMPLE SCREENSHOT**

**DECISION TREE**

****

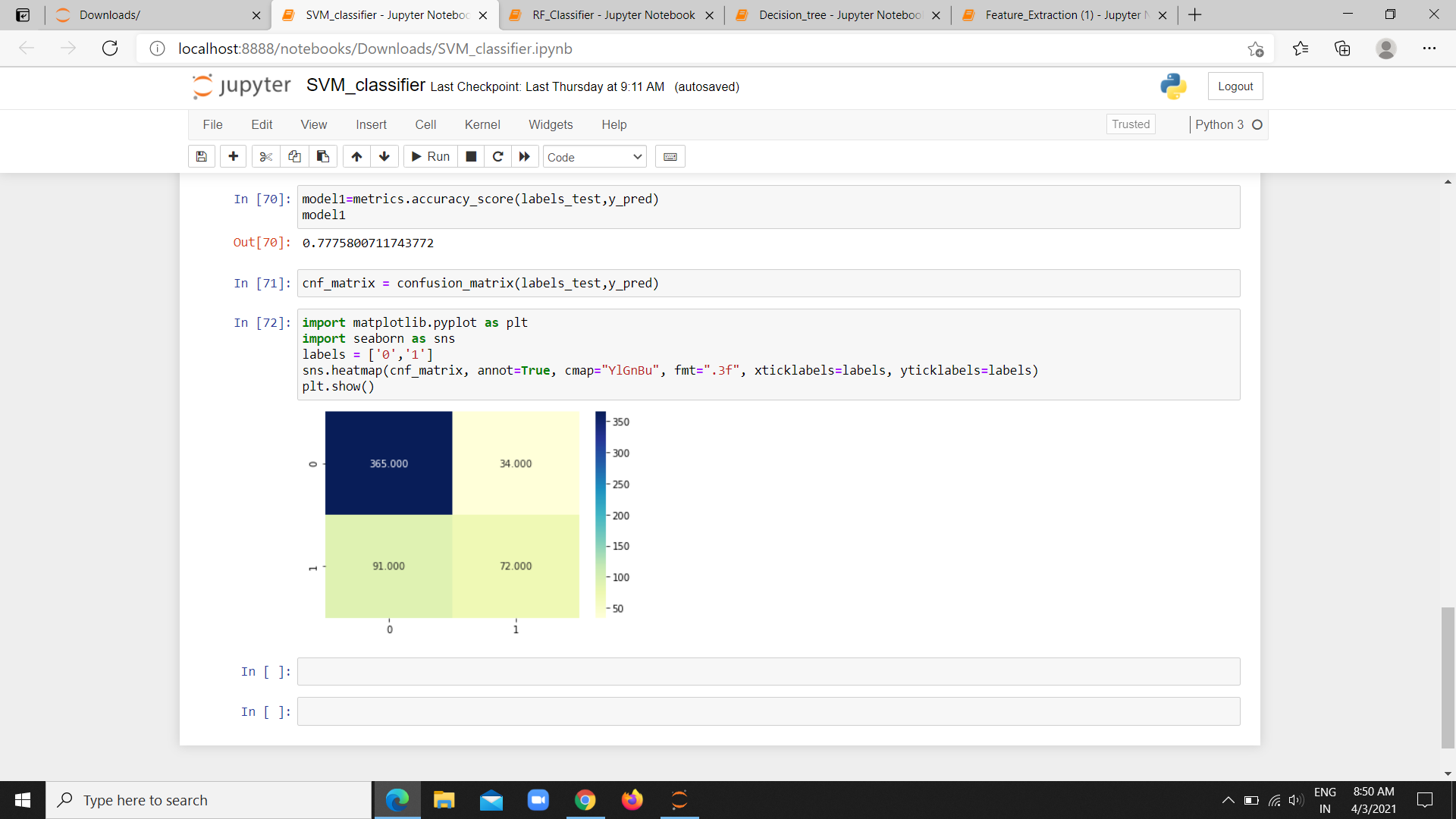
**Figure A.1.1**

**RF CLASSIFIER**

****

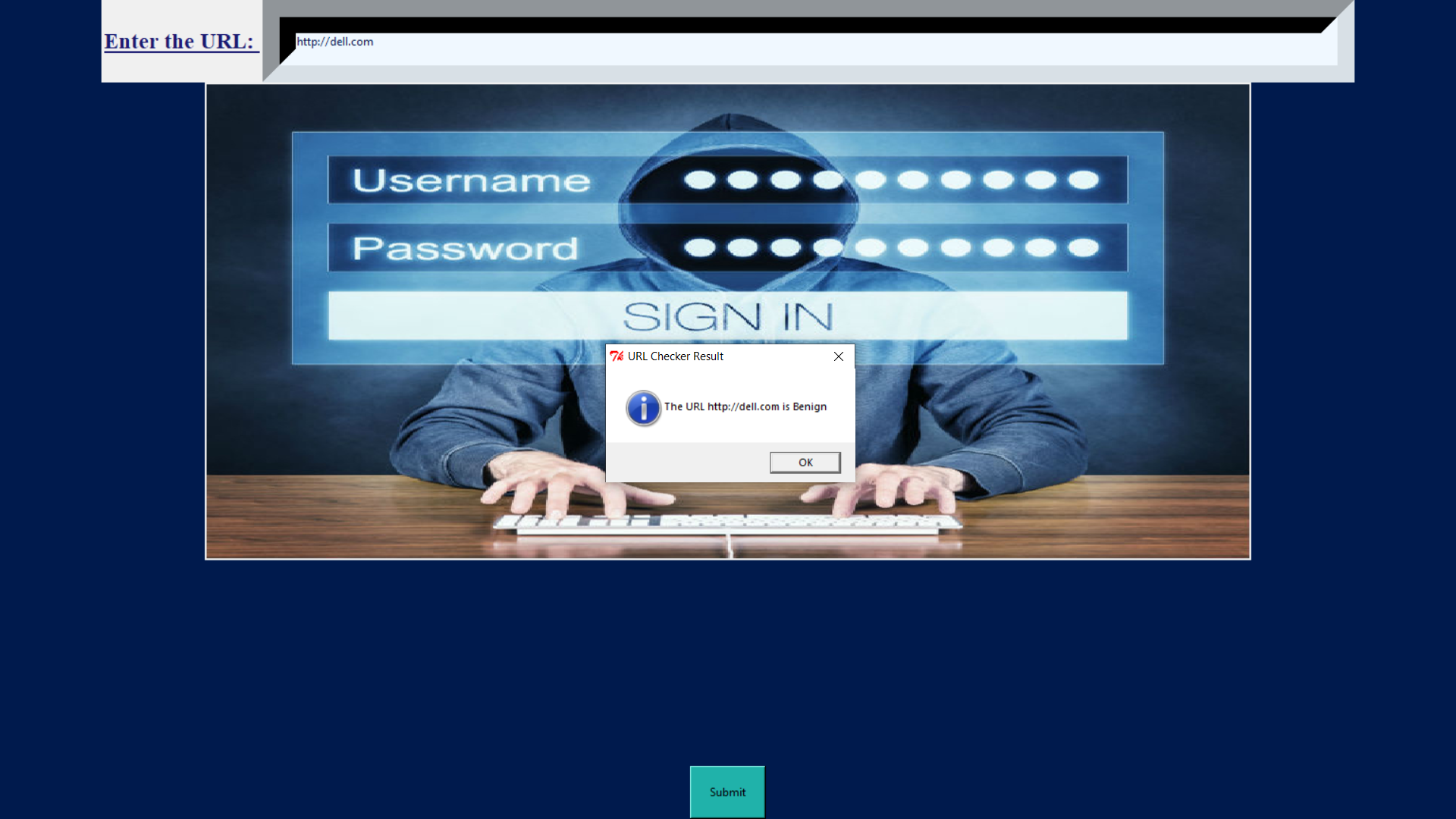
**Figure A.1.2**

**SVM CLASSIFIER**

****

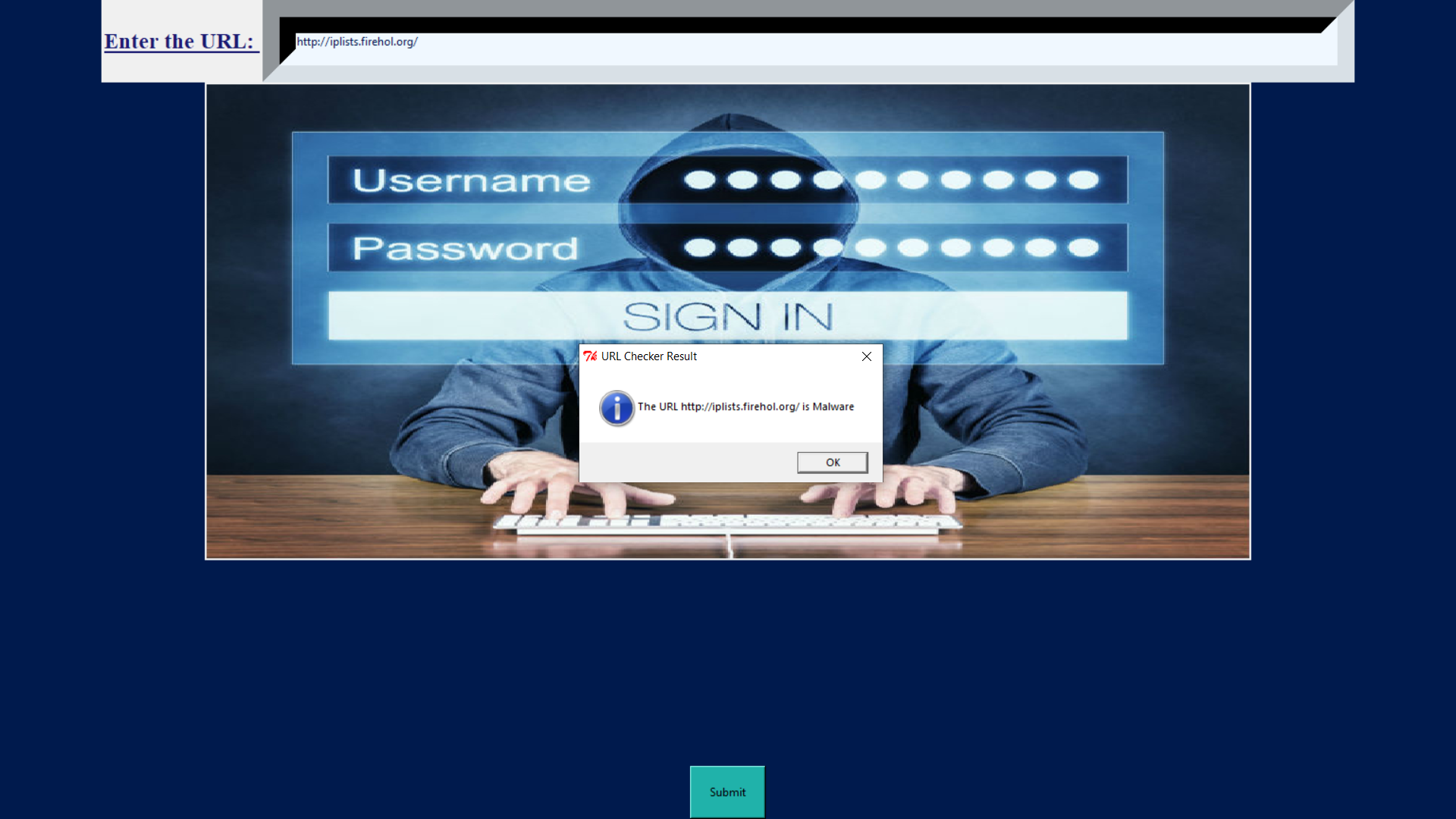
**Figure A.1.3**

**BENIGN SAMPLE:**

****

**Figure A.1.4**

**MALWARE SAMPLE:**

****

**Figure A.1.5**

**A.2 PUBLICATION**

This project was published as a paper in international journal of creative

research thoughts on 28.03.2021 by Giriswarna R and Harini P

**Published issue:** Vol 9 issue 3, March 2021

**Link:** <https://ijcrt.org/track.php?r_id=204951>

**REFERENCES**

**REFERENCES**

1. J. Shad and S. Sharma, “A Novel Machine Learning Approach to Detect Phishing Websites Jaypee Institute of Information Technology,” pp. 425–430, 2018.

2. Wen Zhang, Yu-Xin Ding, Yan Tang, Bin Zhao, ”Malicious web Page detection based on online learning algorithm”, IEEE-10(11),2011.

3. Birhanu Eshete, Adolfo Villafiorita , and Komminist Weldemariam ,”Holistic Analysis and Detection of Malicious Web pages” , International Conference on Security and Privacy in Communication Systems-volume:106, 2012.

4. S. Marchal, J. Francois, R. State, and T. Engel, “Phish Score: Hacking phishers minds” ,IEEE-11(9), 2015.

5. A. A. Ahmed and N. A. Abdullah, “Real time detection of phishing websites,” IEEE-7(11), 2016.

6. Ram B.Basnet , Andrew H.Sung , “ Learning to detect Phishing URLs”, IRJET-03(06), 2014

7. Khonji, Iraqi, Andy Jones,”Phishing Detection: A Literature Survey”,IEEE-13(4), 2010 .

8. M. Karabatak and T. Mustafa,”Performance comparison of classifiers on reduced phishing website dataset” ,IEEE-18(7), 2018.

9. A Novel Framework for learning to Detect Malicious web Pages - Wang Tao, Yu Shun Zheng, Xie Bailin , IJRET, 2010

10. Web Phishing Detection Using a Deep Learning Framework Ping Yi,YuxiangGuan,Futai Zou,Yao Yao, Wei Wang, and Ting Zhu ,IEEE, 2018.

11. Gilan Mark,Julia M.Taylor,“Using syntactic for phishing detection”,IEEE -10(4),2018.