**FAKE WEBSITE PHISHING ATTACK DETECTION USING MACHINE LEARNING**

ABSTRACT:

Because of the fast expansion of internet users, phishing attacks have become a significant menace where the attacker poses as a trusted entity in order to steal sensitive data, causing reputational damage, loss of money, ransomware, or other malware infections. Intelligent techniques mainly Machine Learning (ML) and Deep Learning (D L) are increasingly applied in the field of cyber security ~~due~~ to their ability to learn from available data in order to extract useful insight and predict future events. The effectiveness of applying such intelligent approaches in detecting phishing web sites is investigated in this paper. We used two separate datasets and selected the highest correlated features which comprised of a combination of content-based, URL lexical-based, and domain-based features. A set of ML models were then applied, and a comparative performance evaluation was conducted. Results proved the importance of features selection in improving the models' performance. Furthermore, the results also aimed to identify the best features that influence the model in identifying phishing websites. For classification performance, Random Forest (RF) algorithm achieved the highest accuracy for both datasets.

**EXISTING SYSTEM**

The current technologies will help to detect the malware phishing attacks. If it is program to switch on by 11 am it will switch on even when rain occurs during that time. Later moisture sensor came into the picture, which will measure the moisture content in the water. When water level is reduced the watering system gets automatically on. But if the local weather department predicts rain In the near future watering the plants can be delayed.

**DISADVANTAGE:**

* Forecasts are Never Completely Accurate
* It can be Time-Consuming and Resource-Intensive
* Lack of ability to be spatially invariant to the input data.
* Lots of training data is required.

**PROPOSED SYSTEM**

Deceiver uses sorts of ways to deceit the internet users. Fishers send emails to the users to verify the account. They use links and buttons and request to click them. Behind the links, there has a website where the hackers snatch and store the personal information of the users. This type of phishing is targeted at the higher executives of the organization. The content of the email is about important issues, and it is sent to the executives. The message of the email can be like the customers' complaints

**ADVANTAGE:**

* Early detection and chances of timely treatment.
* Our data set contains features with highly varying magnitudes and range.
* The ability of parallel processing
* The ability to work with insufficient knowledge

**SYSTEM SPECIFICATION**

**HARDWARE CONFIGURATION:**

* Processor - I5
* Speed - 3 GHz
* RAM - 8 GB(min)
* Hard Disk - 500 GB
* Key Board - Standard Windows Keyboard
* Mouse - Two or Three Button Mouse
* Monitor - LCD

**SOFTWARE CONFIGURATION**

* Operating System: Linux, Windows/7/10
* Server: Anaconda, Jupyter,Visual Studio Code
* Front End: Flask |GUI toolkit
* Server side Script: Python

**INTRODUCTION :**

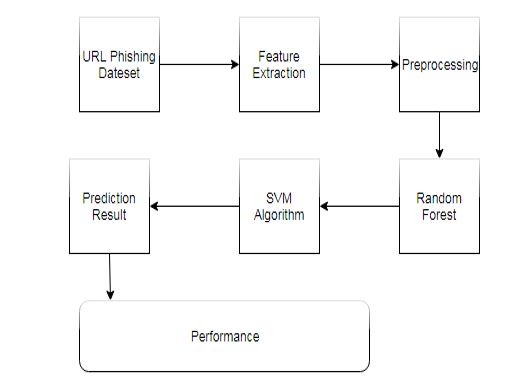
Phishing attacks have become anxiety for the cyber world. It causes enormous problems for privacy and financial issues of internet users. Scammers, namely fishers, create false websites [1, 2] to feel and look like a genuine to deceive the people. They spoof emails to steal the identity of legitimate users. They gather personal covert information, password, account information, and credit card details for the transaction. Fishers always change their strategy to attack the system. Social engineering [3-6] is one of the essential techniques the fishers use. Using this technique, they gather personal credentials from a trustworthy person. Phishers create false websites and spoof email in such a way that they are very similar and sometimes look like a real company website that comes from a source. Sometimes the attackers act like a real source and force the users to update the system. Moreover, they threaten the customer to suspend the account and demand ransom. Email spoofing is another technique used for phishing fraud [7]. Customers are usually misled to disclose private information like passwords and credit card number. Thus fishing is mainly used to steal valuable information such as bank account, password, and credit card details [8]. This type of scam is increasing rapidly, and individuals, business-people are losing their trust in online business. Thus, a negative impression of clients on online business was swarmed as they lost faith in online transactions. Even though encryption software is used to protect the information in the computers' storage, they are also vulnerable to attacks [9]. In this paper, the detection of fishing was performed through ML.

**II. BACKGROUND AND RELATED WORK**

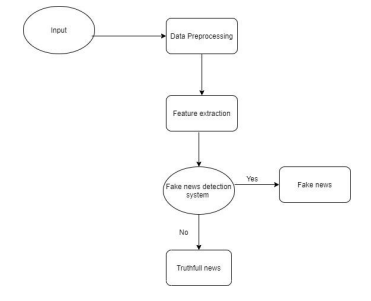
Supervised machine learning approaches are well suited for this type of classification based problem. To train these classifiers, the features of both phishing and legitimate websites need to be extracted and used machine learning

algorithms to train a model that can predict a phishing website's status concretely. While Phishers improve their skills of attacking day after day, machine learning can be used to train updated models that can prevent phishing scams by keeping up with the times. By the use of supervised machine learning methods, through analyzing the URLs, website structure, and other feature differences between phishing websites and legitimate websites, proposed work aimed to predict whether a website is phishing or not. This study mainly focusses on classifying phishing websites and legitimate websites by using several supervised machine learning methods. Their performance is then finally evaluated and taken into account to determine which of our discussed supervised machine learning methods works best to serve its purpose.

**ARCHITECTURE DIAGRAM:**



**UML DIAGRAM**



**LITERATURE SURVEY :**

# TOPI : REAL-TIME HAND POSTURE AND GESTURE-BASED TOUCHLESS AUTOMOTIVE USER INTERFACE USING DEEP LEARNING

**AUTHOR :** V. John, M. Umetsu, A. Boyali, S. Mita, M. Imanishi

YEAR : 2019

# DESCRIPTION : Phishing is an internet scam in which an attacker sends out fake messages that look to come from a trusted source. A URL or file will be included in the mail, which when clicked will steal personal information or infect a computer with a virus. Traditionally, phishing attempts were carried out through wide-scale spam campaigns that targeted broad groups of people indiscriminately. The goal was to get as many people to click on a link or open an infected file as possible. There are various approaches to detect this type of attack. One of the approaches is machine learning. The URL’s received by the user will be given input to the machine learning model then the algorithm will process the input and display the output whether it is phishing or legitimate. There are various ML algorithms like SVM, Neural Networks, Random Forest, Decision Tree, XG boost etc. that can be used to classify these URLs. The proposed approach deals with the Random Forest, Decision Tree classifiers. The proposed approach effectively classified the Phishing and Legitimate URLs with an accuracy of 87.0% and 82.4% for Random Forest and decision tree classifiers respectively.

# TOPIC : Fake Website Prediction Using Random Forest

**AUTHOR :**[Mythilipriya C](https://ieeexplore.ieee.org/author/37088974162); [Priyadharshini S](https://ieeexplore.ieee.org/author/37088974241); [Karan S](https://ieeexplore.ieee.org/author/37089392867); [Sugantha Priyadharshini P](https://ieeexplore.ieee.org/author/37088973584)

YEAR : 2019

**DESCRIPTION :** Fake websites are now producing billions of dollars in fraud at the cost of innocent Internet users. Users will have a difficult time manually identifying these websites as phoney due to their design and appearance. A large number of people buy things online and pay for them using various online payment platforms. Several websites require users to enter sensitive data in order to authenticate. However, some phishing websites make use of this information for nefarious purposes. Automated detection systems have arisen as a means of countering bogus websites, although the most of them are rather basic in terms of fraud and detecting methods. Here, in this project an website is been created for the users with the algorithm having an accuracy of 96% for detecting the website they are using in their daily routine.

# TOPIC : Detection of Phishing Websites from URLs by using Classification Techniques on WEKA

**AUTHOR :** [Buket Geyik](https://ieeexplore.ieee.org/author/37088451904); [Kübra Erensoy](https://ieeexplore.ieee.org/author/37088452520); [Emre Kocyigit](https://ieeexplore.ieee.org/author/37085430488)

YEAR : 2019

**DESCRIPTION :** The Internet is getting stronger day by day and it makes our lives easier with many applications that are executed on cyberworld. However, with the development of the internet, cyber-attacks have increased gradually and identity thefts have emerged. It is a type of fraud committed by intruders by using fake web pages to access people's private information such as userid, password, credit card number and bank account numbers, etc. These scammers can also send e-mail from many important institutions and organizations by using phishing attacks which imitate these web pages and acts as if they are original. Traditional security mechanisms can not prevent these attacks because they directly target the weakest part of connection: end-users. Machine learning technology has been used to detect and prevent this type of intrusions. The anti-phishing method has been developed by detecting the attacks made with the technologies used. In this paper, we combined the websites used by phishing attacks into a dataset, then we obtained some results using 4 classification algorithms with this dataset. The experimental results showed that the proposed systems give very good accuracy levels for the detection of these attacks.

# TOPIC : Phishing websites detection through supervised learning networks

**AUTHOR :** [Priyanka Singh](https://ieeexplore.ieee.org/author/37085680710); [Yogendra P.S. Maravi](https://ieeexplore.ieee.org/author/37085692705); [Sanjeev Sharma](https://ieeexplore.ieee.org/author/37667983900)

YEAR : 2019

**DESCRIPTION :** Phishing is an unlawful activity of making gullible people to reveal their insightful information into fake websites. The Aim of these phishing websites is to acquire confidential information such as usernames, passwords, banking credentials and some other personal information. Phishing website looks similar to legitimate website therefore people cannot make difference among them. Today users are heavily relying on the internet for online purchasing, ticket booking, bill payments, etc. As technology advances, the phishing approaches being used are also getting progressed and hence it stimulates anti-phishing methods to be upgraded. In this paper, we have implemented two algorithms named Adaline and Backpropion along with the support vector machine to enhance the detection rate and classification.

# TOPIC : Detection of Phishing Websites through Computational Intelligence

**AUTHOR :** [Muhammad Taseer Suleman](https://ieeexplore.ieee.org/author/37089281765); [Amir Ali](https://ieeexplore.ieee.org/author/37085484808)

YEAR : 2020

**DESCRIPTION :** Phishing is a technique used by hackers to fool internet users reveal their sensitive information like passwords, credit card numbers, contact information, and address, etc. Web phishing is carried out mostly by sending fake web links to the users through different communication means like Email, Facebook Messenger and WhatsApp, etc. Web phishing detection is significant for making internet browsing safe and secure for users. Different approaches were applied for the detection of fake websites. However, the most efficient method for detecting phishing websites is the one that is based on artificial intelligence and learning mechanism. In this research, an efficient and accurate method is proposed for the detection of phishing websites which is based on computational intelligence. Through the development of different computational models and rigorous testing, it was revealed that Extreme Gradient Boost (XGBoost) based model achieved the maximum scores in all the validation tests. This shows that the model is robust and accurate in terms of web-phishing detection

**CHAPTER-2**

**III. METHODOLOGY :**

Machine Learning is a study of algorithms where using mathematical modelling with probabilistic theories decision making for solving a problem is done based on some amount of previous data or scenario of that problem. Machine learning is building mathematical models, integration of high-level equations which output the value of a target variable based on some dependent variable. Analyzing the data of phishing and legitimate websites, based on their different characteristics, a machine learning model can predict whether a new unknown website would be phishing or a legitimate one. Supervised learning is a predictive model built on known outcomes. The model predicts over a set of known values. In the training dataset, every single instance has a label referring to a class. Real-world classification based problems like phishing detection, spam mail detection are solved using supervised learning methods. Random Forest, Classification and Regression Tree, K Nearest Neighbors, Support Vector Machine, Logistic Regression are some of the popular supervised machine learning methods used for classification based problems.

1. **Dataset**

The dataset is one of the most critical parts of our study. A dataset is nothing but the table containing information about phishing and legitimate websites—the dataset for our proposed model obtained from Kaggle. Kaggle is one of the most popular public repositories with a tremendous amount of dataset collection which can be used for training machine learning models. The data set we have used for our work Fig. 1 Proposed Architecture for phishing attack detection contains 32 attributes 11504 instances. The attributes of this dataset are: Index, UsingIP, GoogleIndex, LongURL, ShortURL, Symbol@, Redirecting, PrefixSuffix-, DNSRecording, SubDomains, HTTPS, DomainRegLen, Favicon, NonStdPort, HTTPSDomainURL, RequestURL, AnchorURL, LinksInScriptTags, ServerFormHandler, InfoEmail, AbnormalURL, WebsiteForwarding, StatusBarCust, DisableRightClick, UsingPopupWindow, AgeofDomain, WebsiteTraffic, PageRank, LinksPointingToPage, IframeRedirection, StatsReport, and class. We don't need the "Index" attribute here as this is just the index number of the instances in the dataset. The "class" attribute is our target variable which we are going to predict

1. Data Preprocessing

Feature scaling is the process of normalizing or standardizing the independent variables of the training dataset to a fixed range, to handle variance in the values among different independent variables. Splitting the dataset into two portions, one for training and one for testing is very important. It is vital to train a model with a subset of the full dataset and test model with the rest to evaluate the model performance satisfactorily. We split the dataset into 80:20 ratio as follows: 80% of the dataset used for training and 20% of dataset for testing using a stratified sampling technique. We did the train test split using the Scikit-Learn library in Python programming language.

**C.Machine Learning Classifiers**

Three machine learning classifiers are applied in this research. They are KNN, logistic regression, and random forest. The k-nearest neighbours (KNN) classifier is a simple supervised machine learning classifier. It is used both classification and regression problems. It relies on labelled data to acquire a function that predict the outcome when given new unlabeled data is given. In this research, the KNN algorithm uses 80% labelled data to acquire a function to predict whether a website is a real or a phishing website. The second classifier name is logistic regression. Logistic is a statistical model. It uses a logistic function to model a binary dependent variable. In our regression analysis, uses 80% labelled data to acquire a logistic function to predict whether a website is a legitimate or a phishing website. The third classifier in this research is the random forest and is a supervised learning algorithm. It uses a set of decision trees which build the forest. It is an ensemble of decision trees, usually trained with the "bagging" technique. The main idea of the bagging technique is that a mixture of learning models surges the global effect. IV. RESULTS AND DISCUSSIONS In our study, we used confusion matrixes, ROC curves, precision, recall, and F1 Score to evaluate the performance of the three machine learning classifiers. Fig. 2 Classification report for KNN Fig. 2, Fig. 3, and Fig.4 show the performance of the KNN algorithm. Fig. 2 shows the precision, recall, and fi score for the KNN algorithm. It is observed that the precision is 91% for a phishing website. On the other hand, the precision is 86% for the legitimate website. Besides, we see that recall and fi score are 94% and 93% respectively for phishing website. The recall and fi score for legitimate website are 79% and 82% respectively.

**II. RELATED WORK**

Due to problems associated to phishing techniques, the scientific community and industry have been researching and implementing mechanisms to reduce and mitigate these attacks, to reduce risks and protect organizations’ information and services. Gastellier-Prevost et al. [6] developed a toolbar (Phishark) for Phishing detection by means of different tests, such as white and black lists of websites, of which the main point of the evaluations has been the heuristic analysis. Martin et al. [7] used criteria and indicators in order to determine whether an email may be already itself Phishing. The various used criteria have been URL, Security, Encryption, JS Code, CSS, HTML and Human Factor, with the purpose to a later use of Neural Networks and the PhishTank application (PHP module). Aburrous et al. [8] presented an approach to overcome the lack of 202 2019 International Conference on Computational Science and Computational Intelligence (CSCI) 978-1-7281-5584-5/19/$31.00 ©2019 IEEE DOI 10.1109/CSCI49370.2019.00041 clarity in the evaluation of e-banking phishing websites. The model is based on fuzzy logic combined with data mining algorithms. Abu-Nimeh et al. [9] compared the predictive accuracy of several machine learning methods that may predict phishing emails, logistic regression, regression and classification trees, Bayesian additive regression trees, support vector machines, random forests, and neural networks. Hamid et al. [10] presents a method of the identification of behavior-based characteristics in phishing emails that lacked to be disguised by an attacker. Basnet et. al [11] presented the evaluation of traditional ML techniques. Hereby, 16 relevant characteristics were chosen, with the collection of 4000 URLs between phishing URLs and benign URLs. Finally, some algorithms have been evaluated such as Support Vector Machines (SVM), Neural Networks, Self-Organizing Maps and K-Means. Saxe et al. [12], presented a different method to analyze URL’s character strings, which has been called subsequently obfuscation. In order to address this concern, they proposed the use of a deep learning technique called eXpose neural network. Vazhayil et al. [13] proposed a comparative analysis between different ML detection methods to detect phishing URLs. Hereby, they firstly evaluated traditional ML methods, and then these were compared with Deep Learning. In Vanhoenshoven [14], several classifiers such as Support Vector Machines, Random Forest, Naive Bayes, Decision Trees, KNeest Nearest, and MultiLayer Perceptron were evaluated and compared with respect to the binary classification of malicious URLs. Zhang et al. [15] proposed a synthetic minority oversampling technique through of a Borderline-Smote algorithm to resolve the problem of unbalanced data. The first three groups of characteristics of URL phishing pages are extracted of web pages and image characteristics. Based on the outcomes of the previously presented studies, the focus of our study has been emphasizing on the availability of a solution through ML techniques directing efforts to cognitive security

**Modeling phases using Machine Learning** :

For the development of the model, the typical Machine Learning modeling cycle (as stated in [16] and [17]) has four stages. The first stage called “Manage data”, starts with a phase of capture a set of training data. Once the training data have been collected, one should explore the data to a better understand about its structure and meaning. The data will usually need to be cleaned to be useful. This may involve formatting and vectorization, which is a process intended to convert data into mathematical constructions that recognizes and understands ML models. Once cleaned, it will need to prepare the data further in order to load it into a programming environment. Lastly, it needs to divide the data into subsets of training and validation. At the training stage, the general approach is decided by selecting a learning task (i.e., prediction, grouping, etc.)In general, the characteristics available in the training data set are studied and new ones are designed if necessary. The learning approach complies with the following process: (1) It begins with the training of the data set; (2) the data are transformed into a representation or a list of vectors; (3) the vector list is processed with a learning algorithm; and (4) the model is constructed with the parameters obtained from the learning algorithm. The inference approach is used to convert the data into a vector of features in order to generate a model and to obtain the prediction of results [17]. Hereby, the process takes less time than learning and consists of testing the model with new data and observing its performance. In the stage, named ”Evaluate model”, the output is evaluated using different approaches and algorithms. Models are evaluated using appropriate metrics, such as punctuation, recovery and accuracy rate. The hyperparameters of the models are further adjusted until they are fulfilled with an acceptable model. The Deploy Model, which represent the final stage, is applied to new data and the results are monitored. Hereby, predictions or deductions are performed on data that have not been previously considered or explored. In the case of improving the model, the cycle needs to be started again. The following sections describes the applied methodological procedure and the ML techniques that were used within the proposed model. B. Jupyter Framework Architecture. The Jupyter framework architecture has been used for the development of the proposed model It allows integrating blocks of code, text, graphics or images into the same document. It is a system based on the implementation of the Kernel as an engine for the execution of programming various languages such as Python, Ruby, Julia, R, among others. This implies that the process needs to contain a queue system based on ZeroMQ. Hereby, it realizes messages to other systems [18].

**The architecture of the detection and mitigation:**

Those model shows the architecture of the model in a controlled environment. It starts with the delivery of emails by the cyber-attacker through the Simple Mail Transfer Protocol (SMTP) to a victim mail server. The Internet Message Access Protocol (IMAP) that the user’s computer receives manages them. The model extracts the data from the emails with the Naive Bayes classification algorithm and continues with the analysis of the information using the Decision Tree classifier for the detection of illegitimate emails. Lastly, if the message has been received by the victim and corresponds to an infected email, than necessary preventive and mitigation measures are executed. Thus, the model starts with the reading of the received emails and performs the extraction of the required information. The data using the Naive Bayes algorithm are scanned, in order to extract the characteristics and to categorize the data. The flow continues with the delivery of the information to the Decision Trees algorithm for the detection 203 of the attack. Finally, a review of the results is accomplished in order to send the phishing emails to be quarantine. Fig. 1. The architecture of the model in a controlled attack environment. For the detection of emails infected with phishing, we conducted the following procedure: (1) It starts with the user’s request to detect phishing in the emails that are in the inbox through the created user graphical interface; (2) Then it proceeds to read the emails to the server; (3) The mails are sent to a database that stores the characteristics (features), where 18 peculiarities of each email are extracted and the data matrix that is stored in a file with CSV extension; (4) This matrix is sent with the data to the Naive Bayes Classifier for model training; (5) If the prediction percentage is lower than 95%, a readjustment check is performed with the Decision Trees Classifier; (6) This classifier sends a response message with the prediction percentage reached by the training algorithm. Once finished, the Decision Tree classifier intervenes in the detection; (7) If the email has been infected with phishing, it is sent to quarantine emails (Figure 2); (7) Finally, it returns a message with the result of each analyzed email.

Algorithm 1 : Algorithm to verify email with phishing. Initialization:

1: repeat

2: Read features in csv

3: until features in csv = 1325

4: x ← features

5: z ← result

6: x training ← train test split (x, z, 0.30,18)

7: x test ← train test split (x, z, 0.30,18)

8: y test ← training test (x, z, 0.30,18)

9: y training ← training test (x, z, 0.30,18)

10: model ← DecisionTree (1.5)

11: model fit ← adjustment (x training, y training)

12: result ← prediction (x test, y test)

13: if (result == 0) then

14: write ”No Phishing”

15: else

16: write ”Yes Phishing”

17: end if

18: End Algorithm

ACKNOWLEDGMENT

Funding of this research has been provided by the Ecuadorian Corporation for the Development of Research and the Academy - CEDIA), within the Project Grant GT-Cybersecurity. We are also grateful to the Mobility Regulation of the Universidad de las Fuerzas Armadas ESPE, in Sangolqu´ı, Ecuador, as part of the Project entitled “Detection and mitigation of Social Engineering attacks using Cognitive Security”.

**CHAPTER-3**

**MODULE DISCRIPTON:**

**Register Module:**

**Description:**

The Register module allows users to create accounts within the fake website detection system.

**Components:**

**User Registration Form:**

Users provide essential information such as a unique username, email address, and password.

Implement form validation to ensure data integrity and security.

**Database Interaction:**

Flask-MySQLDB to interact with the MySQL database, storing user account details securely.

**Login Module:**

**Description:**

The Login module allows registered users to access the fake website detection system.

**Components:**

**User Login Form:**

Authenticated users input their credentials (username and password) for login.

**Credential Verification:**

Verify login credentials against the stored information in the MySQL database.

Implement security measures for failed login attempts (e.g., account lockouts, temporary suspension).

**URL Input Module:**

**Description:**

The URL Input module allows users to submit a website URL for analysis within the fake website detection system.

**Components:**

**User Input Form:**

Users submit the URL they want to analyze through a designated form.

**URL Validation:**

Validate the URL format and structure to ensure accurate analysis.

**Model Checking Module:**

**Description**:

The Model Checking module is responsible for using pre-trained models to assess the authenticity of a given website.

**Components:**

Integration with Machine Learning Models:

Incorporate machine learning models trained to detect fake websites based on various features.

**Model Evaluation:**

Evaluate the website against the pre-trained model to determine its authenticity.

**Feature Extraction Module:**

**Description:**

The Feature Extraction module analyzes various components of the submitted website to extract relevant features for further analysis.

**Components:**

**URL Features:**

Extract features from the URL, such as length, presence of special characters, and domain information.

**Content Features:**

Extract features from the website's content, including text and images.

**Design and Functionality Features:**

Extract features related to the website's design and functionality.

**Output Module:**

**Description:**

The Output module presents the results of the website analysis to the user.

**Components:**

**User-Friendly Output Interface:**

Display the analysis results in a user-friendly format.

**Authenticity Decision:**

Communicate whether the website is determined to be legitimate or flagged as potentially fake.

**Detailed Analysis:**

Provide detailed information on the analysis, highlighting specific aspects that led to the authenticity determination.

**CHAPTER-4**

**PROPOSED SYSTEM ANALYSIS AND DESIGN**

**4.1 INTORDUCTION**

In product development, it is important to understand the difference between the baseline functionality necessary for any system to compete in that product domain, and features that make the system different from their competitor's products. Some strategies have important implications for software architecture. Specifically, it is not just the Software requirements specifications of the initial release that must be supported in the architecture. The Software requirements specifications of initial products need to be explicitly taken into consideration.

**4.2 REQUIREMENT ANALYSIS**

**4.2.1 FUNCTIONAL REQUIREMENTS**

In Software engineering and systems engineering, a functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs. ... This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability.

**4.2.1.1 PRODUCT PERSPECTIVE**

The product is supposed to be an open source, under the GNU general Public License. It is a web based system implementing client-server model. The portal System provides simple mechanism for users to share and acquire knowledge.

**4.2.1.2 PRODUCT FEATURES**

The following are the main features

* Cross platform support:
* Offers operating support for most of the known and commercial operating systems. User account: The system allows the user to create their accounts in the system and provide features of updating and viewing profiles.
* Number of users being supported by the system:
* Though the number is precisely not mentioned but the system is able to support a large number of online users at a time. Search: search is simply local search engine based on key words.
* Discussion Forum:
* Provides users with a platform to discuss and help each other with their problems
* Ticketing system: Allows user to submit his issue to the admin in case his problems are not solved by FAQs and discussion forums.
* FAQs section: Frequently asked section contains answer of problem which tablet user frequently faced.

**4.2.1.3 USER CHARACTERISTICS**

It is considered that the user do have the basic knowledge of operating the internet and to have access to it. The administrator is expected to be familiar with the interface of the tech support system.

**4.2.1.4 ASSUMPTION AND DEPENDENCIES**

This software highly depends on type and version of browser being installed in the system i.e. browser version should be used which have HTML5 support.

**4.2.1.5 DOMAIN REQUIREMENT**

Domain requirement is the Requirement that comes from the application domain of the system that reflects the characteristics of that domain. Therefore, as our System is an inventory System, the domain requirement of this system should concern about the requirements that reflect characteristic of Inventory System.

**4.2.2 NON-FUNCTIONAL REQUIREMENTS**

In systems engineering and **requirements** engineering, a **non**-**functional requirement** (NFR) is a **requirement** that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with **functional requirements** that **define** specific behavior or functions.

**4.2.2.1 PRODUCT REQUIREMENTS**

**4.2.2.1.1 EFFICIENCY**

**SPACE EFFICIENCY**

Storage **efficiency** is the ability to store and manage data that consumes the least amount of **space** with little to no impact on performance; resulting in a lower total operational cost. **Efficiency** addresses the real-world demands of managing costs, reducing complexity and limiting risk.

**TIME EFFICIENCY**

The state or quality of being **efficient**, or able to accomplish something with the least waste of **time** and effort is Time efficiency; competency in performance. And accomplishment of or ability to accomplish a job with a minimum expenditure of **time** and effort.

**4.2.2.1.2 RELIABILITY**

Reliability Posted by: Margaret Rouse WhatIs.com Contributor(s): KajBackholm Word of the Day 5G is the coming fifth-generation wireless broadband technology based on the IEEE 802.11ac standard. An important goal of 5G is to erase the differences between wire line and wireless networking to accommodate the growing mobility of network users. Subscribe to the Word of the Day Word of the Day Archive 20 Newest and Updated Terms competitive advantage mobile application management (MAM) Avro (Apache Avro) quality assurance (QA) gross revenue voice recognition (speaker recognition) Amazon Pinpoint employee engagement software Microsoft Project Honolulu project scope unstructured data hands-off infrastructure management Microsoft Windows Insider Program for Business risk map (risk heat map) VMware vCenter Server (formerly VMware Virtual Center) Advanced Message Queuing Protocol (AMQP) network engineer cloud storage service Ensile cloud backup (online backup) Reliability is an attribute of any computer-related component (software, or hardware, or a network, for example) that consistently performs according to its specifications. It has long been considered one of three related attributes that must be considered when making, buying, or using a computer product or component. Reliability, availability, and serviceability - RAS, for short - are considered to be important aspects to design into any system. In theory, a reliable product is totally free of technical errors; in practice, however, vendors frequently express a product's reliability quotient as a percentage.

**4.2.2.1.3 PORTABILITY**

**Portability** is a characteristic attributed to a computer program if it can be used in an operating systems other than the one in which it was created without requiring major rework. Porting is the task of doing any work necessary to make the computer program run in the new environment.

**4.2.2.1.4 USABILITY**

**Usability** is the ease of use and learnability of a human-made object such as a tool or device. In **software engineering**, **usability** is the degree to which a **software**can be used by specified consumers to achieve quantified objectives with effectiveness, efficiency, and satisfaction in a quantified context of use.

**4.2.2.2 ORGANISATIONAL REQUIREMENTS**

**4.2.2.2.1 IMPLEMENTATION REQUIREMENTS**

## LANGUAGEDESCRIPTION:

**PYTHON:**

Python is an interpreter, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding; make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace.

A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.It ranges from simple automation tasks to gaming, web development, and even complex enterprise systems. These are the areas where this technology is still the king with no or little competence: Machine learning as it has a plethora of libraries implementing machine learning algorithms.

Python is a one-stop shop and relatively easy to learn, thus quite popular now. What other reasons exist for such universal popularity of this programming language and what companies have leveraged its opportunities to the max? Let’s talk about that. Python technology is quite popular among programmers, but the practice shows that business owners are also Python development believers and for good reason. Software developers love it for its straightforward syntax and reputation as one of the easiest programming languages to learn. Business owners or CTOs appreciate the fact that there’s a framework for pretty much anything – from web apps to machine learning. Moreover, it is not just a language but more a technology platform that has come together through a gigantic collaboration from thousands of individual professional developers forming a huge and peculiar community of aficionados. So what is python used for and what are the tangible benefits the language brings to those who decided to use it? Below we’re going to discover that. Productivity and Speed It is a widespread theory within development circles that developing Python applications is approximately up to 10 times faster than developing the same application in Java or C/C++. The impressive benefit in terms of time saving can be explained by the clean object-oriented design, enhanced process control capabilities, and strong integration and text processing capacities. Moreover, its own unit testing framework contributes substantially to its speed and productivity.

**PYCHARM**

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

Choose the best PyCharm for you﻿

**PyCharm is available in three editions:**

* Community (free and open-sourced): for smart and intelligent Python development, including code assistance, refactorings, visual debugging, and version control integration.
* Professional (paid) : for professional Python, web, and data science development, including code assistance, refactorings, visual debugging, version control integration, remote configurations, deployment, support for popular web frameworks, such as Django and Flask, database support, scientific tools (including Jupyter notebook support), big data tools.
* Edu (free and open-sourced): for learning programming languages and related technologies with integrated educational tools.
* For details, see the editions comparison matrix.

**Supported languages﻿**

To start developing in Python with PyCharm you need to download and install Python from python.org depending on your platform.

PyCharm supports the following versions of Python:

Python 2: version 2.7

Python 3: from the version 3.6 up to the version 3.10

Besides, in the Professional edition, one can develop Django, Flask, and Pyramid applications. Also, it fully supports HTML (including HTML5), CSS, JavaScript, and XML: these languages are bundled in the IDE via plugins and are switched on for you by default. Support for the other languages and frameworks can also be added via plugins (go to Settings | Plugins or PyCharm | Preferences | Plugins for macOS users, to find out more or set them up during the first IDE launch).

### SUPPORTED PLATFORMS﻿

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Minimum** | **Recommended** |
| RAM | 4 GB of free RAM | 8 GB of total system RAM |
| CPU | Any modern CPU | Multi-core CPU. PyCharm supports multithreading for different operations and processes making it faster the more CPU cores it can use. |
| Disk space | 2.5 GB and another 1 GB for caches | SSD drive with at least 5 GB of free space |
| Monitor resolution | 1024x768 | 1920×1080 |
| Operating system | Officially released 64-bit versions of the following:   * Microsoft Windows 8 or later * macOS 10.13 or later * Any Linux distribution that supports Gnome, KDE, or Unity DE. PyCharm is not available for some Linux distributions, such as RHEL6 or CentOS6, that do not include [GLIBC](https://ftp.gnu.org/gnu/libc/) 2.14 or later.   Pre-release versions are not supported. | Latest 64-bit version of Windows, macOS, or Linux (for example, Debian, Ubuntu, or RHEL) |

**SPYDER**

**Spyder** is an open-source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the scientific Python stack, including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open-source software. It is released under the MIT license.

Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first-party and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, [Pylint](https://en.wikipedia.org/wiki/Pylint" \o "Pylint) and Rope. It is available cross-platform through [Anaconda](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)), on Windows, on macOS through [MacPorts](https://en.wikipedia.org/wiki/MacPorts" \o "MacPorts), and on major Linux distributions such as [Arch Linux](https://en.wikipedia.org/wiki/Arch_Linux), [Debian](https://en.wikipedia.org/wiki/Debian), [Fedora](https://en.wikipedia.org/wiki/Fedora_(operating_system)), [Gentoo Linux](https://en.wikipedia.org/wiki/Gentoo_Linux), [openSUSE](https://en.wikipedia.org/wiki/OpenSUSE) and [Ubuntu](https://en.wikipedia.org/wiki/Ubuntu_(operating_system)).

Spyder uses [Qt](https://en.wikipedia.org/wiki/Qt_(software)) for its GUI and is designed to use either of the [PyQt](https://en.wikipedia.org/wiki/PyQt" \o "PyQt) or [PySide](https://en.wikipedia.org/wiki/PySide" \o "PySide) Python bindings. QtPy, a thin abstraction layer developed by the Spyder project and later adopted by multiple other packages, provides the flexibility to use either backend.

**FEATURES**

Features include:

* An editor with [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)), [code completion](https://en.wikipedia.org/wiki/Code_completion)
* Support for multiple [IPython](https://en.wikipedia.org/wiki/IPython" \o "IPython) [consoles](https://en.wikipedia.org/wiki/Command-line_interface)
* The ability to explore and edit [variables](https://en.wikipedia.org/wiki/Variable_(computer_science)) from a [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface)
* A Help pane able to retrieve and render rich text [documentation](https://en.wikipedia.org/wiki/Application_programming_interface#Documentation) on functions, classes and methods automatically or on-demand
* A [debugger](https://en.wikipedia.org/wiki/Debugger) linked to IPdb, for step-by-step execution
* [Static code analysis](https://en.wikipedia.org/wiki/Static_program_analysis), powered by [Pylint](https://en.wikipedia.org/wiki/Pylint" \o "Pylint)
* A run-time [Profiler](https://en.wikipedia.org/wiki/Profiling_(computer_programming)), to benchmark code
* Project support, allowing work on multiple development efforts simultaneously
* A built-in [file explorer](https://en.wikipedia.org/wiki/File_manager), for interacting with the file system and managing projects
* A "Find in Files" feature, allowing full [regular expression](https://en.wikipedia.org/wiki/Regular_expression) search over a specified scope
* An online help browser, allowing users to search and view Python and package documentation inside the IDE
* A [history log](https://en.wikipedia.org/wiki/Command_history), recording every user command entered in each console
* An internal console, allowing for introspection and control over Spyder's own operation

**PLUGINS**

Available plugins include:

* Spyder-Unittest, which integrates the popular [unit testing](https://en.wikipedia.org/wiki/Unit_testing) frameworks Pytest, Unittest and Nose with Spyder
* Spyder-Notebook, allowing the viewing and editing of [Jupyter Notebooks](https://en.wikipedia.org/wiki/IPython" \l "Project_Jupyter" \o "IPython) within the IDE
* Download Spyder Notebook
* Using conda: *conda install spyder-notebook -c spyder-ide*
* Using pip: *pip install spyder-notebook*
* Spyder-Reports, enabling use of [literate programming](https://en.wikipedia.org/wiki/Literate_programming) techniques in Python
* Spyder-Terminal, adding the ability to open, control and manage cross-platform [system shells](https://en.wikipedia.org/wiki/Shell_(computing)) within Spyder
  + Download Spyder Terminal
  + Using conda: *conda install spyder-terminal -c spyder-ide*
  + Using pip: *pip install spyder-terminal*
* Spyder-Vim, containing commands and shortcuts emulating the [Vim text editor](https://en.wikipedia.org/wiki/Vim_(text_editor))
* Spyder-AutoPEP8, which can automatically conform code to the standard PEP 8 [code style](https://en.wikipedia.org/wiki/Programming_style)
* Spyder-Line-Profiler and Spyder-Memory-Profiler, extending the built-in profiling functionality to include testing an individual line, and measuring [memory](https://en.wikipedia.org/wiki/Random_access_memory) usage

**ANACONDA PYTHON**

Anaconda® is a package manager, an environment manager, a Python/R data science distribution, and a collection of [over 7,500+ open-source packages](https://docs.anaconda.com/anaconda/packages/pkg-docs/). Anaconda is free and easy to install, and it offers [free community support](https://groups.google.com/a/anaconda.com/forum/?fromgroups#!forum/anaconda).

Get the Anaconda Cheat Sheet and then [download Anaconda](https://www.anaconda.com/downloads).

Want to install conda and use conda to install just the packages you need? Get [Miniconda](http://conda.pydata.org/miniconda.html).

**Anaconda Navigator or conda?**

After you install Anaconda or Miniconda, if you prefer a desktop graphical user interface (GUI) then use [Navigator](https://docs.anaconda.com/anaconda/navigator/). If you prefer to use Anaconda prompt (or terminal on Linux or macOS), then use that and conda. You can also switch between them.

You can install, remove, or update any Anaconda package with a few clicks in Navigator, or with a single conda command in Anaconda Prompt (terminal on Linux or macOS).

* **To try Navigator**, after installing Anaconda, click the Navigator icon on your operating system’s program menu, or in Anaconda prompt (or terminal on Linux or macOS), run the command anaconda-navigator.
* **To try conda**, after installing Anaconda or Miniconda, take the [20-minute conda test drive](https://conda.io/projects/conda/en/latest/user-guide/getting-started.html) and download a [conda cheat sheet](https://docs.conda.io/projects/conda/en/latest/user-guide/cheatsheet.html).

**Packages available in Anaconda**

* Over [250 packages](https://docs.anaconda.com/anaconda/packages/pkg-docs/) are automatically installed with Anaconda.
* Over 7,500 additional open-source packages (including R) can be individually installed from the Anaconda repository with the conda install command.
* Thousands of other packages are available from [Anaconda.org](https://anaconda.org/).
* You can download other packages using the pip install command that is installed with Anaconda. [Pip packages](https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-pkgs.html#installing-non-conda-packages) provide many of the features of conda packages and in some cases they can work together. However, the preference should be to install the conda package if it is available.
* You can also make your own [custom packages](https://conda.io/projects/conda-build/en/latest/) using the conda build command, and you can share them with others by uploading them to [Anaconda.org](http://anaconda.org/), PyPI, or other repositories.

**Previous versions**

Previous versions of Anaconda are available in the [archive](https://repo.anaconda.com/archive/). For a list of packages included in each previous version, see [Old package lists](https://docs.anaconda.com/anaconda/packages/oldpkglists/).

Anaconda2 includes Python 2.7 and Anaconda3 includes Python 3.7. However, it does not matter which one you download, because you can create new environments that include any version of Python packaged with conda. See [Managing Python with conda](https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-python.html).

**Tkinter – Python**

Tk/Tcl has long been an integral part of Python. It provides a robust and platform independent windowing toolkit, that is available to Python programmers using the [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) package, and its extension, the [tkinter.tix](https://docs.python.org/3/library/tkinter.tix.html" \l "module-tkinter.tix" \o "tkinter.tix: Tk Extension Widgets for Tkinter) and the [tkinter.ttk](https://docs.python.org/3/library/tkinter.ttk.html" \l "module-tkinter.ttk" \o "tkinter.ttk: Tk themed widget set) modules.

The [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) package is a thin object-oriented layer on top of Tcl/Tk. To use [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces), you don’t need to write Tcl code, but you will need to consult the Tk documentation, and occasionally the Tcl documentation. [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) is a set of wrappers that implement the Tk widgets as Python classes.

[tkinter](https://docs.python.org/3/library/tkinter.html#module-tkinter)’s chief virtues are that it is fast, and that it usually comes bundled with Python. Although its standard documentation is weak, good material is available, which includes: references, tutorials, a book and others. [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) is also famous for having an outdated look and feel, which has been vastly improved in Tk 8.5. Nevertheless, there are many other GUI libraries that you could be interested in. The Python wiki lists several alternative [GUI frameworks and tools](https://wiki.python.org/moin/GuiProgramming).

**Main tkinter module.**

[tkinter.colorchooser](https://docs.python.org/3/library/tkinter.colorchooser.html" \l "module-tkinter.colorchooser" \o "tkinter.colorchooser: Color choosing dialog (Tk))

**Dialog to let the user choose a color.**

[tkinter.commondialog](https://docs.python.org/3/library/dialog.html#module-tkinter.commondialog)

**Base class for the dialogs defined in the other modules listed here.**

[tkinter.filedialog](https://docs.python.org/3/library/dialog.html#module-tkinter.filedialog)

**Common dialogs to allow the user to specify a file to open or save.**

[tkinter.font](https://docs.python.org/3/library/tkinter.font.html" \l "module-tkinter.font" \o "tkinter.font: Tkinter font-wrapping class (Tk))

**Utilities to help work with fonts.**

[tkinter.messagebox](https://docs.python.org/3/library/tkinter.messagebox.html#module-tkinter.messagebox)

**Access to standard tk dialog boxes.**

[tkinter.scrolledtext](https://docs.python.org/3/library/tkinter.scrolledtext.html#module-tkinter.scrolledtext)

**Text widget with a vertical scroll bar built in.**

[tkinter.simpledialog](https://docs.python.org/3/library/dialog.html#module-tkinter.simpledialog)

**Basic dialogs and convenience functions.**

[tkinter.ttk](https://docs.python.org/3/library/tkinter.ttk.html#module-tkinter.ttk)

Themed widget set introduced in Tk 8.5, providing modern alternatives for many of the classic widgets in the main [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) module.

Additional modules:

\_tkinter

A binary module that contains the low-level interface to Tcl/Tk. It is automatically imported by the main [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) module, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

idlelib

Python’s Integrated Development and Learning Environment (IDLE).Based on [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces).

tkinter.constants

Symbolic constants that can be used in place of strings when passing various parameters to Tkintercalls.Automatically imported by the main [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) module.

[tkinter.dnd](https://docs.python.org/3/library/tkinter.dnd.html#module-tkinter.dnd)

(experimental) Drag-and-drop support for [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces). This will become deprecated when it is replaced with the Tk DND.

[tkinter.tix](https://docs.python.org/3/library/tkinter.tix.html#module-tkinter.tix)

(deprecated) An older third-party Tcl/Tk package that adds several new widgets. Better alternatives for most can be found in [tkinter.ttk](https://docs.python.org/3/library/tkinter.ttk.html" \l "module-tkinter.ttk" \o "tkinter.ttk: Tk themed widget set).

[turtle](https://docs.python.org/3/library/turtle.html#module-turtle)

Turtle graphics in a Tk window.

**CHAPTER 5: TESTING**

**SYSTEM TESTING AND MAINTENANCE:**

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. In the testing process we test the actual system in an organization and gather errors from the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed to ensuring that the system works accurately and efficiently.

In the testing process we test the actual system in an organization and gather errors from the new system and take initiatives to correct the same. All the front-end and back-end connectivity are tested to be sure that the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently.

The main objective of testing is to uncover errors from the system. For the uncovering process we have to give proper input data to the system. So we should have more conscious to give input data. It is important to give correct inputs to efficient testing.

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works. Inadequate testing or non-testing leads to errors that may appear few months later.

This will create two problems, Time delay between the cause and appearance of the problem. The effect of the system errors on files and records within the system. The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the system to its limits.

The testing process focuses on logical intervals of the software ensuring that all the statements have been tested and on the function intervals (i.e.,) conducting tests to uncover errors and ensure that defined inputs will produce actual results that agree with the required results. Testing has to be done using the two common steps Unit testing and Integration testing. In the project system testing is made as follows:

The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. This is the final step in system life cycle. Here we implement the tested error-free system into real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every months or year based on company policies, and is checked for errors like runtime errors, long run errors and other maintenances like table verification and reports.

Integration Testing is a level of software testing where individual units are combined and tested as a group.

The purpose of this level is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration testing.

**METHOD**

Any of Black Box Testing, White Box Testing, and Gray Box Testing methods can be used. Normally, the method depends on your definition of ‘unit’.

TASKS

* + Integration Test Plan
  + Prepare
  + Review
  + Rework
  + Baseline
  + Integration Test Cases/Scripts
  + Prepare
  + Review
  + Rework
  + Baseline
  + Integration Test
  + Perform

**UNIT TESTING**:

Unit testing verification efforts on the smallest unit of software design, module. This is known as “Module Testing”. The modules are tested separately. This testing is carried out during programming stage itself. In these testing steps, each module is found to be working satisfactorily as regard to the expected output from the module.

**BLACK BOX TESTING**

Black box testing, also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.

**WHITE-BOX TESTING**

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing).

**GREY BOX TESTING**

Grey box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. To test the Web Services application usually the Grey box testing is used. Grey box testing is performed by end-users and also by testers and developers.

**INTEGRATION TESTING:**

Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps.

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance testing for Data Synchronization:**

 The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node

 The Route add operation is done only when there is a Route request in need

 The Status of Nodes information is done automatically in the Cache Updating process

**BUILD THE TEST PLAN**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**SCOPE OF PROJECT :**

* Phishing is the most commonly used social engineering and cyber attack. Through such attacks, the phisher targets naïve online users by tricking them into revealing confidential information, with the purpose of using it fraudulently. In order to avoid getting phished, users should have awareness of phishing websites. Have a blacklist of phishing websites which requires the knowledge of website being detected as phishing. Detect them in their early appearance, using machine learning and deep neural network algorithms of the above three, the machine learning based method is proven to be most effective than the other methods. Even then, online users are still being trapped into revealing sensitive information in phishing websites. A phishing website is a common social engineering method that mimics trustful uniform resource locators (URLs) and webpages.
* The objective of this project is to train machine learning models and deep neural nets on the dataset created to predict phishing websites. Both phishing and benign URLs of websites are gathered to form a dataset and from them required URL and website content-based features are extracted.
* The performance level of each model is measures and compared. The phishing website has evolved as a major cybersecurity threat in recent times. The phishing websites host spam, malware, ransomware, drive-by exploits, etc. A phishing website many a time look-alike a very popular website and lure an unsuspecting user to fall victim to the trap. The victim of the scams incurs a monetary loss, loss of private information and loss of reputation.
* Hence, it is imperative to find a solution that could mitigate such security threats in a timely manner. Traditionally, the detection of phishing websites is done using blacklists. There are many popular websites which host a list of blacklisted websites.

**CONCLUSION :**

A machine learning (ML) based phishing attack detection was proposed in this paper. The investigation utilizes many strategies to identify phishing detection. Standard datasets of phishing attacks from kaggle.com were used as input for the ML algorithms. Two popular machine learning algorithms, namely decision tree and random forest, are implemented to analyze and select datasets for classification and detection. Principal component analysis (PCA) was applied to identify and classify the components of the datasets. DT was used to classify the website and RF was used for classification. Finally, the confusion matrix was drawn to evaluate the performance of these two algorithms. RF had less variance, and it could handle the over-fitting problem. The random forest tree achieved an accuracy of 97%. In our future work, fishing attacks will be predicted from the logged dataset of attacks by using a convolution neural network (CNN). It will be added as a tool for intrusion detection system (IDS).

**REFERENCE:**

[1]S. Mahdavifar and A. A. Ghorbani, "DeNNeS: deep embedded neural network expert system for detecting cyber attacks," (in English), Neural Computing & Applications, Article; Early Access p. 28.

[2] S. Mishra and D. Soni, "Smishing Detector: A security model to detect smishing through SMS content analysis and URL behavior analysis," (in English), Future Generation Computer Systems-the International Journal of Escience, Article vol. 108, pp. 803-815, Jul 2020.

[3] K. S. Adewole, T. Hang, W. Q. Wu, H. B. Songs, and A. K. Sangaiah, "Twitter spam account detection based on clustering and classification methods," (in English), Journal of Supercomputing, Article vol. 76, no. 7, pp. 4802-4837, Jul 2020.

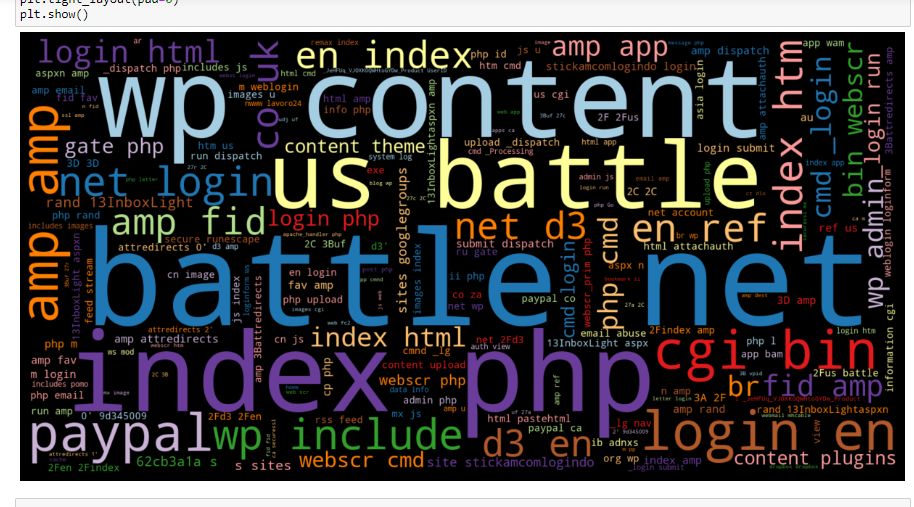
[4] B. A. Eduardo, F. D. Walter, and S. G. Sandra, "An Experiment to Create Awareness in People concerning Social Engineering Attacks," (in Spanish), Ciencia Unemi, Article vol. 13, no. 32, pp. 27-40, Jan-Apr 2020.

[5] S. Farrugia, J. Ellul, and G. Azzopardi, "Detection of illicit accounts over the Ethereum blockchain," (in English), Expert Systems with Applications, Article vol. 150, p. 11, Jul 2020, Art. no. 113318.

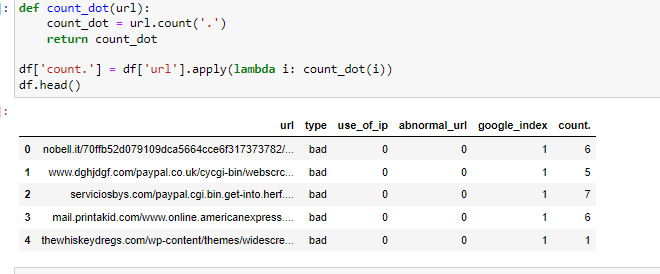
[6] C. D. Xuan, H. D. Nguyen, and T. V. Nikolaevich, "Malicious URL Detection based on Machine Learning," (in English), International Journal of Advanced Computer Science and Applications, Article vol. 11, no. 1, pp. 148-153, Jan 2020. [7] J. C. Gonzalez, V. Garcia-Diaz, E. R. Nunez-Valdez, A. G. Gomez, and R. G. Crespo, "Replacing email protocols with blockchain-based smart contracts," (in English), Cluster Computing-the Journal of Networks Software Tools and Applications, Article; Early Access p.

SCREENSHOTS:

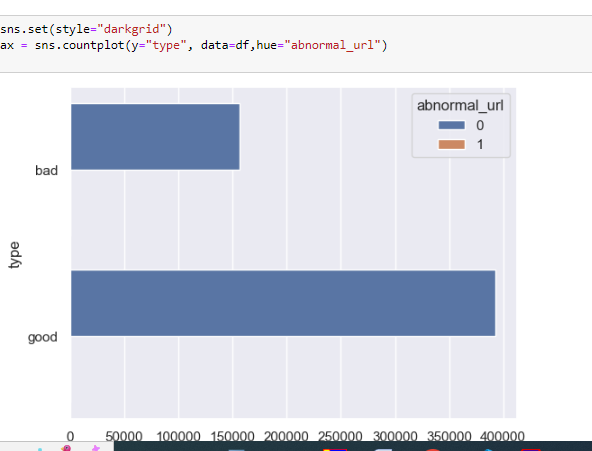
GRAPHS:

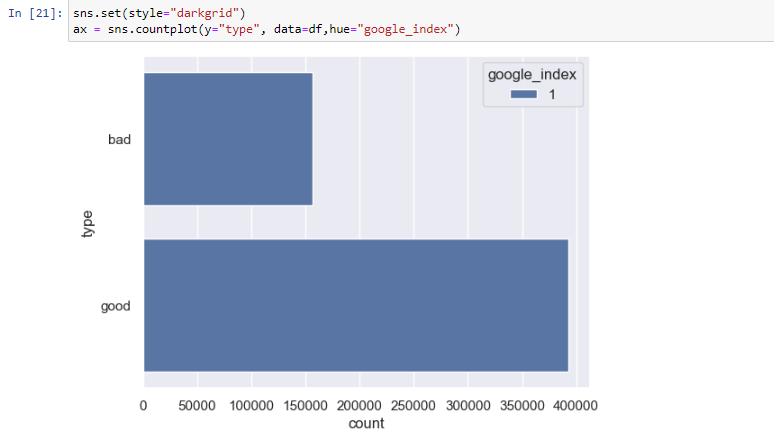


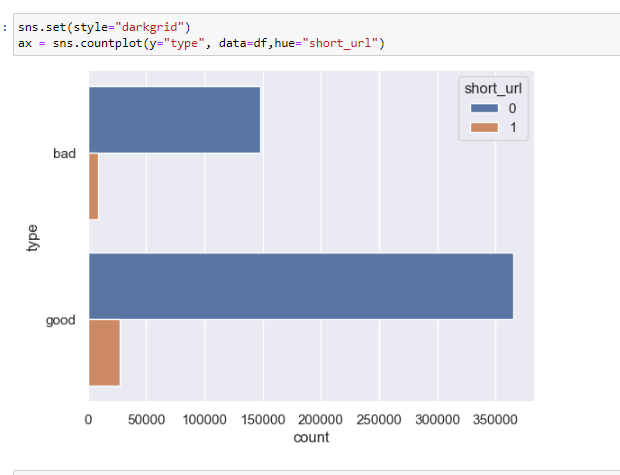


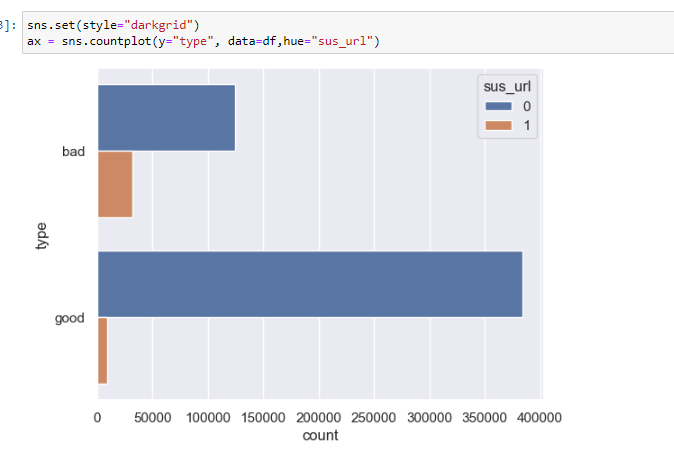


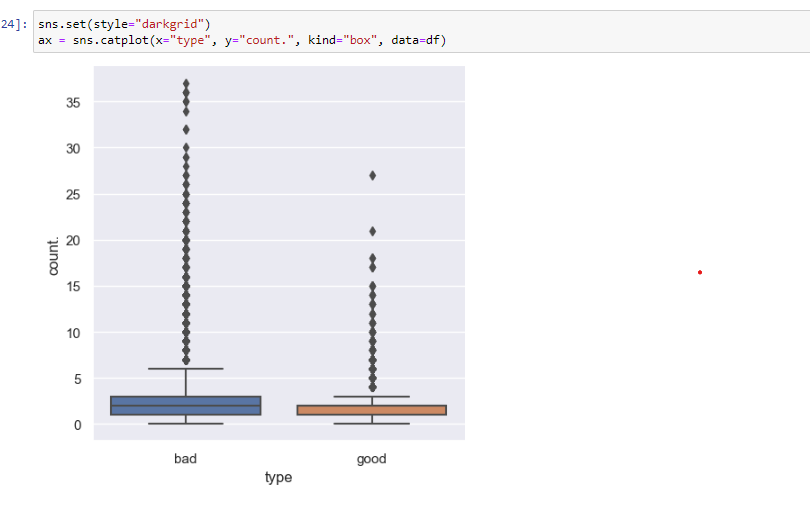




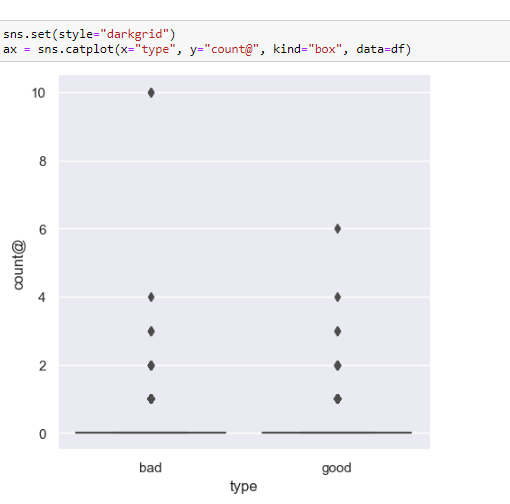


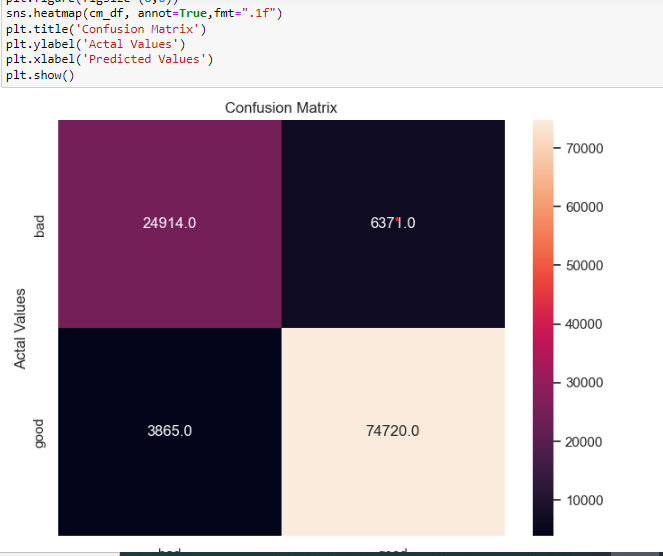












SOURCE CODE:

from flask import Flask, render\_template, request, redirect, url\_for,flash,session

import numpy as np

import pandas as pd

from sklearn import metrics

import warnings

import pickle

warnings.filterwarnings('ignore')

from feature import FeatureExtraction

import time

import mysql.connector

conn=mysql.connector.connect(host="localhost",user="root",password="root",autocommit=True)

mycursor=conn.cursor(dictionary=True,buffered=True)

mycursor.execute("create database if not exists fakewebsite")

mycursor.execute("use fakewebsite")

mycursor.execute("create table if not exists fake(id int primary key auto\_increment,cname varchar(255),email varchar(30) unique,cpassword text)")

file = open("pickle/model.pkl","rb")

gbc = pickle.load(file)

file.close()

app = Flask(\_\_name\_\_)

app.secret\_key = 'super secret key'

UPLOAD\_FOLDER = 'static'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

app.config['MAX\_CONTENT\_LENGTH'] = 16 \* 1024 \* 1024

@app.route('/')

def home():

    return render\_template('index.html')

@app.route('/registration',methods =['GET', 'POST'])

def registration():

  if request.method == 'POST' and 'pass' in request.form and 'email' in request.form and 'hos' in request.form:

        name = request.form.get('pass')

        password=request.form.get('hos')

        mob = request.form.get('mob')

        email = request.form.get('email')

        mycursor.execute("SELECT \* FROM fake WHERE email = '"+ email +"' ")

        account = mycursor.fetchone()

        if account:

            flash('You are already registered, please log in')

        else:

            mycursor.execute("insert into fake values(NULL,'"+ name +"','"+ email +"','"+ password +"')")

            # msg=flash('You have successfully registered !')

            return render\_template("login.html")

  return render\_template("register.html")

@app.route('/login',methods =['GET', 'POST'])

def login():

    if request.method == 'POST' and 'nm' in request.form and 'pass' in request.form:

        print('hello')

        email = request.form['nm']

        password = request.form['pass']

        mycursor.execute("SELECT \* FROM fake WHERE email = '"+ email +"' AND cpassword = '"+ password +"'")

        account = mycursor.fetchone()

        print(account)

        if account:

            session['loggedin'] = True

            session['email'] = account['email']

            msg = flash('Logged in successfully !')

            return render\_template('predict.html')

        else:

            msg = flash('Incorrect username / password !')

            return render\_template('login.html',msg=msg)

    return render\_template('login.html')

import random

@app.route("/index", methods=["GET", "POST"])

def index():

    virus = {0: 'Spear Phishing', 1: 'Whaling', 2: 'Smishing', 3: 'Vishing'}

    if request.method == "POST":

        url = request.form["url"]

        obj = FeatureExtraction(url)

        x = np.array(obj.getFeaturesList()).reshape(1, 30)

        start\_time = time.time()  # Record start time

        y\_pred = gbc.predict(x)[0]

        print(y\_pred)

        end\_time = time.time()  # Record end time

        prediction\_time = end\_time - start\_time  # Calculate time taken for prediction

        prediction\_time\_formatted = "{:.3f}".format(prediction\_time)

        y\_pro\_phishing = gbc.predict\_proba(x)[0, 0]

        y\_pro\_non\_phishing = gbc.predict\_proba(x)[0, 1]

        pred = "It is {0:.2f} % safe to go ".format(y\_pro\_phishing \* 100)

        if y\_pred == -1:  # Check if predicted vulnerability exists

            virus\_name = random.choice(list(virus.values()))

        else:

            virus\_name = None

        return render\_template('predict.html',

                               xx=round(y\_pro\_non\_phishing, 2),

                               url=url,

                               y\_pred=y\_pred,

                               prediction\_time=prediction\_time\_formatted,

                               virus\_name=virus\_name)

    return render\_template("predict.html", xx=-1)

if \_\_name\_\_ == "\_\_main\_\_":

    app.run(debug=True)

OUTPUT:



