**DETECTING PHISHING WEBSITES USING MACHINE LEARNING**



A major project report submitted in partial fulfillment of requirements for the award of Degree of

**BACHELOR OF TECHNOLOGY IN**

**ELECTORNICS AND COMMUNICATION ENGINEERING**

**By**

**N. YASHOVARDHAN (21P15A0408**)

**Under the Esteemed guidance of Amejoy exson**

**Department of Electronics and Communication Engineering**

**CHADALAWADA RAMANAMMA ENGINEERING COLLEGE**

**(AUTONOMOUS)**

(Accredited by NAAC, Approved by AICTE, New Delhi, Affiliated to JNTU Anantapur)

Renigunta road, Tirupati – 517 506, Andhra Pradesh, India

**2020 - 2024**

**Department of Electronics and Communication Engineering**

**CHADALAWADA RAMANAMMA ENGINEERING COLLEGE**

**(AUTONOMOUS)**

(Accredited by NAAC, Approved by AICTE, New Delhi, Affiliated to JNTU Anantapur)

Renigunta road, Tirupati – 517 506, Andhra Pradesh, India

**2020 - 2024**



**CERTIFICATE**

*This is to certify that the major project work entitled* ‘DETECTING PHISHING WEBSITES USING MACHINE LEARNING’ *is a*

*bona fide record of work carried out by*

#### N.YASHOVARDHAN (21P11A0408)

**BACHELOR OF TECHNOLOGY IN**

**ELECTORNICS COMMUNICATION & ENGINEERING**

### DECLARATION

I hereby declare that the project titled “DETECTING PHISHING WEBSITES USING MACHINE LEARNING” is the authentic work carried out by me as the student of CHADALAWADA RAMANAMMA ENGINEERING COLLEGE (Autonomous): Tirupati, during March – June 2021 and has not been submitted elsewhere for the award of degree in part or in full to any institute.

**N. YASHOVARDHAN**

**(21P15A0408)**

# ACKNOWLEDGEMET

We are very much thankful to **Dr. CHADALAWADA KRISHNA MURTHY**, **Chairman**, C.V.S. Krishna Murthy Teja Charities for providing education in this esteemed Institution.

I am highly indebted to Principal **Dr. P. RAMESH KUMAR** for his providing the opportunity to do my internship course and others.

I am very much thankful to Dean (Academics) **Dr. C. SUBHAS** for his continuous support in academics.

I would like to extend my thanks to our Head of the Department **Dr. P. KRISHNA MURTHY** for his constructive criticism throughout my internship.

I would like thank my guide **Dr. M. VIJAYA LAXMI** for her guidance and support.

I would like to thank the Director **Slash Mark IT Startupindia**  of **Mr. MUKESH RAJ** for his allowing me to do an internship within the organization.

I would also like all the people who worked along with me in **Slash Mark IT Startupindia** with their patience and openness created an enjoyable working environment.

I am extremely grateful to my parents, department staff members, and friends who helped me in the successful completion of this internship.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

**(NALLABOINA YASHOVARDHAN)**

**Regd. No.: 21P15A0408**

**CONTENTS**

1. [INTRODUCTION. 1](#_TOC_250028)
   1. [Introduction 2](#_TOC_250027)
   2. [Motivation. 3](#_TOC_250026)
   3. [Problem Definition… 4](#_TOC_250025)
   4. Objective the Project 4
   5. [Limitations of the Project 4](#_TOC_250024)
   6. [Organization of the report… 5](#_TOC_250023)
2. [SYSTEM SPECIFICATIONS 6](#_TOC_250022)
   1. [Software Requirements 7](#_TOC_250021)
   2. [Hardware Requirements 22](#_TOC_250020)
3. [LITERATURE SURVEY 23](#_TOC_250019)
   1. [Existing System. 24](#_TOC_250018)
   2. [Disadvantages of Existing System. 24](#_TOC_250017)
   3. [Proposed System. 25](#_TOC_250016)
4. [ANALYSIS 26](#_TOC_250015)
   1. [Module Organization. 27](#_TOC_250014)
   2. [Feasibility Study. 28](#_TOC_250013)
5. [DESIGN. 30](#_TOC_250012)
   1. [Input and Output Design. 31](#_TOC_250011)
   2. [UML Diagrams 33](#_TOC_250010)
6. [IMPLEMENTATION AND RESULT ANALYSIS 35](#_TOC_250009)
   1. [Introduction 36](#_TOC_250008)
   2. [Method of Implementation 37](#_TOC_250007)
   3. [Source Code 50](#_TOC_250006)
   4. [Output Screens. 58](#_TOC_250005)
7. [TESTING AND VALIDATIONS 64](#_TOC_250004)
   1. [Introduction 65](#_TOC_250003)
   2. [System Testing… 66](#_TOC_250002)
   3. [Levels of Testing 67](#_TOC_250001)
8. CONCLUSION AND FUTURE ENHANCEMENTS 69

[REFERENCES 71](#_TOC_250000)

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **S.No**  1. | **Figure No.**  Fig.2.1.2.1.1 | **Title Page No.**  Anaconda Website 9 |
| 2. | Fig.2.1.2.1.2 | Downloading Anaconda 9 |
| 3. | Fig.2.1.2.1.3 | Welcome To Anaconda Setup 10 |
| 4. | Fig.2.1.2.1.4 | Anaconda License Agreement 11 |
| 5. | Fig.2.1.2.1.5 | Selection Of Installation Type 11 |
| 6. | Fig.2.1.2.1.6 | Installation Location 12 |
| 7. | Fig.2.1.2.1.7 | Installation Options 13 |
| 8. | Fig.2.1.2.1.8 | Installation Complete 13 |
| 9. | Fig.2.1.2.1.9 | Visual Studio Code Installation 14 |
| 10. | Fig.2.1.2.1.10 | Installation Complete 14 |
| 11. | Fig.2.1.2.2.1 | Opening Of Command Prompt 15 |
| 12. | Fig.2.1.2.2.2 | Checking Of Anaconda and 16  Python Installation |
| 13. | Fig.2.1.2.2.3 | Knowing Location Of Python 16  and Anaconda |
| 14. | Fig.2.1.2.2.4 | System And Environment Variables 17 |
| 15. | Fig.2.1.2.2.5 | Checking Of Python And Anaconda 18 |
| 16. | Fig.2.1.3.1.1 | Checking Jupyter Notebook Installation 19 |
| 17. | Fig.2.1.3.2.1 | Command Prompt After Successful 20  Installation |
| 18. | Fig.2.1.3.2.2 | Jupyter Notebook in Browser 20 |
| 19. | Fig.2.1.3.2.3 | Opening Jupyter Notebook Using 21  Anaconda |
| 20. | Fig.4.1.1 | Block Diagram of Phishing Website 27  Detection System |
| 21. | Fig.5.2.1 | System Architecture Phishing Detection 33 |
| 22. | Fig.3.2.1.1 | System  Supervised Learning 37 |
| 23. | Fig.6.2.2.1 | Decision Tree Diagram 39 |
| 24. | Fig.6.2.3.1 | Random Forest Classifier 40 |

|  |  |  |
| --- | --- | --- |
| 25. Fig.6.2.4.1 | XGBoost Classifier | 42 |
| 26. Fig.6.2.4.2 | Multilayer Perceptron Model | 44 |
| 27. Fig.6.2.5.1 | SVM Hyperplanes | 46 |
| 28. Fig.6.2.5.2 | Hyperplanes in 2D and 3D Feature space | 47 |

|  |  |  |  |
| --- | --- | --- | --- |
| 29. | Fig.6.2.5.3 | Support Vectors | 47 |
| 30. | Fig.6.2.5.4 | Hinge Loss Function | 48 |
| 31. | Fig.6.2.5.5 | Loss Function Of SVM | 49 |
| 32. | Fig.6.2.5.6 | Gradients | 49 |
| 33. | Fig.6.2.5.7 | Gradient Update-No Misclassification | 49 |
| 34. | Fig.6.2.5.8 | Gradient Update-Misclassification | 49 |
| 35. | Fig.6.4.1 | Loading and displaying Dataset | 58 |
| 36. | Fig.6.4.2 | Checking Shape & Listing Features | 58 |
| 37. | Fig.6.4.3 | Listing Columns of Dataset | 59 |
| 38. | Fig.6.4.4 | Histogram Visualization of Dataset | 59 |
| 39. | Fig.6.4.5 | Correlation Heatmap of Dataset | 60 |
| 40. | Fig.6.4.6 | Splitting Dataset into Train and Test Data | 60 |
| 41. | Fig.6.4.7 | Accuracy Score & Feature Importance | 61 |
|  |  | of Decision Tree |  |
| 42. | Fig.6.4.8 | Accuracy Score & Feature Importance | 61 |
|  |  | of Random Forest |  |
| 43. | Fig 6.4.9 | Accuracy Score of Multilayer Perceptron | 62 |
|  |  | Model |  |
| 44. | Fig 6.4.10 | Accuracy Score of XGBoost Classifier | 62 |
| 45. | Fig 6.4.11 | Accuracy Score of SVM | 62 |
| 46. | Fig 6.4.12 | Data Frame of All Classifiers Along | 63 |
|  |  | With Accuracy Rates |  |

**ABSTRACT**

Phishing is the technique of extracting user credentials and sensitive data from users by masquerading as a genuine website. In phishing, the user is provided with a mirror website which is identical to the legitimate one but with malicious code to extract and send user credentials to phishes. Phishing attacks can lead to huge financial losses for customers of banking and financial services. The traditional approach to phishing detection has been to either to use a blacklist of known phishing links or heuristically evaluate the attributes in a suspected phishing page to detect the presence of malicious codes. The heuristic function relies on trial and error to define the threshold which is used to classify malicious links from benign ones. The drawback to this approach is poor accuracy and low adaptability to new phishing links. To overcome these drawbacks the basic idea is to use machine learning algorithms on available dataset of phishing pages to generate a model which can be used to make classifications in real time if a given web page is a phishing page or a legitimate webpage. This system will test algorithms such as Decision Tree Classifier, SVM, Random Forest, Multilayer perceptron and XGBoost on a dataset of phishing dataset.

v

# INTRODUCTION

### INTRODUCTION

Phishing is an act which combines social engineering and technical expertise to obtain the information of the users in an illegitimate way. Phishing can happen through emails, websites and malware. The “phishing” term is coined based on the fact that web users get enticed to fish their own sensitive information from sea of Internet by using websites and email. It was first mentioned in hacker newsletter 2600 on January 1996[1]. In 1996, the term phishing was used to mention the incidents in which hackers exploited the passwords to steal AOL accounts. In certain scenario an attacker makes the victim user believe that he/she is a AOL staff member, who sends the mail or message to the victim, typically meant to extract/reveal the password of the potential victim. In order for the mail or message to appear as genuine to the victim, the phisher includes the phrases such as “Verify Account” or “your subscription will be cancelled” which makes the victim access that particular link [2]. Once the link is accessed, the phisher tries to steal the sensitive information such as passwords for fraudulent purposes. In today’s world, phishing has been used in broader impact to describe attacks which target the sensitive information of the online users.

#### INTRODUCTION

Phishing is the technique of extracting user credentials and sensitive data from users

by masquerading as a genuine website. In phishing, the user is provided with a mirror website which is identical to the legitimate one but with malicious code to extract and send user credentials to phishers. Phishing attacks can lead to huge financial losses for customers of banking and financial services. The traditional approach to phishing detection has been to either to use a blacklist of known phishing links or heuristically evaluate the attributes in a suspected phishing page to detect the presence of malicious codes. The heuristic function relies on trial and error to define the threshold which is used to classify malicious links from benign ones. The drawback to this approach is poor accuracy and low adaptability to new phishing links. We plan to use machine learning to overcome these drawbacks by implementing some classification algorithms and comparing the performance of these algorithms on our dataset. We will test algorithms such as Logistic Regression, SVM, Decision Trees and Neural Networks on a dataset of phishing links from UCI Machine Learning repository and pick the best model to develop a browser plugin, which can be published as a chrome extension. In today’s world, technology has become an integral part of the twenty-first century. The internet is

one of these technologies, which is growing rapidly every year and plays an important role in individuals’ lives. It has become a valuable and a convenient mechanism for supporting public transactions such as e-banking and e-commerce transactions. That has led the users to trust it is convenient to provide their private information to the Internet. As a result, the security thieves that have started to target this information havebecome a major security problem. Phishing websites are considered to be one of these problems. They are using a social engineering trick, which can be described as fraudsters that try to manipulate the user into giving them their personal information based on exploiting human vulnerabilities rather than software vulnerabilities.

#### MOTIVATION

In the early days of “intelligent” applications, many systems used hand coded rules of “if ” and “else” decisions to process data or adjust to user input. Think of a spam filter whose job is to move the appropriate incoming email messages to a spam folder. You could make up a blacklist of words that would result in an email being marked as spam. This would be an example of using an expert-designed rule system to design an “intelligent” application. Manually crafting decision rules is feasible for some applications, particularly those in which humans have a good understanding of the process to model. However, using hand coded rules to make decisions has two major disadvantages are the logic required to make a decision is specific to a single domain and task. Changing the task even slightly might require a rewrite of the whole system. Designing rules requires a deep understanding of how a decision should be made by a human expert. One example of where this hand coded approach will fail is in detecting faces in images. Today, every smart phone can detect a face in an image. However, face detection was an unsolved problem until as recently as 2001. The main problem is that the way in which pixels (which make up an image in a computer) are “perceived” by the computer is very different from how humans perceive a face. This difference in representation makes it basically impossible for a human to come up with a good set of rules to describe what constitutes a face in a digital image. Using machine learning, however, simply presenting a program with a large collection of images of faces is enough for an algorithm to determine what characteristics are needed to identify a face.

#### PROBLEM DEFINITION

Phishing has been a major security threat in which there is a huge loss for companies as well as customers. These phishing attacks are increasing day by day due to lack of efficient detection techniques and effective preventive measures. A comprehensive efficient detection technique should be developed in order to detect and inform the web users about the phishing attacks to make sure that their sensitive data will not be disclosed during these attacks. As there are many phishing detection techniques proposed by the researchers in past few years, most of the techniques doesn’t detect more than 60% of the phishing attacks [15]. Also, there are no such web applications based on phishing detection which classify the websites based on features of the URL using several heuristics. This research project deals with a comprehensive heuristic based method for phishing detection which is based on content of the website through which phishing attacks can be discovered.

#### OBJECTIVE OF THE PROJECT

The goal of this project is to develop a heuristic based phishing detection system through which a user can determine whether a URL is a phished, suspicious or legitimate one. Using several heuristics, the features of the URL will be extracted and examined in classifying the website based on the results obtained.

The following are the main objectives of this project:

1. To provide knowledge and insight to the inexperienced web users in identifying the phishing URLs.
2. To give in-depth information about the website features which are used in predicting the phishing attacks.
3. To categorize the website features into several classifiers which contribute to the final prediction of a given website
4. To evaluate the classification accuracy of a phishing dataset using Confusion Matrix.

#### LIMITATIONS OF THE PROJECT

If Internet connection fails, this system won’t work Feature selection is difficult in url.

Accuracy is not hundred percent. It does not detect in all cases . Highest accuracy is 85%

hence it can detect only 85 phishing websites from hundred websites selection of algorithm is difficult. It does not work if the internet connection fails.

#### ORGANIZATION OF THE REPORT

The first chapter deals with the introduction of thePhishing attack,Phishing website detection system, motivation for developing this project, problem definition, objective of the project. The second chapter deals with the system specifications required for developing the project. It includes hardware and software specifications. The third chapter gives you the preview about literature survey which includes the information about existing system, disadvantages of existing system and proposed system. The fourth chapter is for analysis of the project which includes data and requirement analysis, module organization. The fifth chapter deals with the design of the project which includes data flow diagrams and UML diagrams. The sixth chapter deals with the implementation of the project which includes module definition and result analysis of the project. The seventh chapter tells about the testing and validations. Finally the eighth chapter deals with the conclusion and future enhancement.

# SYSTEM SPECIFICATIONS

### SYSTEM SPECIFICATIONS

#### SOFTWARE REQUIREMENTS :

|  |  |  |
| --- | --- | --- |
|  | **Operating system :** | Windows |
|  | **Technology :** | Python |
|  | **IDE :** | Anaconda IDE – Jupyter Note Book |
| **2.1.1** | **PYTHON** |  |

**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language) [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language](https://en.wikipedia.org/wiki/Language_construct) [constructs](https://en.wikipedia.org/wiki/Language_construct) as well as its [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects.

Python is [dynamically-typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming](https://en.wikipedia.org/wiki/Programming_paradigms) [paradigms,](https://en.wikipedia.org/wiki/Programming_paradigms) which including [structured](https://en.wikipedia.org/wiki/Structured_programming) (particularly, [procedural](https://en.wikipedia.org/wiki/Procedural_programming)), object-oriented and [functional](https://en.wikipedia.org/wiki/Functional_programming) [programming.](https://en.wikipedia.org/wiki/Functional_programming) Python is often described as a "batteries included" language due to its comprehensive [standard library.](https://en.wikipedia.org/wiki/Standard_library)

[Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) began working on Python in the late 1980s, as a successor to the [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)) [programming language,](https://en.wikipedia.org/wiki/ABC_(programming_language)) and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features, such as [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension) and a garbage collection system using [reference counting](https://en.wikipedia.org/wiki/Reference_counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely [backward-compatible](https://en.wikipedia.org/wiki/Backward_compatibility) and much Python 2 code does not run unmodified on Python 3. Python 2 was discontinued with version 2.7.18 in 2020.

#### ANACONDA

**Anaconda** is a [distribution](https://en.wikipedia.org/wiki/Software_distribution) of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)) programming languages for [scientific](https://en.wikipedia.org/wiki/Scientific_computing) [computing](https://en.wikipedia.org/wiki/Scientific_computing) ([data science,](https://en.wikipedia.org/wiki/Data_science) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications, large-scale data processing, [predictive](https://en.wikipedia.org/wiki/Predictive_analytics) [analytics,](https://en.wikipedia.org/wiki/Predictive_analytics) etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by

Anaconda, Inc., which was founded by Peter Wang and [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) in 2012. As an Anaconda, Inc. product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free.

Anaconda is a package manager, an environment manager, and Python distribution that contains a collection of many open source packages. This is advantageous as when you are working on a data science project, you will find that you need many different packages (numpy, scikit-learn, scipy, pandas to name a few), which an installation of Anaconda comes preinstalled with. If you need additional packages after installing Anaconda, you can use Anaconda's package manager, conda, or pip to install those packages. This is highly advantageous as you don't have to manage dependencies between multiple packages yourself. Conda even makes it easy to switch between Python 2 and 3 (you can learn more [here](https://towardsdatascience.com/environment-management-with-conda-python-2-3-b9961a8a5097)). In fact, an installation of Anaconda is also the [recommended way to install](http://jupyter.org/install.html) [Jupyter Notebooks](http://jupyter.org/install.html) which you can learn more about [here](https://www.datacamp.com/community/tutorials/tutorial-jupyter-notebook) on the DataCamp community.

Package versions in Anaconda are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager) [conda](https://en.wikipedia.org/wiki/Conda_(package_manager)). This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) as well as the [conda](https://en.wikipedia.org/wiki/Conda_(package_manager)) package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

#### INSTALLING ANACONDA ON WINDOWS

Step 1 : Go to the [Anaconda Website](https://www.anaconda.com/download/#windows) and choose a Python 3.x graphical installer (A) or a Python 2.x graphical installer (B).

Do not choose both.

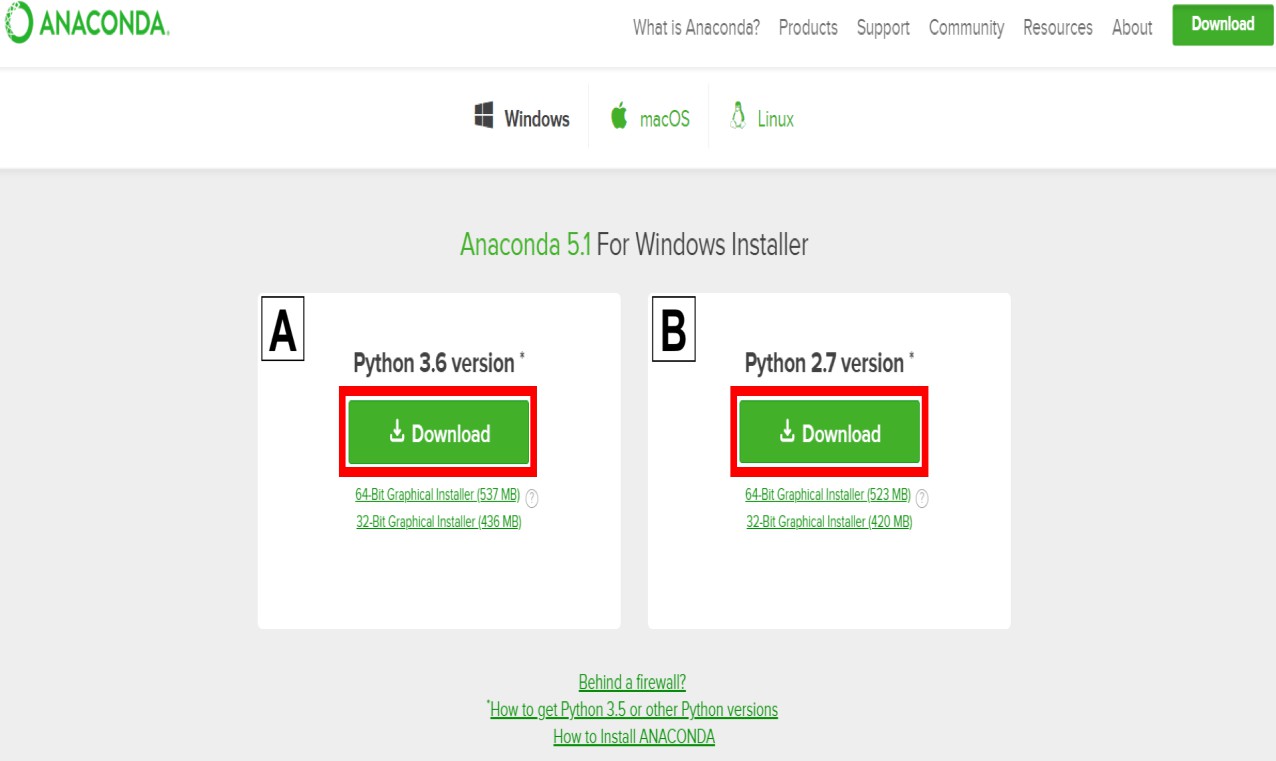


Fig 2.1.2.1.1 Anaconda Website

Step 2 : Locate your download and double click it.

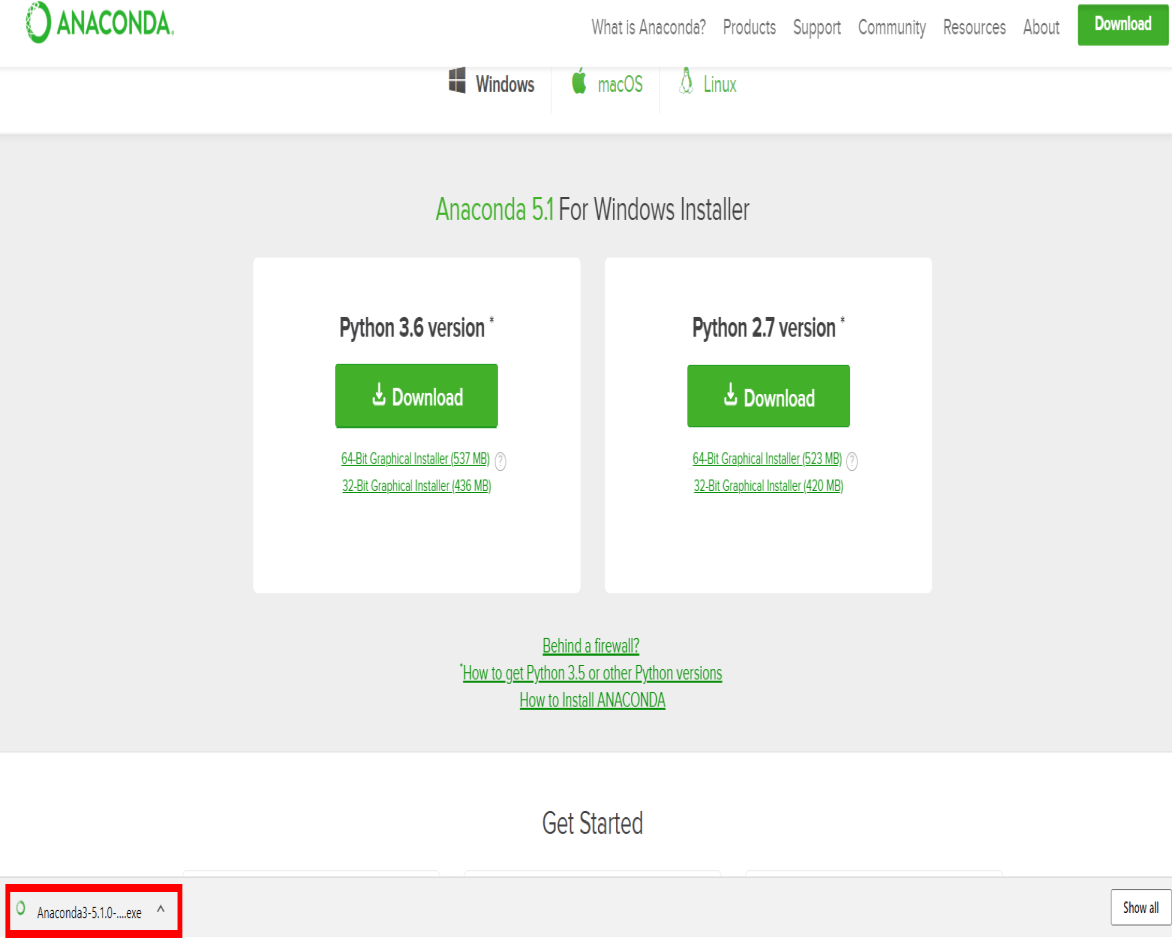


Fig 2.1.2.1.2 Downloading Anaconda

When the screen below appears, click on Next.

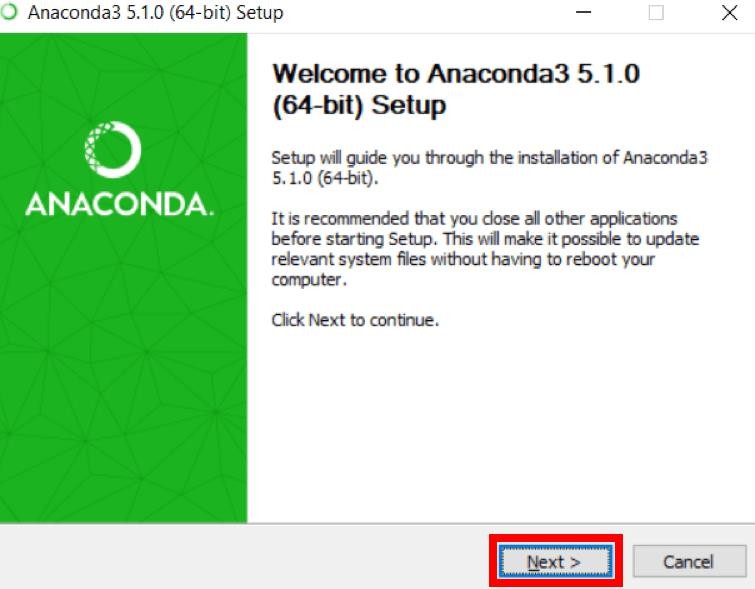


Fig 2.1.2.1.3 Welcome To Anaconda Setup

Step 3 : Read the license agreement and click on I Agree.

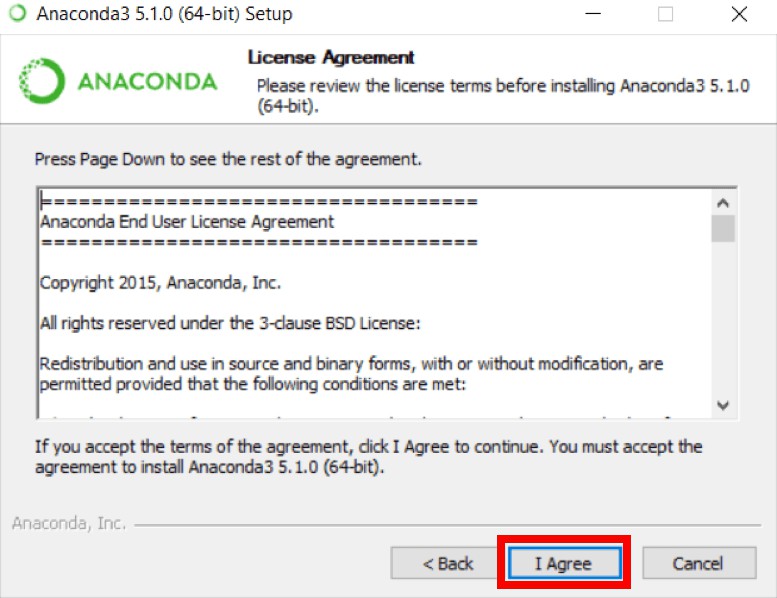


Fig 2.1.2.1.4 Anaconda License Agreement

Step 4 : Click on Next.

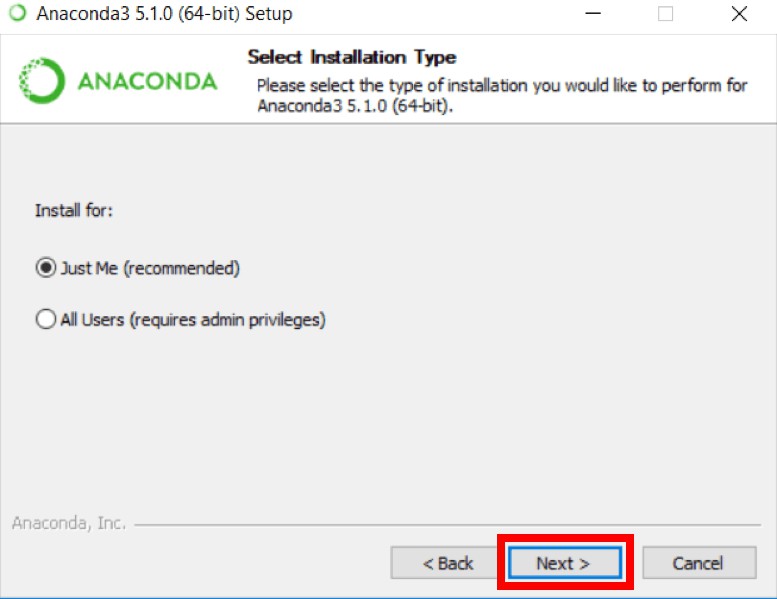


Fig 2.1.2.1.5 Selection of Installation Type

Step 5 : Note your installation location and then click Next.

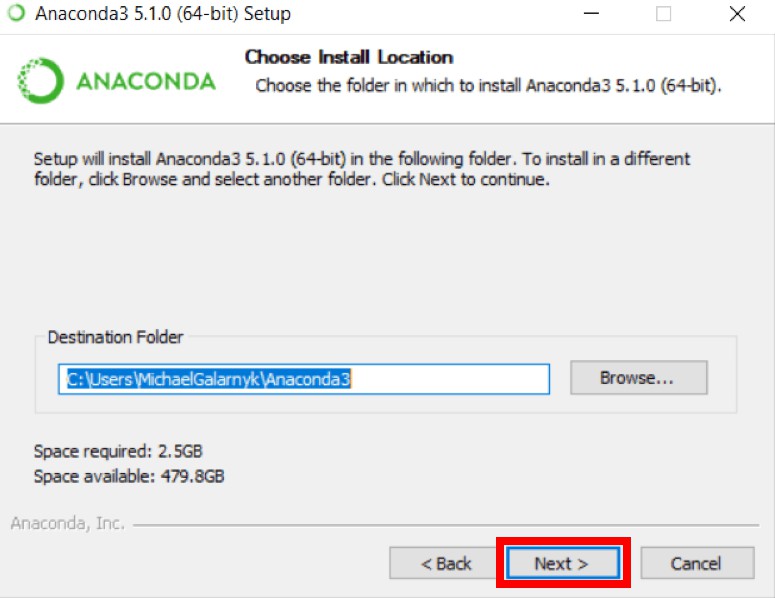


Fig 2.1.2.1.6 Installation Location

Step 6 : This is an important part of the installation process. The recommended approach is to not check the box to add Anaconda to your path. This means you will have to use Anaconda Navigator or the Anaconda Command Prompt (located in the Start Menu under "Anaconda") when you wish to use Anaconda (you can always add Anaconda to your PATH later if you don't check the box). If you want to be able to use Anaconda in your command prompt (or git bash, [cmder,](http://cmder.net/) powershell etc), please use the alternative approach and check the box.

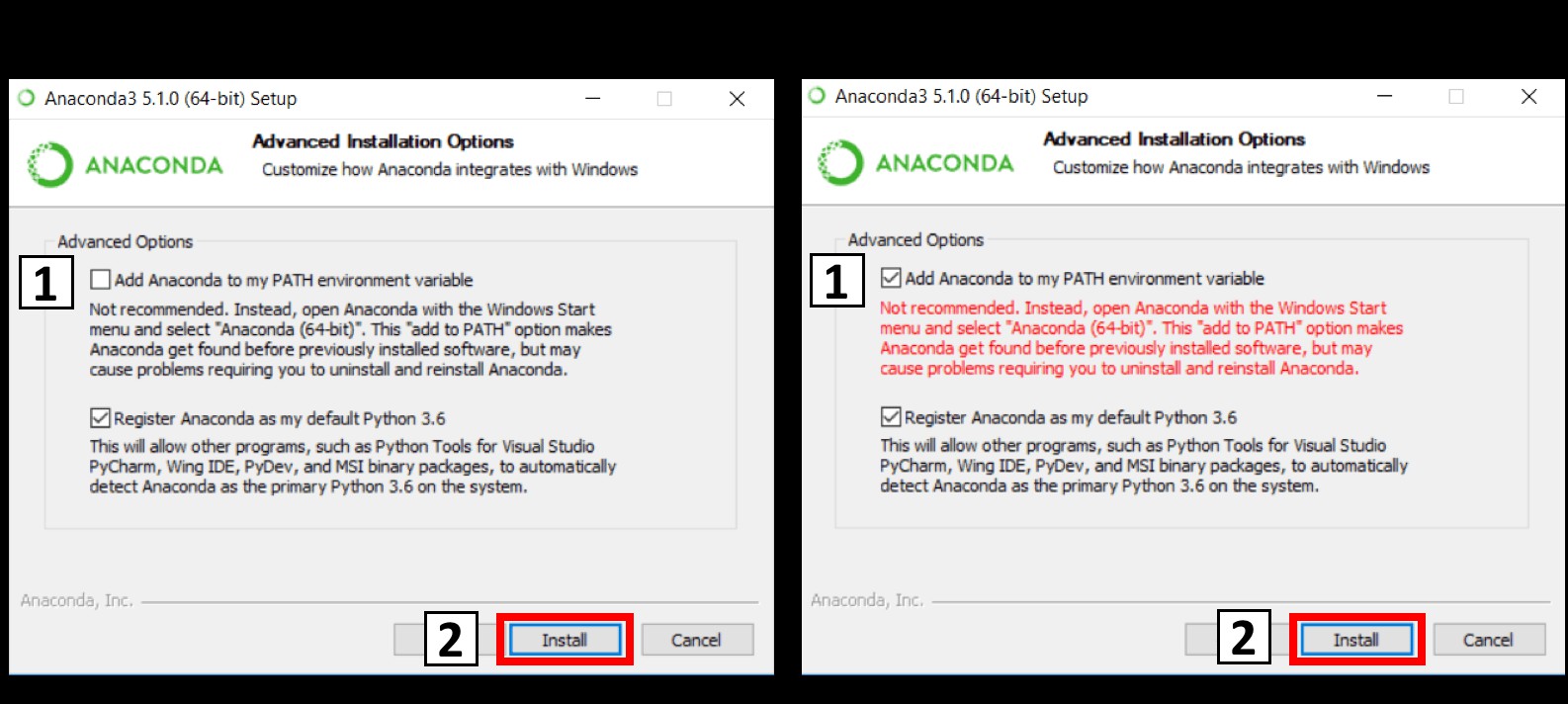


Fig 2.1.2.1.7 Installation Options

Step 7 : Click on Next.

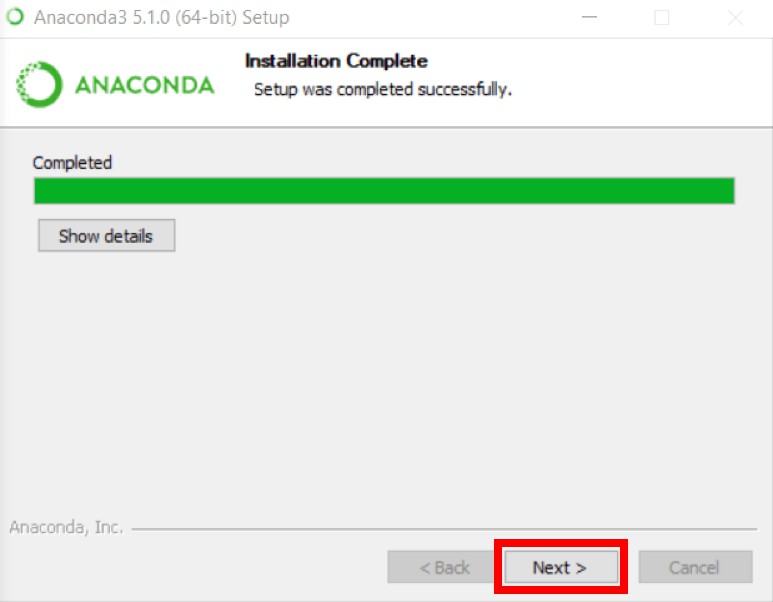


Fig 2.1.2.1.8 Installation Complete

Step 8 : You can install Microsoft VSCode if you wish, but it is optional.

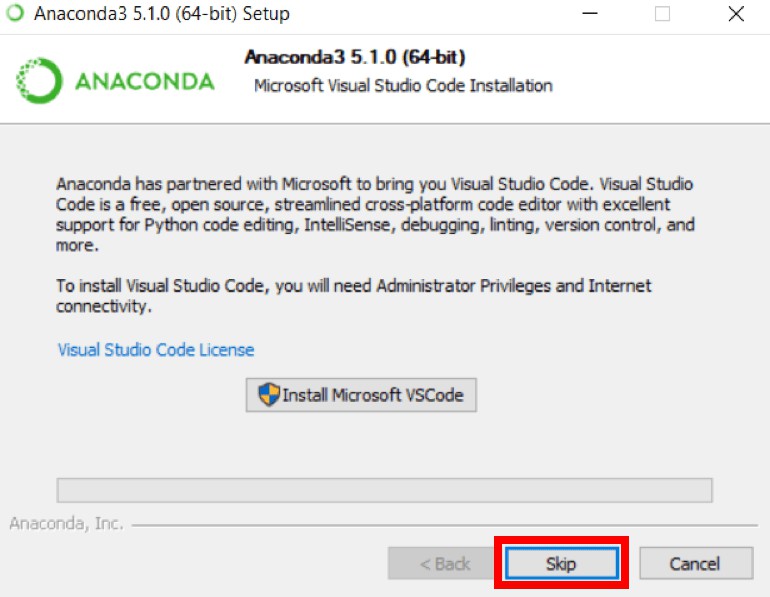


Fig 2.1.2.1.9 Visual Studio Code installation

Step 9 : Click on Finish.

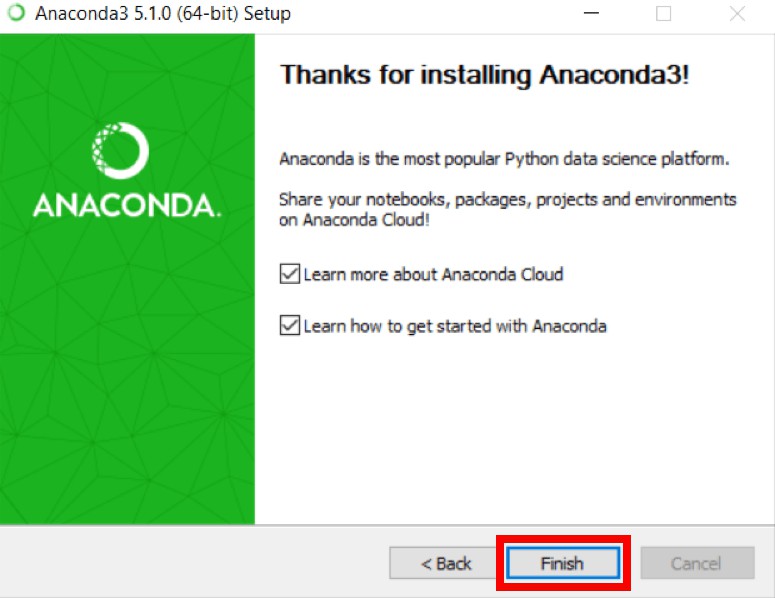


Fig 2.1.2.1.10 Installation Complete

#### ADD ANACONDA TO PATH

Step 1. Open a Command Prompt.

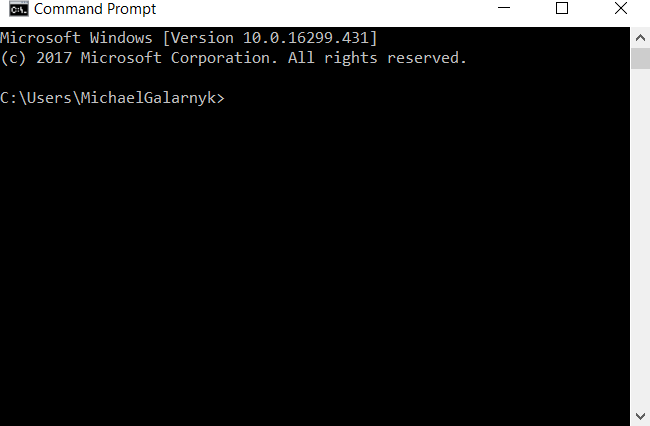


Fig 2.1.2.2.1 Opening Of Command Prompt

Step 2. Check if you already have Anaconda added to your path. Enter the commands below into your Command Prompt. This is checking if you already have Anaconda added to your path. If you get a command not recognized error like in the left side of the image below, proceed to step 3. If you get an output similar to the right side of the image below, you have already added Anaconda to your path.

conda --version

python --version

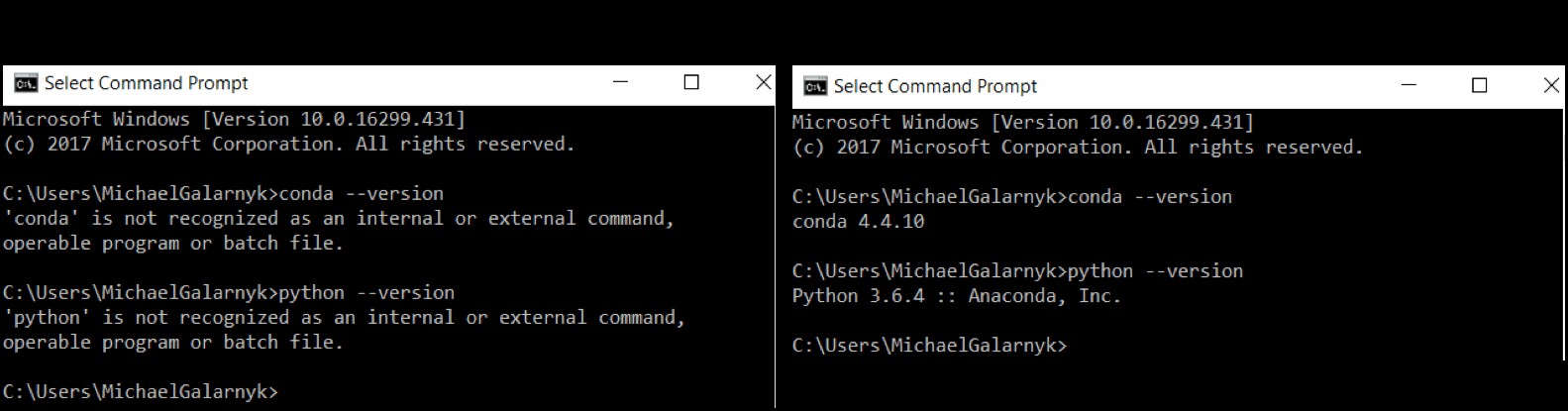


Fig 2.1.2.2.2 Checking Of Anaconda and Python Installation

Step 3. If you don't know where your conda and/or python is, open an Anaconda Prompt and type in the following commands. This is telling you where conda and python are located on your computer.

>where conda

>where python

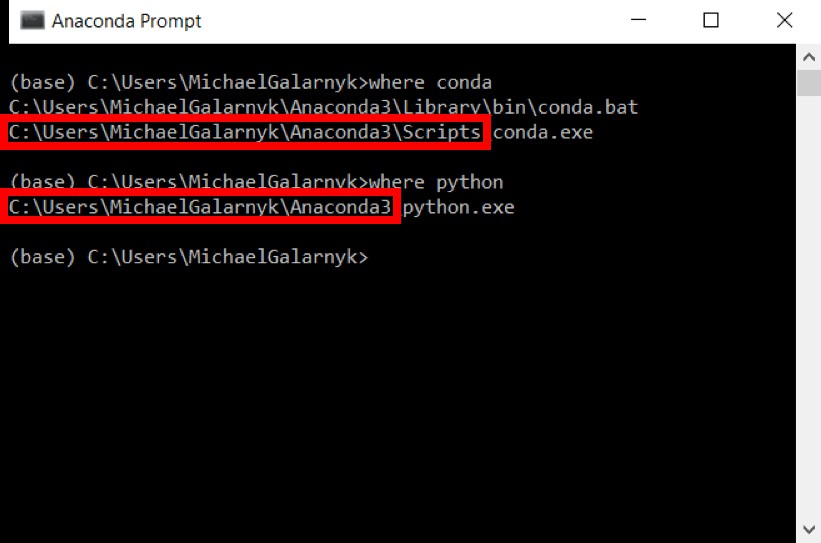


Fig 2.1.2.2.3 Knowing Location Of Python and Anaconda

Step 4. Add conda and python to your PATH. You can do this by going to your Environment Variables and adding the output of step 3 (enclosed in the red rectangle) to your path. If you are having issues, here is a short [video](https://youtu.be/mf5u2chPBjY?t=15m45s) on adding conda and python to your PATH.

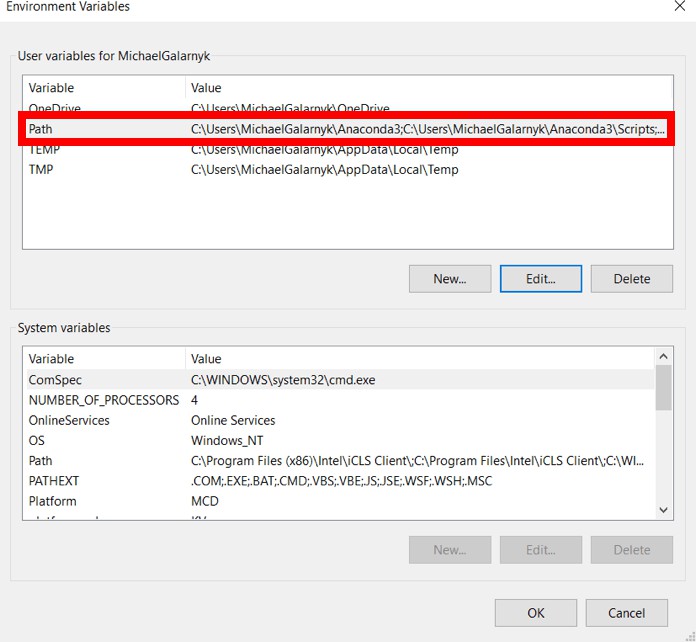


Fig 2.1.2.2.4 System and Environment Variables

Open a new Command Prompt. Try typing conda --version and python --version into the Command Prompt to check to see if everything went well.

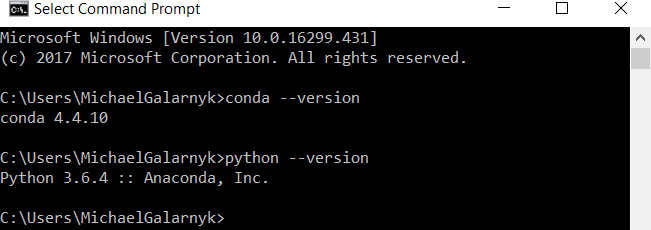


Fig 2.1.2.2.5 Checking of Python and Conda

#### 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 modelling, data visualization, machine learning, and much more.

#### INSTALLING JUPYTER NOTEBOOK USING ANACONDA

Anaconda Distribution includes Python, the Jupyter Notebook, and other commonly used packages for the scientific community.

If you don't have any version of Python installed, the recommended way to install Python is using Anaconda Distribution. It should be pretty simple to get Python installed.

* + - * + First, download the latest version of Anaconda Distribution.
        + Second, install the downloaded version of Anaconda.

To check whether the installation is successful or not, and run the Jupyter Notebook, run the following command in the Anaconda prompt or command prompt (Windows) or terminal (Mac/Linux).

>jupyter notebook

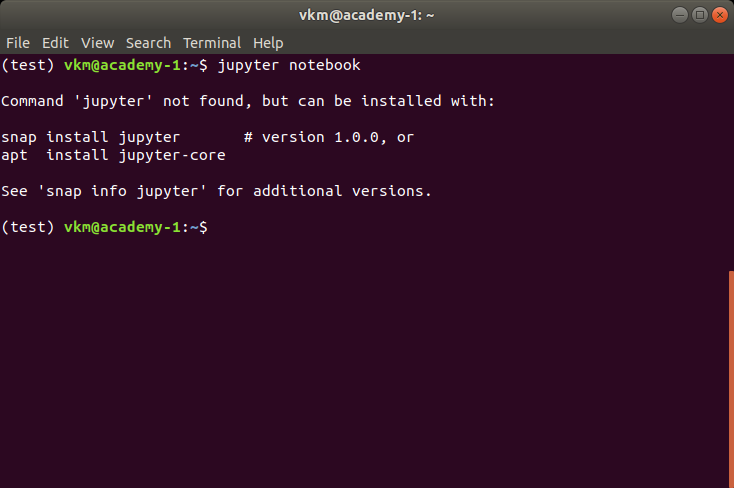


Fig 2.1.3.1.1 Checking Jupyter Notebook Installation

If the above command fails, you can continue reading this section. Otherwise, you can safely skip this and proceed to the next section. In case the command fails and you get the error similar (not exact) to the one shown below, continue with this section to understand the installation process.

#### HOW TO RUN OR OPEN JUPYTER NOTEBOOKS?

After you have installed the Jupyter Notebook on your computer, you are ready to run the notebook server.

First, open a new command prompt (Windows) or terminal (Mac/Linux) on your workstation, and second, execute the following command:

>jupyter notebook

Upon executing the above command, the terminal or command prompt will print some information about the Jupyter Notebooks being loaded. It might look something like as shown in the below snapshot. Be mindful that the information printed would be different for each workstation.

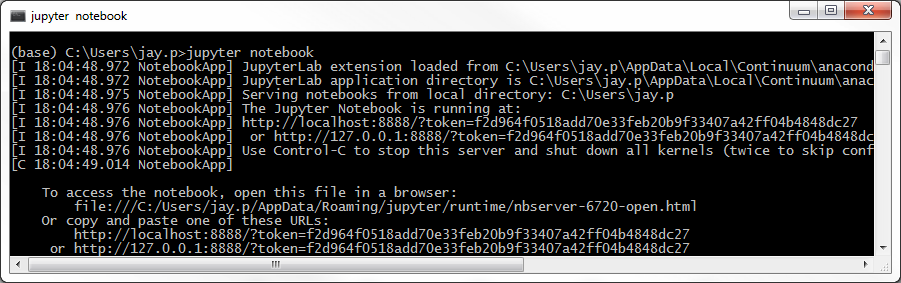


Fig 2.1.3.2.1 Command Prompt After Successful Installation

Keep the terminal open as it is. It will then open the default web browser with the URL mentioned in the command prompt or terminal.

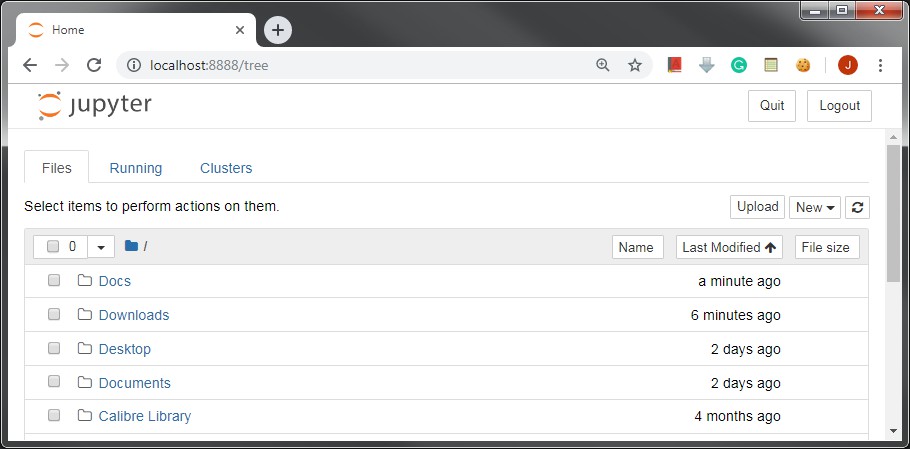
When the notebook opens in your browser, you will see the *Notebook Homepage* as shown in the below snapshot. This will list the notebook files and subdirectories in the directory where the notebook server was started.

Fig 2.1.3.2.2 Jupyter Notebook in Browser

In case you are using Anaconda distribution for Python, you can open Anaconda Navigator (using the Start Menu(Windows), Applications folder(Mac), or Softwares folder(Linux)) shown below which allows you to open Jupyter Notebook using point and click.

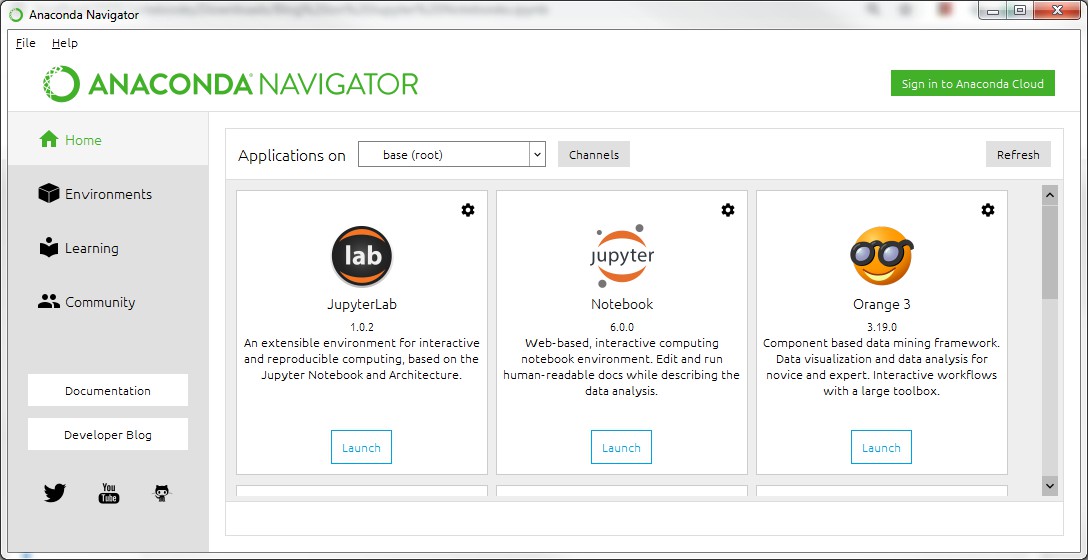


Fig 2.1.3.2.3 Opening Jupyter Notebook Using Anconda

Once the Navigator application is open, you can click on the Launch button within the Notebook dialogue to launch the Jupyter Notebook application. Upon clicking the Launch button, you will be presented by the homepage that we'd seen earlier.

Now, let's understand how Jupyter environment works, I won't be going technical, though. As the Jupyter Notebook is a web application, it works on a server-client architecture. When you execute the command jupyter notebook, the Jupyter software starts the server locally in the console where the command is executed, and the Jupyter Notebook homepage that opens in the web browser works as the client. Whatever you perform, that is, create or upload a new notebook, or save the existing one, the client notebook on which you are working, will keep communicating with the server running in the console/command line.

To keep notebooks running smoothly, we need to keep the command prompt or terminal open, even after it has opened homepage. If you close it, notebooks that you are working with, won't be able to communicate with the local server, and hence, any work you do, will not be saved on persistent memory.

#### HARDWARE REQUIREMENTS :

* **Processor** : Any updated Processor
* **Ram** : 4 GB or above
* **Hard Disk** : 50 GB or above

# LITERATURE SURVEY

### LITERATURE SURVEY

#### EXISTING SYSTEM

Phishing website detection system basically has two approaches to which are the blacklist approach and the heuristic evaluation of the source code to check for attributes commonly associated with phishing sites. The blacklist approach involves maintaining a publicly available list of reported phishing sites so that the number of victims of any new phishing site is kept very low. The main drawback with this approach is that a new phishing site has to be detected first before it can be reported. The other approach to phishing detection is to analyse the source code of a suspected phishing web page and identify attributes commonly associated with phishing sites. Use machine learning to train the system to detect phishing attacks and relearn from the data whenever a new phishing attack pattern is uncovered researchers describes a technique to classify phishing emails based on their structural properties such as style markers and the structure of the subject line, etc. Later they used Support Vector Machines to classify the emails as phishing and legitimate based on their chosen features, also describes the approach where they compared commonly used learning algorithms on a dataset of detected phishing emails composed of 860 phishing emails and 6950 legitimate emails. They handpicked ten features and used Random Forests algorithm on the dataset.

#### DISADVANTAGES OF EXISTING SYSTEM

* Inability to detect the phishing url which are not present in the blacklist which might increase the false positives rate.
* It can't handle the one time password between the user and the legitimate site.
* It provides fake identity of user, the phishers may find out the fake user by consistently analyzing fake response from user.
* Time consuming and has to evaluate huge number of features.

#### PROPOSED SYSTEM

In these machine learning algorithms are used to overcome the drawbacks associated with the traditional approaches to phishing detection. The problem of phishing detection is an ideal candidate for the application of machine learning solutions because of the easy availability of sufficient amounts of data on phishing attack patterns. The basic idea is to use machine learning algorithms on available dataset of phishing pages to generate a model which can be used to make classifications in real time if a given web page is a phishing page or a legitimate webpage. We intend to productionize the learned model into a software tool which can be deployed easily to end users for combating phishing attempts. For this purpose, we have chosen to implement a machine learning algorithm from scratch using JavaScript and build a Chrome extension with it. A Chrome extension will enable us to deploy the learned model easily on the Chrome Web Store, from where anyone can download and use our product for phishing detection. In order to successfully implement this project, we need to consider three constraints when choosing the machine learning algorithm for our product. First, the accuracy of the trained model should be high, as a product being used by end users in the real world should not give wrong results. Second, the algorithm which is being implemented should be able to make classifications in real-time; i.e. have very low execution time and also use less computational resources. Third, false positives and false negatives are important considerations when choosing a machine learning algorithm for the problem of phishing detection. This is because a user should not be wrongly led to believe that a phishing website is legitimate. Thus, we should look at these three constraints when selecting our phishing detection classifier.

# ANALYSIS

### ANALYSIS

#### MODULE ORGANIZATION

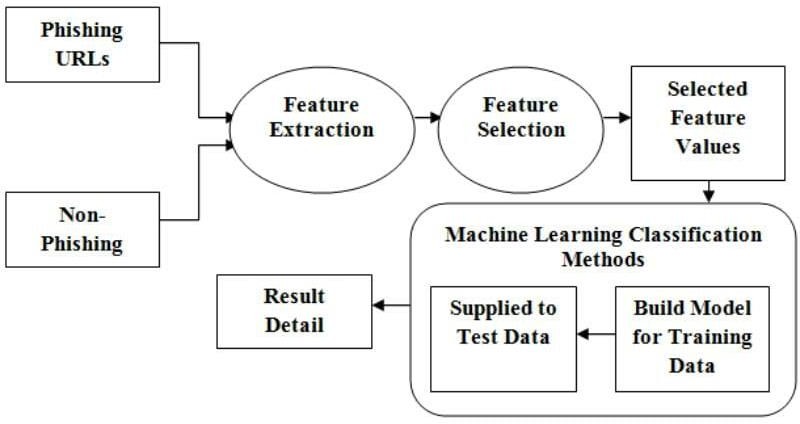


Fig 4.1.1 Block Diagram of Phishing Website Detection System

The system proposed is composed of feature selection and learning algorithm show in Fig 4.1.1 Feature selection component are responsible to extract most relevant features or attributes to identify the instance to a particular group or class. The learning algorithm component builds the necessary intelligence or knowledge using the result found from the feature selection component. Using the training dataset, the model gets trained and builds its intelligence. Then the learned intelligences are applied to the testing dataset to measure the accuracy of home much the model correctly classified on unseen data.

Feature selection is an important part in machine learning to reduce data dimensionality and extensive research carried out for a reliable feature selection method. For feature selection filter method and wrapper method have been used. In filter method, features are selected on the basis of their scores in various statistical tests that measure the relevance of features by their correlation with dependent variable or outcome variable. Wrapper method finds a subset of features by measuring the

usefulness of a subset of feature with the dependent variable. Hence filter methods are independent of any machine learning algorithm whereas in wrapper method the best feature subset selected depends

on the machine learning algorithm used to train the model. In wrapper method a subset evaluator uses all possible subsets and then uses a classification algorithm to convince classifiers from the features in each subset. The classifier consider the subset of feature with which the classification algorithm performs the best. To find the subset, the evaluator uses different search techniques like depth first search, random search, breadth first search or hybrid search. The filter method uses an attribute evaluator along with a ranker to rank all the features in the dataset. Here one feature is omitted at a time that has lower ranks and then sees the predictive accuracy of the classification algorithm. Weights or rank put by the ranker algorithms are different than those by the classification algorithm. Wrapper method is useful for machine learning test whereas filter method is suitable for data mining test because data mining has thousands of millions of features.

With the features found in feature selection part, total four models are built using the training dataset. Classification using supervised machine learning first requires training the model using training dataset. We used 20% of phishtank dataset as training data that have 10,000 labelled data instances. To training the model we used Decision Tree, Random Forest, XGBoost Classifier, Multilayer Perceptron learning algorithm and Support Vector Machines for each type of feature selection method. Hence we build five learning models. Among the models built for each learning algorithm, one is built using 17 features found in the feature selection part. Next these five trained models were evaluated using 8,000 instances of testing data picked from the Phishtank testing dataset. Based on the best features found in the feature selection process, learning models are developed. To develop the learning model, machine learning algorithm is used. Training dataset is used to train the algorithm with the selected features. In supervised machine learning, each instance in the training dataset has the class it belongs to. The algorithm build the learning model based on which machine learning algorithm is being used.

#### FEASIBILITY STUDY

Phishing costs Internet users billions of dollars per year. It refers to luring techniques used by identity thieves to fish for personal information in a pond of unsuspecting internet users. Phishers use spoofed e-mail, phishing software to steal personal information and financial account details such as usernames and passwords. This paper deals with methods for detecting phishing web sites by analyzing various features of benign and phishing URLs by Machine learning techniques. We discuss

the methods used for detection of phishing websites based on lexical features, host properties and page importance properties. We consider various data mining algorithms for evaluation of the features in order to get a better understanding of the structure of URLs that spread phishing. The fine-tuned parameters are useful in selecting the apt machine learning algorithm for separating the phishing sites from benign sites.

# DESIGN

### DESIGN:

#### INPUT AND OUTPUT DESIGN

* + 1. **INPUT DESIGN**

The [‘Phishing Dataset - A Phishing and Legitimate Dataset for Rapid](http://www.fcsit.unimas.my/research/legit-phish-set) [Benchmarking’](http://www.fcsit.unimas.my/research/legit-phish-set) dataset consists of 15,000 websites out of which 7,500 are phishing and 7,500 are legitimate. Each website in the data set comes with HTML code, whois info, URL, and all the files embedded in the web page. This is a goldmine for someone looking to apply machine learning for phishing detection. There are several ways this data set can be used. We can try to detect phishing websites by looking at the URLs and whois information and manually extracting features as some previous studies have done . However, we are going to use the raw HTML code of the web pages to see if we can effectively combat phishing websites by building a machine learning system. Among URLs, whois information, and HTML code, the last is the most difficult to obfuscate or change if an attacker is trying to prevent a system from detecting his/her phishing websites, hence the use of HTML code in our system. Another approach is to combine all three sources, which should give better and more robust results but for the sake of simplicity, we will only use HTML code and show that it alone garners effective results for phishing website detection. One final note on the data set: we will only be using 20,000 total samples because of computing constraints. We will also only consider websites written in English since data for other languages is sparse.

#### OBJECTIVE

The goal of this project is to develop a heuristic based phishing detection system through which a user can determine whether a URL is a phished, suspicious or legitimate one. Using several heuristics, the features of the URL will be extracted and examined in classifying the website based on the results obtained.

The following are the main objectives of this project.

1. To provide knowledge and insight to the inexperienced web users in identifying the phishing URLs.
2. To give in-depth information about the website features which are used in predicting the phishing attacks.
3. To categorize the website features into several classifiers which contribute to the final prediction of a given website
4. To evaluate the classification accuracy of a phishing dataset using Confusion Matrix.

#### OUTPUT DESIGN

The basic idea is to use machine learning algorithms on available dataset of phishing pages to generate a model which can be used to make classifications in real time if a given web page is a phishing page or a legitimate webpage. We intend to productionize the learned model into a software tool which can be deployed easily to end users for combating phishing attempts. For this purpose, we have chosen to implement a machine learning algorithm from scratch using JavaScript and build a Chrome extension with it. A Chrome extension will enable us to deploy the learned model easily on the Chrome Web Store, from where anyone can download and use our product for phishing detection. In order to successfully implement this project, we need to consider three constraints when choosing the machine learning algorithm for our product. First, the accuracy of the trained model should be high, as a product being used by end users in the real world should not give wrong results. Second, the algorithm which is being implemented should be able to make classifications in real-time; i.e. have very low execution time and also use less computational resources. Third, false positives and false negatives are important considerations when choosing a machine learning algorithm for the problem of phishing detection. This is because a user should not be wrongly led to believe that a phishing website is legitimate. Thus, we should look at these three constraints when selecting our phishing detection classifier.

The output form of a phishing website detection system should accomplish one or more of the following objectives:

* + - * Detects the phishing website accurately.
      * Real-time use in future.
      * Displays the accuracy of various classifiers.

#### UML DIAGRAMS

This section focuses on the system architecture of the proposed system in detecting the phishing URLs . The main goal of the proposed system is to detect an URL which is provided as input by the user as a phished, suspicious or legitimate URL. It uses several heuristics in order to determine whether a URL is phished or not. The system design involves designing a User Interface through which user inputs an URL and thereafter, the system displays the output results to the user. Once the input URL is submitted, the system extracts the website features using Python built-in functions and collect all features which would be used in classification phase for classifying the input URL.

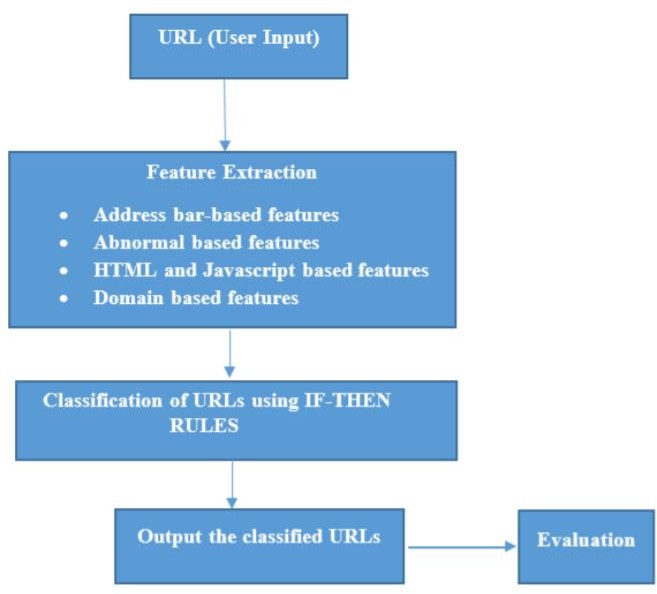


Fig 5.2.1 System Architecture Phishing Detection System

The following are the steps included in the system architecture shown in Figure.2.

1. **User Input:** As part of first step, the user inputs a URL (either Phishing or a Legitimate URL) or uploads a file containing URLs. Once the URL is fed to the system, the system extracts features such

as number of visitors, number of pages visited by them. These features give a brief overview on category of URL thereby increasing the response time of the system.

1. **Feature Extraction:** In this step, all the relevant features of the URLs are extracted which are used to differentiate between phishing URLs and legitimate URLs. A URL feature is classified into three groups such as Address-bar based features, Abnormal features, HTML and JavaScript based features and Domain based features
2. **URL Classification:** The features which are extracted from the previous step are subjected to different heuristics. A total of 16 heuristics will be used to determine whether a URL is a phished, suspicious or legitimate one. Based on the given heuristics and the features extracted, the proposed rules are applied to these features in order to categorize a URL.
3. **Output the results:** The results are displayed on to the UI in form of tables, graphs which gives a pictorial representation of the results obtained. Also, the severity of the phishing is mentioned for each URL on a scale of 1-5 to determine the impact of phishing on that URL.
4. **Evaluation:** The results of the classification are evaluated using Precision and Recall. Confusion matrix is generally used to evaluate the classification process by calculating the True Positives, False Positives, False Negatives and True Negatives.

## IMPLEMENTATION AND RESULT ANALYSIS

### IMPLEMENTATION AND RESULT ANALYSIS

#### INTRODUCTION

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 set of phishing URLs are collected from opensource service called PhishTank. This service provide a set of phishing URLs in multiple formats like csv, json etc. Before stating the ML model training, the data is split into 80-20 i.e., 8000 training samples & 2000 testing samples. From the dataset, it is clear that this is a supervised machine learning task. There are two major types of supervised machine learning problems, called classification and regression.

This data set comes under classification problem, as the input URL is classified as phishing (1) or legitimate (0).

#### METHOD OF IMPLEMENTATION

The dataset comes under a classification problem, so supervised learning algorithms are built and accuracy of each model is found.

#### SUPERVISED LEARNING

Supervised learning (SL) is the [machine learning](https://en.wikipedia.org/wiki/Machine_learning) task of learning a function that [maps](https://en.wikipedia.org/wiki/Map_(mathematics)) an input to an output based on example input-output pairs. It infers a function from labeled [training](https://en.wikipedia.org/wiki/Training_set) [data](https://en.wikipedia.org/wiki/Training_set) consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. This statistical quality of an algorithm is measured through the so-called [generalization error.](https://en.wikipedia.org/wiki/Generalization_error) Supervised learning is the types of machine learning in which machines are trained using well "labelled" training data, and on basis of that data,

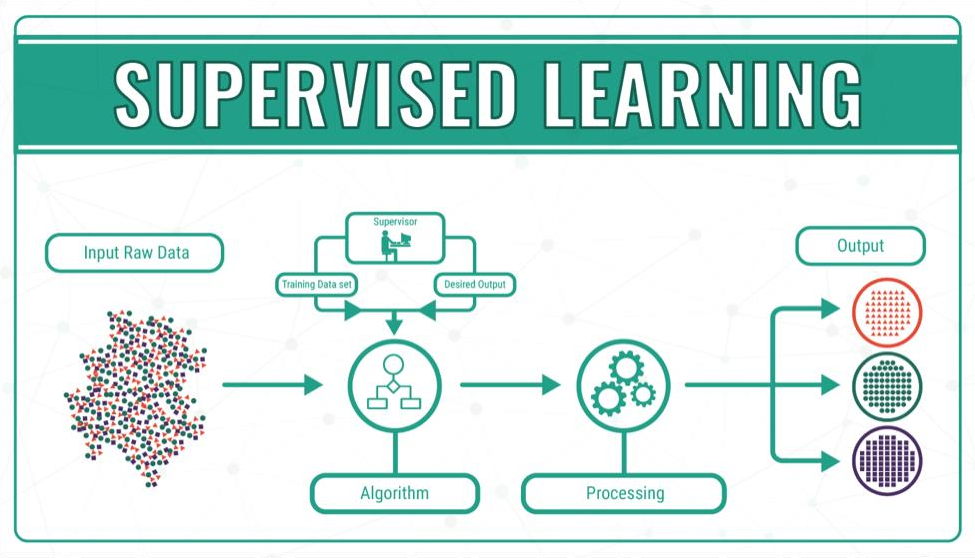
machines predict the output. The labelled data means some input data is already tagged with the correct output.

Fig 6.2.1.1 Supervised Learning

In supervised learning, the training data provided to the machines work as the supervisor that teaches the machines to predict the output correctly. It applies the same concept as a student learns in the supervision of the teacher.

Supervised learning is a process of providing input data as well as correct output data to the machine learning model. The aim of a supervised learning algorithm is to find a mapping function to map the input variable(x) with the output variable(y).

We have used various supervised learning models for prediction. They are:

* Decision Tree Classifier
* Random Forest Classifier
* XGBoost Classifier
* Multilayer Perceptrons
* Support Vector Machines

#### DECISION TREE CLASSIFIER

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surface.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

Decision tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

##### Construction of Decision Tree :

A tree can be “learned” by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called recursive partitioning. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data. In general decision tree classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

Types of decision trees are based on the type of target variable we have. It can be of two types:

* Categorical Variable Decision Tree: Decision Tree which has a categorical target variable then it called a Categorical variable decision tree.
* Continuous Variable Decision Tree: Decision Tree has a continuous target variable then it is called Continuous Variable Decision Tree.

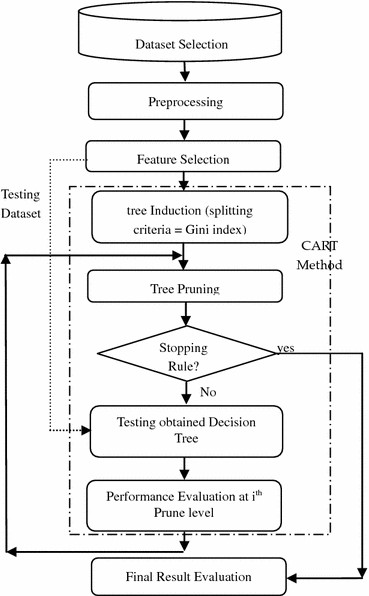


Fig 6.2.2.1 Decision Tree Diagram

#### RANDOM FOREST CLASSIFIER

Random forests or random decision forests are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification,](https://en.wikipedia.org/wiki/Statistical_classification) [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks that operates by constructing a multitude of [decision](https://en.wikipedia.org/wiki/Decision_tree_learning) [trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of [overfitting](https://en.wikipedia.org/wiki/Overfitting) to their [training](https://en.wikipedia.org/wiki/Test_set) [set](https://en.wikipedia.org/wiki/Test_set). Random forests generally outperform [decision trees,](https://en.wikipedia.org/wiki/Decision_tree_learning) but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.

The first algorithm for random decision forests was created in 1995 by [Tin Kam Ho](https://en.wikipedia.org/wiki/Tin_Kam_Ho) using the [random](https://en.wikipedia.org/wiki/Random_subspace_method) [subspace method,](https://en.wikipedia.org/wiki/Random_subspace_method) which, in Ho's formulation, is a way to implement the "stochastic discrimination" approach to classification proposed by Eugene Kleinberg.

An extension of the algorithm was developed by [Leo Breiman](https://en.wikipedia.org/wiki/Leo_Breiman) and [Adele Cutler](https://en.wikipedia.org/wiki/Adele_Cutler), who registered "Random Forests" as a [trademark](https://en.wikipedia.org/wiki/Trademark) in 2006 (as of 2019, owned by [Minitab, Inc.](https://en.wikipedia.org/wiki/Minitab)). The extension combines Breiman's "[bagging](https://en.wikipedia.org/wiki/Bootstrap_aggregating)" idea and random selection of features, introduced first by Ho and later independently by Amit and [Geman](https://en.wikipedia.org/wiki/Donald_Geman) in order to construct a collection of decision trees with controlled variance.

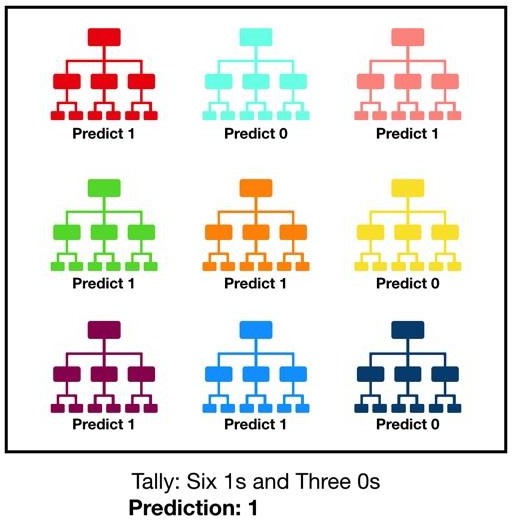


Fig 6.2.3.1 Random Forest Classifier

Random forests are frequently used as "blackbox" models in businesses, as they generate reasonable predictions across a wide range of data while requiring little configuration.

Decision trees are a popular method for various machine learning tasks. Tree learning "come closest to meeting the requirements for serving as an off-the-shelf procedure for data mining", say [Hastie](https://en.wikipedia.org/wiki/Trevor_Hastie) et al., "because it is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features, and produces inspectable models. However, they are seldom accurate".

In particular, trees that are grown very deep tend to learn highly irregular patterns: they [overfit](https://en.wikipedia.org/wiki/Overfitting) their training sets, i.e. have [low bias, but very high variance](https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff). Random forests are a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance. This comes at the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model.

Forests are like the pulling together of decision tree algorithm efforts. Taking the teamwork of many trees thus improving the performance of a single random tree. Though not quite similar, forests give the effects of a K-fold cross validation.

#### XGBOOST CLASSIFIER

**XGBoost** is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library](https://en.wikipedia.org/wiki/Library_(computing)) which provides a [regularizing](https://en.wikipedia.org/wiki/Regularization_(mathematics)) gradient boosting framework for C++, Ruby, C#, Java, Python, R, Julia, Perl and Scala. It works on [Linux](https://en.wikipedia.org/wiki/Linux), [Windows,](https://en.wikipedia.org/wiki/Windows) and [macOS](https://en.wikipedia.org/wiki/MacOS). From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". It runs on a single machine, as well as the distributed processing frameworks [Apache Hadoop,](https://en.wikipedia.org/wiki/Apache_Hadoop) [Apache Spark](https://en.wikipedia.org/wiki/Apache_Spark), and [Apache Flink](https://en.wikipedia.org/wiki/Apache_Flink).

It has gained much popularity and attention recently as the algorithm of choice for many winning teams of machine learning competitions.

XGBoost initially started as a research project by Tianqi Chen as part of the Distributed (Deep) Machine Learning Community (DMLC) group. Initially, it began as a terminal application which could be configured using a [libsvm](https://en.wikipedia.org/wiki/Libsvm) configuration file. It became well known in the ML competition circles after its use in the winning solution of the Higgs Machine Learning Challenge. Soon after, the Python and R packages were built, and XGBoost now has package implementations for Java, [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)),

Julia, [Perl](https://en.wikipedia.org/wiki/Perl), and other languages. This brought the library to more developers and contributed to its popularity among the [Kaggle](https://en.wikipedia.org/wiki/Kaggle) community, where it has been used for a large number of competitions.

It was soon integrated with a number of other packages making it easier to use in their respective communities. It has now been integrated with [scikit-learn](https://en.wikipedia.org/wiki/Scikit-learn) for [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) users and with the [caret](https://cran.rstudio.com/web/packages/caret/vignettes/caret.html) package for [R](https://en.wikipedia.org/wiki/R_(programming_language)) users. It can also be integrated into Data Flow frameworks like [Apache](https://en.wikipedia.org/wiki/Apache_Spark) [Spark,](https://en.wikipedia.org/wiki/Apache_Spark) [Apache Hadoop,](https://en.wikipedia.org/wiki/Apache_Hadoop) and [Apache Flink](https://en.wikipedia.org/wiki/Apache_Flink) using the abstracted Rabit and XGBoost4J. XGBoost is also available on [OpenCL](https://en.wikipedia.org/wiki/OpenCL) for [FPGAs.](https://en.wikipedia.org/wiki/Field-programmable_gate_array) An efficient, scalable implementation of XGBoost has been published by Tianqi Chen and Carlos Guestrin.



Fig 6.2.4.1 XGBoost Classifier

Salient features of XGBoost which make it different from other gradient boosting algorithms include:

* Clever penalization of trees
* A proportional shrinking of leaf nodes
* [Newton Boosting](https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization)
* Extra [randomization](https://en.wikipedia.org/wiki/Randomization) parameter
* Implementation on single, [distributed](https://en.wikipedia.org/wiki/Distributed_computing) systems and [out-of-core](https://en.wikipedia.org/wiki/Out-of-core) computation
* Automatic [Feature selection](https://en.wikipedia.org/wiki/Feature_selection)

#### MULTILAYER PERCEPTRON MODEL

A multilayer perceptron (MLP) is a class of [feedforward](https://en.wikipedia.org/wiki/Feedforward_neural_network) [artificial neural network](https://en.wikipedia.org/wiki/Artificial_neural_network) (ANN). The term MLP is used ambiguously, sometimes loosely to any feedforward ANN, sometimes strictly to refer to networks composed of multiple layers of [perceptrons](https://en.wikipedia.org/wiki/Perceptron) (with threshold activation). Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer.

An MLP consists of at least three [layers](https://en.wikipedia.org/wiki/Layer_(deep_learning)) of nodes: an input [layer,](https://en.wikipedia.org/wiki/Layer_(deep_learning)) a hidden [layer](https://en.wikipedia.org/wiki/Layer_(deep_learning)) and an output [layer.](https://en.wikipedia.org/wiki/Layer_(deep_learning)) Except for the input nodes, each node is a neuron that uses a nonlinear [activation](https://en.wikipedia.org/wiki/Activation_function) [function.](https://en.wikipedia.org/wiki/Activation_function) MLP utilizes a [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) technique called [backpropagation](https://en.wikipedia.org/wiki/Backpropagation) for training. Its multiple layers and non-linear activation distinguish MLP from a linear [perceptron.](https://en.wikipedia.org/wiki/Perceptron) It can distinguish data that is not [linearly separable.](https://en.wikipedia.org/wiki/Linear_separability)

Activation function:

If a multilayer perceptron has a linear [activation function](https://en.wikipedia.org/wiki/Activation_function) in all neurons, that is, a linear function that maps the [weighted inputs](https://en.wikipedia.org/wiki/Synaptic_weight) to the output of each neuron, then [linear algebra](https://en.wikipedia.org/wiki/Linear_algebra) shows that any number of layers can be reduced to a two-layer input-output model. In MLPs some neurons use a nonlinear activation function that was developed to model the frequency of [action potentials](https://en.wikipedia.org/wiki/Action_potentials), or firing, of biological neurons.

The two historically common activation functions are both [sigmoids](https://en.wikipedia.org/wiki/Sigmoids), and are described by



In recent developments of [deep learning](https://en.wikipedia.org/wiki/Deep_learning) the [rectifier linear unit (ReLU)](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)) is more frequently used as one of the possible ways to overcome the numerical [problems](https://en.wikipedia.org/wiki/Vanishing_gradient_problem) related to the sigmoids.

The first is a [hyperbolic tangent](https://en.wikipedia.org/wiki/Hyperbolic_tangent) that ranges from -1 to 1, while the other is the [logistic function](https://en.wikipedia.org/wiki/Logistic_function), which is similar in shape but ranges from 0 to 1. Here is the output of the yi th node(neuron) and is weighted sum of the input connections. Alternative activation functions have been proposed, including the [rectifier and softplus](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)) functions. More specialized activation functions include [radial basis](https://en.wikipedia.org/wiki/Radial_basis_functions) [functions](https://en.wikipedia.org/wiki/Radial_basis_functions) (used in [radial basis networks,](https://en.wikipedia.org/wiki/Radial_basis_network) another class of supervised neural network models).

The MLP consists of three or more layers (an input and an output layer with one or more hidden layers) of nonlinearly-activating nodes. Since MLPs are fully connected, each node in one layer connects with a weight following to another layer.

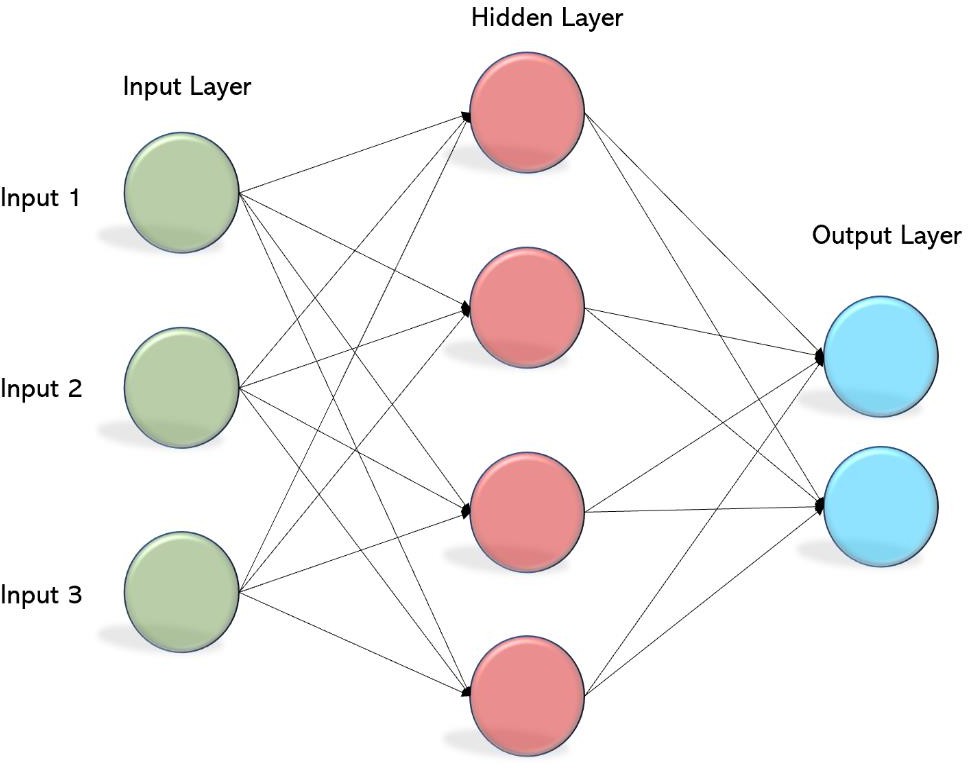


Fig 6.2.5.1 Multilayer Perceptron Model

The term "multilayer perceptron" does not refer to a single perceptron that has multiple layers. Rather, it contains many perceptrons that are organized into layers. An alternative is "multilayer perceptron network". Moreover, MLP "perceptrons" are not perceptrons in the strictest possible sense. True perceptrons are formally a special case of artificial neurons that use a threshold activation function such as the [Heaviside step function](https://en.wikipedia.org/wiki/Heaviside_step_function). MLP perceptrons can employ arbitrary activation functions. A true perceptron performs binary classification, an MLP neuron is free to either perform classification or regression, depending upon its activation function.

The term "multilayer perceptron" later was applied without respect to nature of the nodes/layers, which can be composed of arbitrarily defined artificial neurons, and not perceptrons specifically. This interpretation avoids the loosening of the definition of "perceptron" to mean an artificial neuron in general.

MLPs are useful in research for their ability to solve problems stochastically, which often allows approximate solutions for extremely [complex](https://en.wikipedia.org/wiki/Computational_complexity_theory) problems like [fitness approximation](https://en.wikipedia.org/wiki/Fitness_approximation).

MLPs are universal function approximators as shown by Cybenko's theorem, so they can be used to create mathematical models by regression analysis. As [classification](https://en.wikipedia.org/wiki/Statistical_classification) is a particular case of [regression](https://en.wikipedia.org/wiki/Regression_analysis) when the response variable is [categorical](https://en.wikipedia.org/wiki/Categorical_variable), MLPs make good classifier algorithms.

MLPs were a popular machine learning solution in the 1980s, finding applications in diverse fields such as [speech recognition](https://en.wikipedia.org/wiki/Speech_recognition), [image recognition](https://en.wikipedia.org/wiki/Image_recognition), and [machine translation](https://en.wikipedia.org/wiki/Machine_translation) software, but thereafter faced strong competition from much simpler (and related) [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine). Interest in backpropagation networks returned due to the successes of [deep learning](https://en.wikipedia.org/wiki/Deep_learning).

#### SUPPORT VECTOR MACHINE

In [machine learning,](https://en.wikipedia.org/wiki/Machine_learning) support-vector machines (SVMs, also support-vector networks) are [supervised](https://en.wikipedia.org/wiki/Supervised_learning) [learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis) [analysis.](https://en.wikipedia.org/wiki/Regression_analysis) Developed at [AT&T Bell Laboratories](https://en.wikipedia.org/wiki/AT%26T_Bell_Laboratories) by [Vladimir Vapnik](https://en.wikipedia.org/wiki/Vladimir_Vapnik) with colleagues (Boser et al., 1992, Guyon et al., 1993, Vapnik et al., 1997), SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or [VC theory](https://en.wikipedia.org/wiki/VC_theory) proposed by Vapnik (1982, 1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier) (although methods such as [Platt](https://en.wikipedia.org/wiki/Platt_scaling) [scaling](https://en.wikipedia.org/wiki/Platt_scaling) exist to use SVM in a probabilistic classification setting). SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing [linear classification](https://en.wikipedia.org/wiki/Linear_classifier), SVMs can efficiently perform a non-linear classification using what is called the [kernel trick,](https://en.wikipedia.org/wiki/Kernel_method#Mathematics%3A_the_kernel_trick) implicitly mapping their inputs into high- dimensional feature spaces.

When data are unlabelled, supervised learning is not possible, and an [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) approach is required, which attempts to find natural [clustering of the data](https://en.wikipedia.org/wiki/Cluster_analysis) to groups, and then map new data to these cluster formed groups. The support-vector machine clustering algorithm, created by [Hava](https://en.wikipedia.org/wiki/Hava_Siegelmann)

[Siegelmann](https://en.wikipedia.org/wiki/Hava_Siegelmann) and [Vladimir Vapnik,](https://en.wikipedia.org/wiki/Vladimir_Vapnik) applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications.

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N — the number of features) that distinctly classifies the data points.

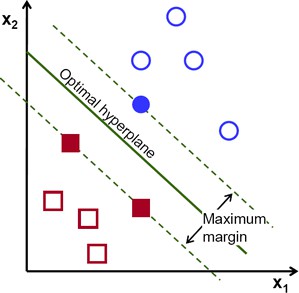
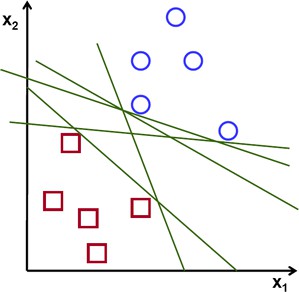


Fig 6.2.6.1 SVM Hyperplanes

To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

##### Hyperplanes and Support Vectors

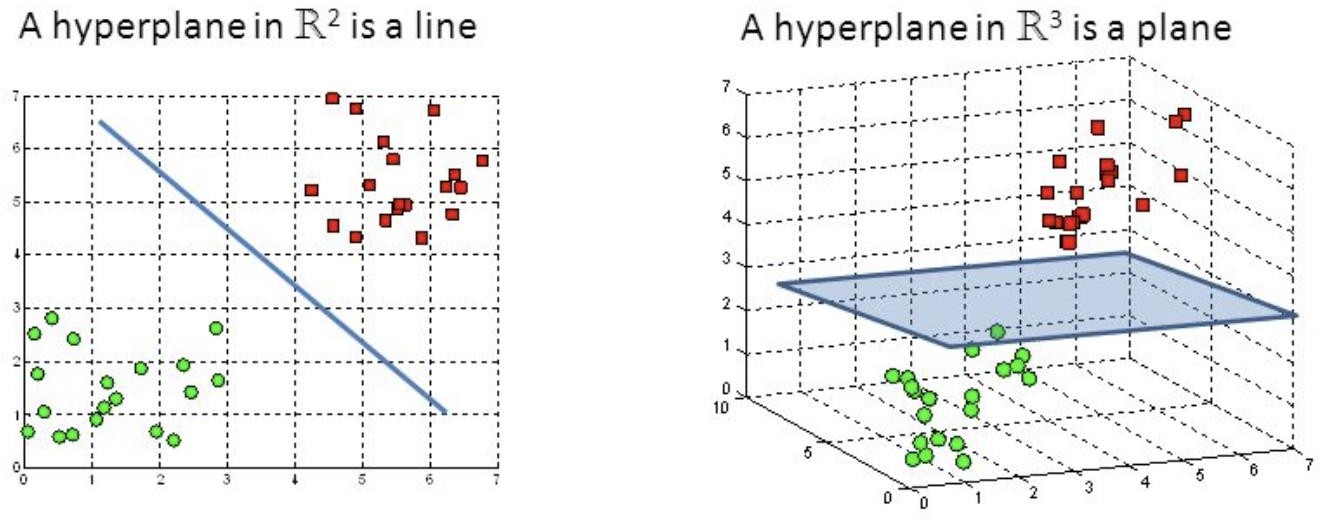


Fig 6.2.6.2 Hyperplanes in 2D and 3D feature space

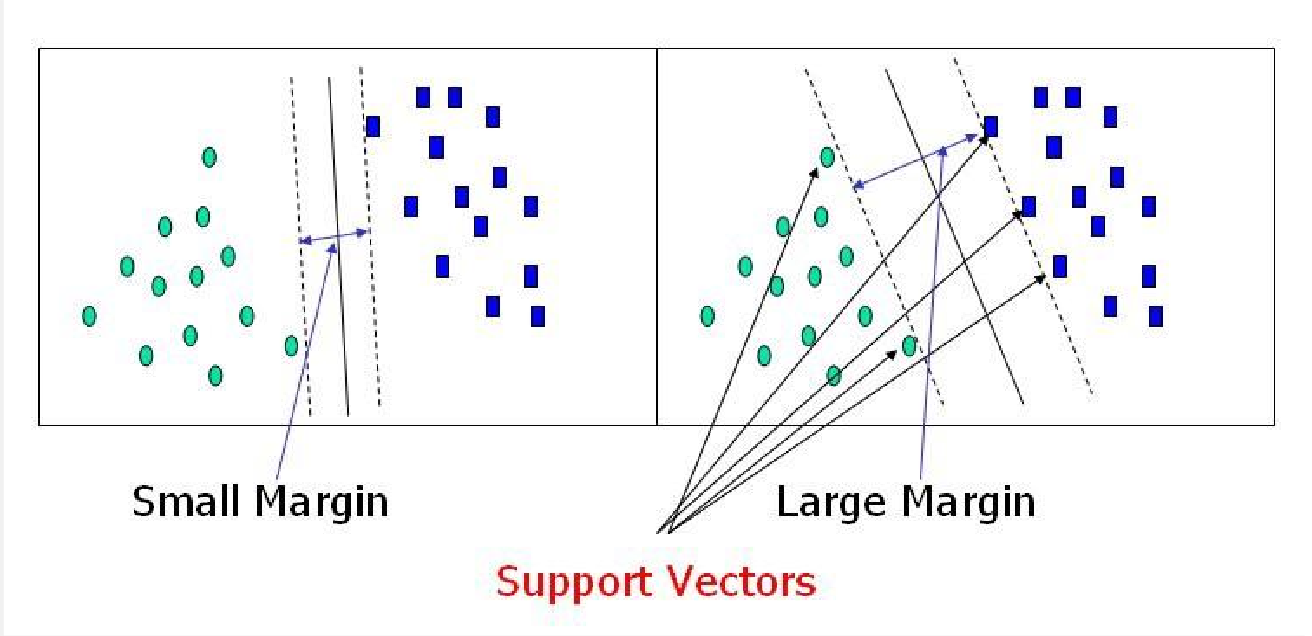
Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two- dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

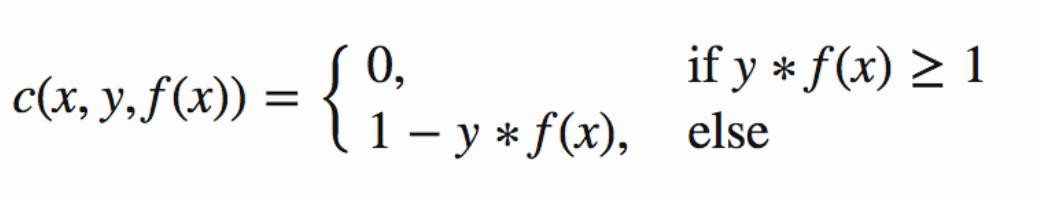
Fig 6.2.6.3 Support Vectors

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these su pport vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

##### Large Margin Intuition

In logistic regression, we take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function. If the squashed value is greater than a threshold value(0.5) we assign it a label 1, else we assign it a label 0. In SVM, we take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify is with another class. Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values([-1,1]) which acts as margin.

##### Cost Function and Gradient Updates

In the SVM algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function that helps maximize the margin is hinge loss.

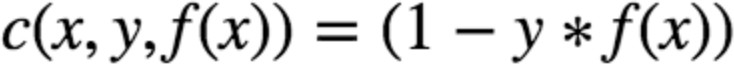


Fig 6.2.6.4 Hinge loss function

The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions looks as below.

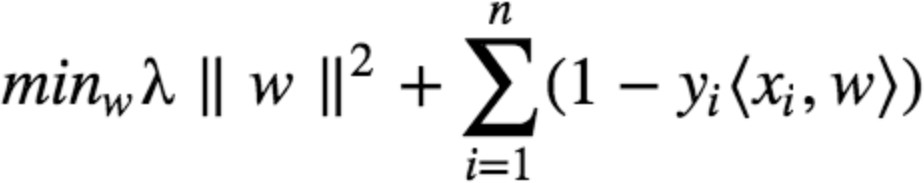


Fig 6.2.6.5 Loss Function for SVM

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

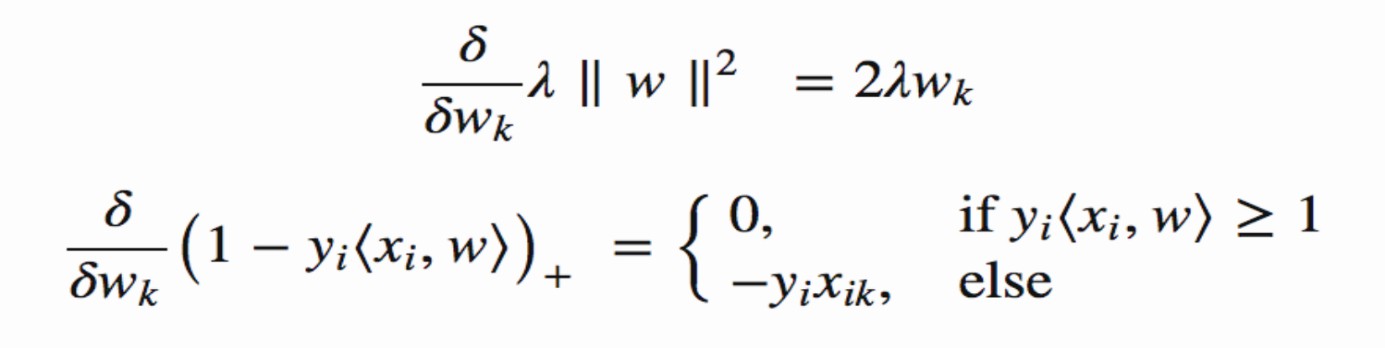


Fig 6.2.6.6 Gradients

When there is no misclassification, i.e our model correctly predicts the class of our data point, we only have to update the gradient from the regularization parameter.



Fig 6.2.6.7 Gradient Update-No Misclassification

When there is a misclassification, i.e our model make a mistake on the prediction of the class of our data point, we include the loss along with the regularization parameter to perform gradient update.



Fig 6.2.6.8 Gradient Update-Misclassification

#### SOURCE CODE

#importing basic packages import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt

#Loading the data

data0 = pd.read\_csv('5.urldata.csv') data0.head()

#Checking the shape of the dataset data0.shape

#Listing the features of the dataset data0.columns

#Listing the features of the dataset data0.columns

#DATA VISUALISATION

#Plotting the data distribution data0.hist(bins = 50,figsize = (15,15)) plt.show()

#Correlation heatmap plt.figure(figsize=(15,13)) sns.heatmap(data0.corr()) plt.show()

#DATA PREPROCESSING

data0.describe()

#Dropping the Domain column

data = data0.drop(['Domain'], axis = 1).copy() #checking the data for null or missing values data.isnull().sum()

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

data.head() #SPLITTING DATA

# Sepratating & assigning features and target columns to X & y y = data['Label']

X = data.drop('Label',axis=1)

X.shape, y.shape

# Splitting the dataset into train and test sets: 80-20 split from sklearn.model\_selection import train\_test\_split X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y,

test\_size = 0.2, random\_state = 12)

X\_train.shape, X\_test.shape #importing packages

from sklearn.metrics import accuracy\_score

# Creating holders to store the model performance results ML\_Model=[]

acc\_train = [] acc\_test = []

#function to call for storing the results

def storeResults(model, a,b): ML\_Model.append(model) acc\_train.append(round(a, 3))

acc\_test.append(round(b, 3)) # Decision Tree model

from sklearn.tree import DecisionTreeClassifier # instantiate the model

tree = DecisionTreeClassifier(max\_depth = 5) # fit the model

tree.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_test\_tree = tree.predict(X\_test)

y\_train\_tree = tree.predict(X\_train)

#computing the accuracy of the model performance acc\_train\_tree = accuracy\_score(y\_train,y\_train\_tree) acc\_test\_tree = accuracy\_score(y\_test,y\_test\_tree) #printing acuuracy

print("Decision Tree: Accuracy on training Data: {:.3f}".format(acc\_train\_tree)) print("Decision Tree: Accuracy on test Data: {:.3f}".format(acc\_test\_tree)) #checking the feature improtance in the model

plt.figure(figsize=(9,7)) n\_features = X\_train.shape[1]

plt.barh(range(n\_features), tree.feature\_importances\_, align='center')

#plotting

plt.yticks(np.arange(n\_features), X\_train.columns) plt.xlabel("Feature importance") plt.ylabel("Feature")

plt.show()

*#storing the results*

storeResults('Decision Tree', acc\_train\_tree, acc\_test\_tree) # Random Forest model

from sklearn.ensemble import RandomForestClassifier

# instantiate the model

forest = RandomForestClassifier(max\_depth=5)

# fit the model forest.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_test\_forest = forest.predict(X\_test)

y\_train\_forest = forest.predict(X\_train) #computing the accuracy of the model performance

acc\_train\_forest = accuracy\_score(y\_train,y\_train\_forest) acc\_test\_forest = accuracy\_score(y\_test,y\_test\_forest)

print("Random forest: Accuracy on training Data: {:.3f}".format(acc\_train\_forest)) print("Random forest: Accuracy on test Data: {:.3f}".format(acc\_test\_forest)) #checking the feature improtance in the model

plt.figure(figsize=(9,7))

n\_features = X\_train.shape[1]

plt.barh(range(n\_features), forest.feature\_importances\_, align='center') plt.yticks(np.arange(n\_features), X\_train.columns)

plt.xlabel("Feature importance") plt.ylabel("Feature")

plt.show() #storing the results

storeResults('Random Forest', acc\_train\_forest, acc\_test\_forest) # Multilayer Perceptrons model

from sklearn.neural\_network import MLPClassifier

# instantiate the model

mlp = MLPClassifier(alpha=0.001, hidden\_layer\_sizes=([100,100,100]))

# fit the model mlp.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_test\_mlp = mlp.predict(X\_test)

y\_train\_mlp = mlp.predict(X\_train)

#computing the accuracy of the model performance acc\_train\_mlp = accuracy\_score(y\_train,y\_train\_mlp) acc\_test\_mlp = accuracy\_score(y\_test,y\_test\_mlp)

print("Multilayer Perceptrons: Accuracy on training Data: {:.3f}".format(acc\_train\_mlp))

print("Multilayer Perceptrons: Accuracy on test Data: {:.3f}".format(acc\_test\_mlp)) #storing the results

storeResults('Multilayer Perceptrons', acc\_train\_mlp, acc\_test\_mlp) #XGBoost Classification model

from xgboost import XGBClassifier from xgboost import plot\_tree

# instantiate the model

xgb = XGBClassifier(learning\_rate=0.4,max\_depth=7) #fit the model

xgb.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_test\_xgb = xgb.predict(X\_test)

y\_train\_xgb = xgb.predict(X\_train)

#computing the accuracy of the model performance acc\_train\_xgb = accuracy\_score(y\_train,y\_train\_xgb) acc\_test\_xgb = accuracy\_score(y\_test,y\_test\_xgb)

print("XGBoost: Accuracy on training Data: {:.3f}".format(acc\_train\_xgb)) print("XGBoost : Accuracy on test Data: {:.3f}".format(acc\_test\_xgb)) #storing the results

storeResults('XGBoost', acc\_train\_xgb, acc\_test\_xgb) #Support vector machine model

from sklearn.svm import SVC # instantiate the model

svm = SVC(kernel='linear', C=1.0, random\_state=12) #fit the model

svm.fit(X\_train, y\_train)

#predicting the target value from the model for the samples y\_test\_svm = svm.predict(X\_test)

y\_train\_svm = svm.predict(X\_train)

#computing the accuracy of the model performance acc\_train\_svm = accuracy\_score(y\_train,y\_train\_svm) acc\_test\_svm = accuracy\_score(y\_test,y\_test\_svm)

print("SVM: Accuracy on training Data: {:.3f}".format(acc\_train\_svm)) print("SVM : Accuracy on test Data: {:.3f}".format(acc\_test\_svm)) #storing the results

storeResults('SVM', acc\_train\_svm, acc\_test\_svm) #COMPARISON OF MODELS

#creating dataframe

results = pd.DataFrame({ 'ML Model': ML\_Model, 'Train Accuracy': acc\_train,

'Test Accuracy': acc\_test}) results

results.sort\_values(by=['Test Accuracy', 'Train Accuracy'], ascending=False)

# save XGBoost model to file import pickle

pickle.dump(xgb, open("XGBoostClassifier.pickle.dat", "wb")) # load model from file

loaded\_model = pickle.load(open("XGBoostClassifier.pickle.dat", "rb")) loaded\_model

#### OUTPUT SCREENS

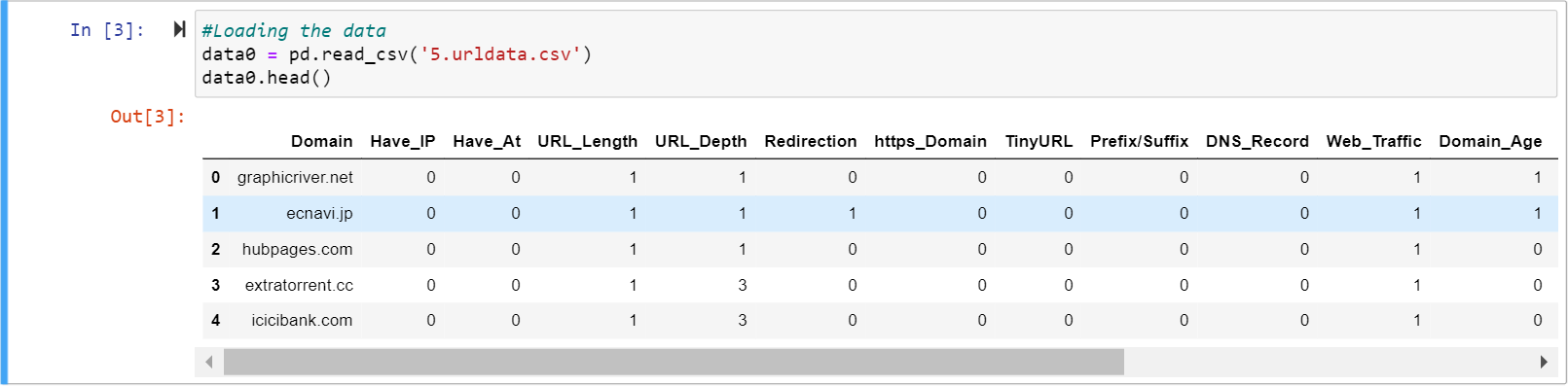


Fig 6.4.1 Screen Showing Loading and displaying of Dataset

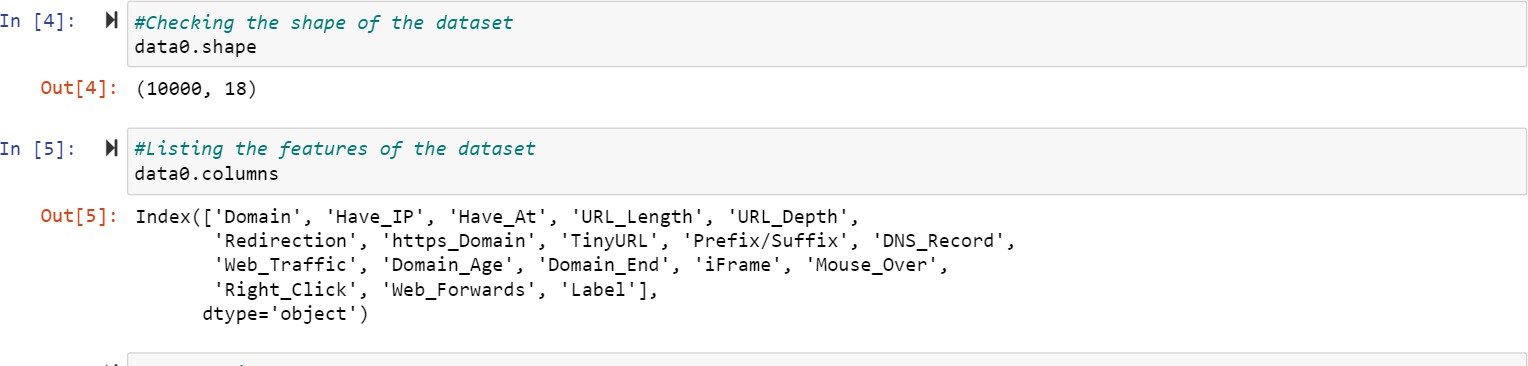


Fig 6.4.2 Screen showing Dataset Shape & Listing Features

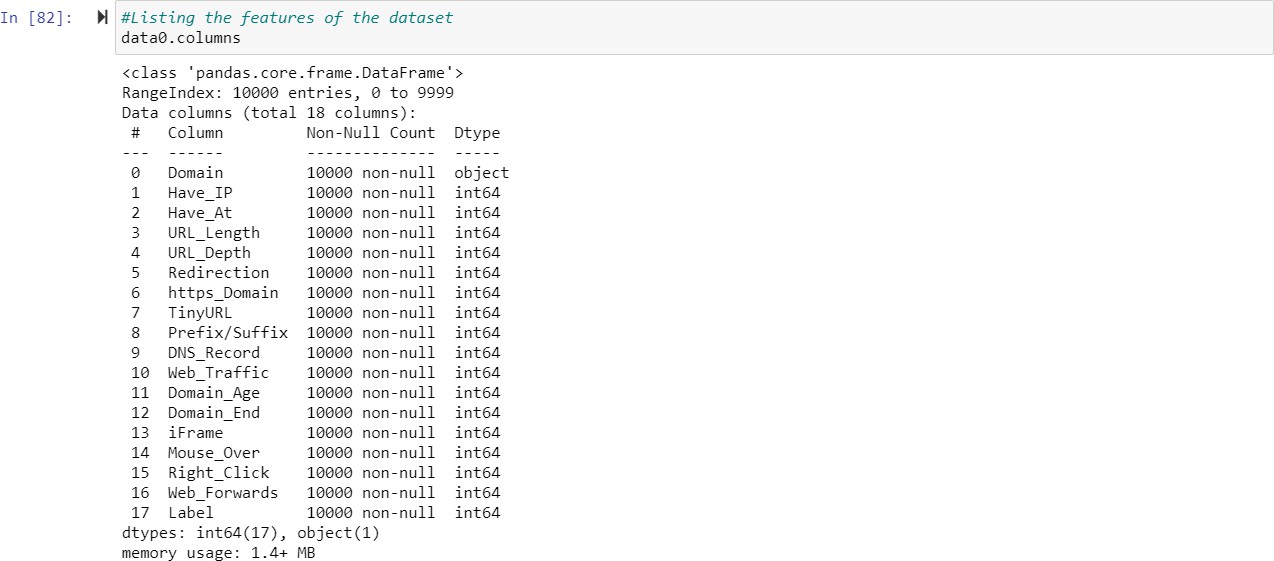


Fig 6.4.3 Listing Columns of Dataset

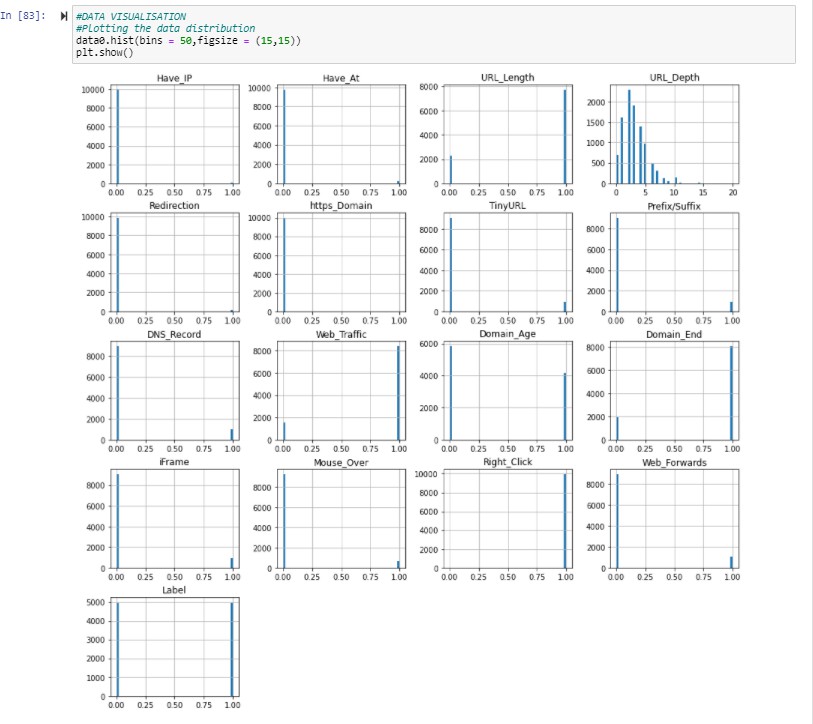


Fig 6.4.4 Histogram Visualization of Dataset

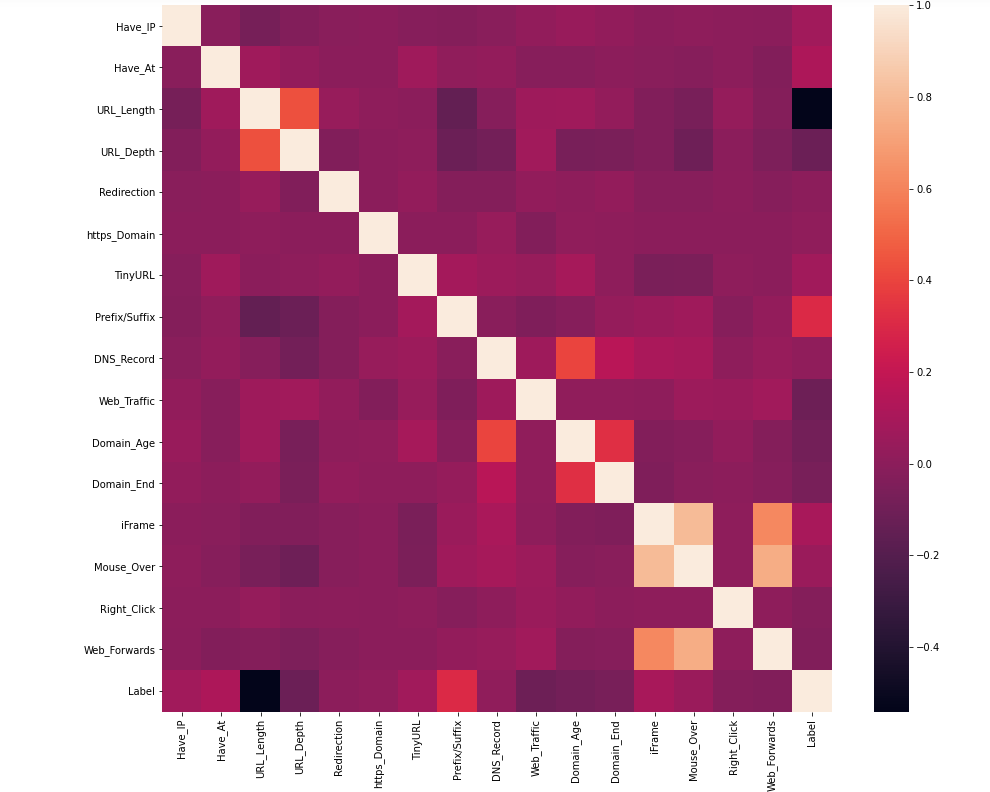


Fig 6.4.5 Correlation Heatmap of Dataset

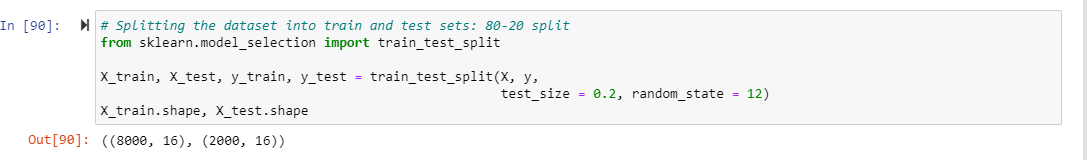


Fig 6.4.6 Splitting Dataset into Train and Test Data

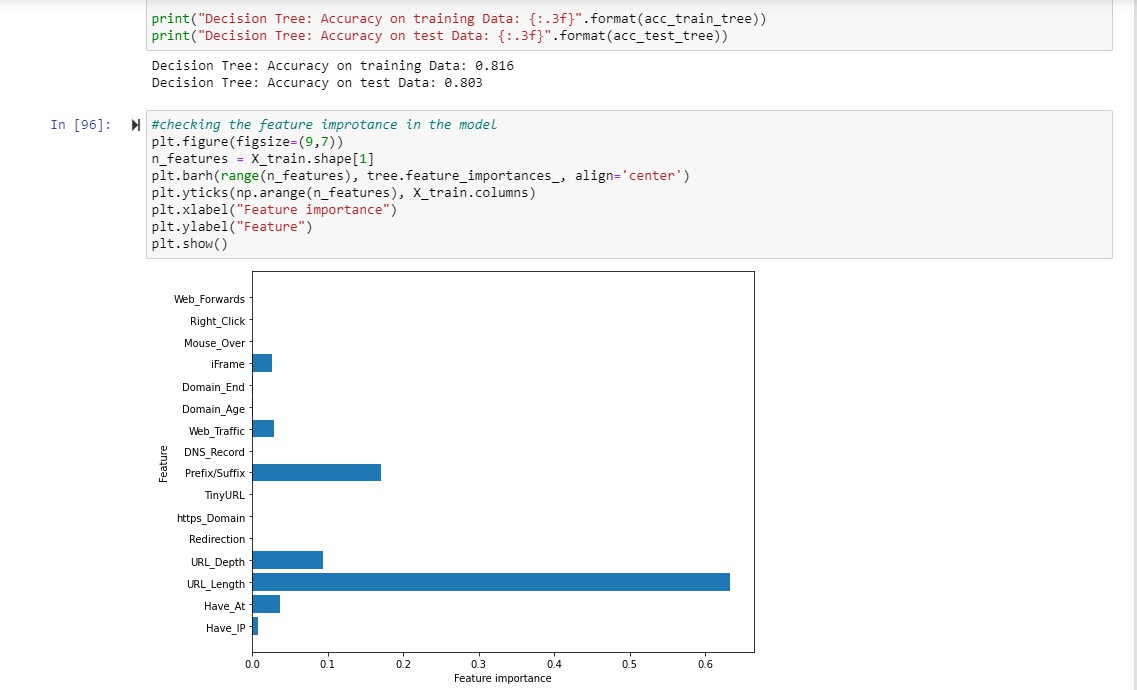


Fig 6.4.7 Accuracy Score & Feature Importance of Decision Tree

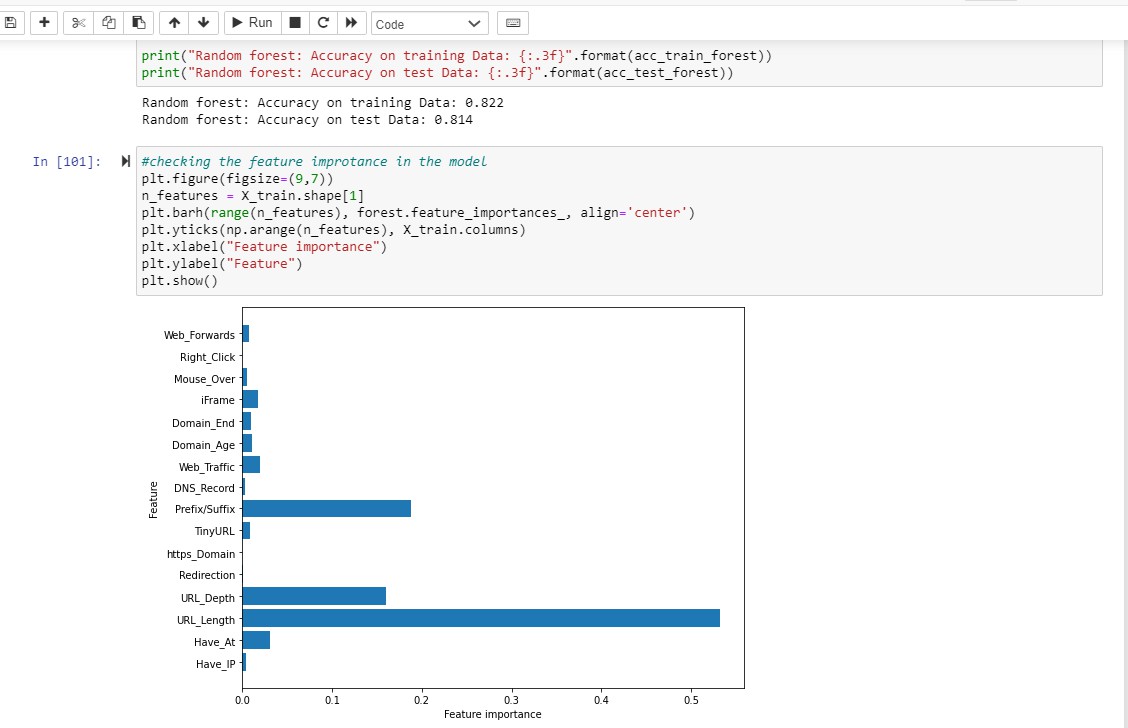


Fig 6.4.8 Accuracy Score & Feature Importance of Random Forest

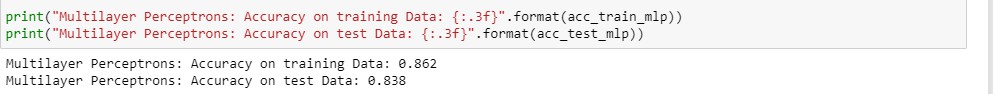


Fig 6.4.9 Screen showing Accuracy Score of Multilayer Perceptron Model

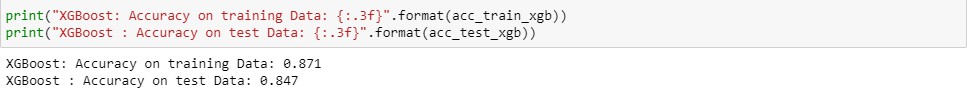


Fig 6.4.10 Screen showing Accuracy Score of XGBoost Classifier

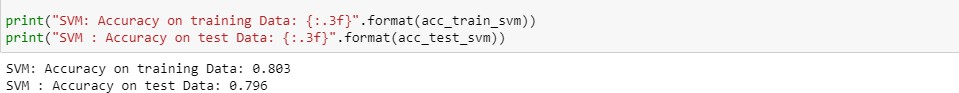


Fig 6.4.11 Screen showing Accuracy Score of SVM



Fig 6.4.12 Data Frame of All Classifiers Along With Accuracy Rates

# TESTING AND VALIDATIONS

### TESTING AND VALIDATIONS

#### INTRODUCTION:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

Testing is a critical element which assures quality and effectiveness of the proposed system in satisfying its objectives. Testing is done in various stages in the system designing and Implementation process with an objective of developing a transparent, flexible and secured system. Testing is an integral part of software development. Testing process, in way certifies, whether the product, that is developed, complies with the standards, that it was designed to. Testing process involves building of test cases against which the product has to be tested.

##### Test Objectives

* Testing is a process of executing a program with the intent of finding an error.
* A goo case is one that has a high probability of finding an undiscovered error.
* A Successful test is one that uncovers a yet undiscovered error. If the testing is conducted successfully, it will uncover errors in the software. Testing can’t show the absence of defects. It can only show that software defects are present.

##### Testing Principles

Before applying methods to design effective test cases, a software engineer must understand the basic principle that guides software testing. All the tests should be traceable to customer requirements.

#### SYSTEM TESTING:

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

##### White Box Testing

White Box Testing is a testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It has a purpose. It is used to test areas that cannot be reached from a black box level.

##### Black Box Testing

In this testing by knowing the internal operation of a product, tests can be conducted to ensure that "all gears mesh", that is the internal operation performs according to specification and all internal components have been adequately exercised. It fundamentally focuses on the functional requirements of the software.

##### The steps involved in black box test case design are:

* Graph based testing methods
* Equivalence partitioning
* Boundary value analysis
* Compaison Testing

##### Testing Strategies

A software testing strategy provides a road map for the software developer. Testing is a set of activities that can be planned in advance and conducted systematically. For this reason, a template for software testing, a set of steps into which we can place specific test case design methods should be defined for the software engineering process.

##### Any software testing strategy should have the following characteristics:

* Testing begins at the module level and works outward toward the integration of the entire computer-based system.
* Different testing techniques are appropriate at different points in time.
* The developer of the software and an independent test group conducts testing.
* Testing and debugging are different activities but debugging must be accommodated in any testing strategy

#### LEVELS OF TESTING:

##### Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application

.It is done after the completion of an individual unit before integration. This is a structural testing that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

##### Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

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.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

##### Functional Testing

Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identifying Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

##### 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.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

## CONCLUSION AND FUTURE ENHANCEMENTS

### CONCLUSION

The Phishing website detection system mainly works on detecting whether a website is phishing or legitimate accurately.Various supervised learning models are build using training data and tested using test data and accuracy on both training and testing data are measured and model with higher accuracy is taken. The model which is built using XGBoost Classifier and feature selection outperformed other models in detecting phishing websites correctly with 86% accuracy on train data and 84% accuracy on test data.The Phishing website detection system build using XGBoost Classifier can predict the phishing websites with good accuracy and allow user to secure there personal data from various phishing attacks.

### FUTURE WORK

1. This system can be implemented with web browsers to detect whether website is legitimate or not and help users to secure there personal information.
2. This model be implemented in real-time applications to develop a browser plugin, which can be published as a chrome extension.
3. Better algorithms can be developed in future to increase efficiency and quality of results.

## REFERENCES

### REFERENCES

1. AO Kaspersky lab. (2017). The Dangers of Phishing: Help employees avoid the lure of cybercrime. [Online] Available: https://go.kaspersky.com/Dangers-Phishing-Landing-Page- Soc.html [Oct 30, 2017].
2. ” Financial threats in 2016: Every Second Phishing Attack Aims to Steal Your Money” Internet: https:/[/www.kaspersky.com/](http://www.kaspersky.com/about/pressreleases/2017)a[bout/pressreleases/2017](http://www.kaspersky.com/about/pressreleases/2017) financial-threats-in-2016. Feb 22, 2017 [Oct 30,

2017].

1. Y. Zhang, J. I. Hong, and L. F. Cranor,” Cantina: A Content-based Approach to Detecting Phishing Web Sites,” New York, NY, USA, 2007, pp. 639-648.
2. M. Blasi,” Techniques for detecting zero-day phishing websites.” M.A. thesis, Iowa State University, USA, 2009.
3. R. S. Rao and S. T. Ali,” Phish Shield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach,” Procedia Computer Science, vol. 54, no. Supplement C, pp. 147-156, 2015.
4. E. Jakobsson, and E. Myers, Phishing and Counter-Measures: Understanding the Increasing Problem of Electronic Identity Theft. Wiley, 2006, pp.2–3.

##### Textbooks Referred:

* + Machine Learning For Absolute Beginners: A Plain English Introduction (2nd Edition) -

Oliver Theobald

* + Machine Learning for Hackers: Case Studies and Algorithms to Get You Started (1st Edition) - Drew Conway & John Myles