CHAPTER 1:

INTRODUCTION

1.1 INTRODUCTION

Phishing is a fraudulent technique that uses social and technological tricks to steal customer identification and financial credentials. Social media systems use spoofed e-mails from legitimate companies and agencies to enable users to use fake websites to divulge financial details like usernames and passwords. Hackers install malicious software on computers to steal credentials, often using systems to intercept username and passwords of consumers' online accounts. Phishers use multiple methods, including email, Uniform Resource Locators (URL), instant messages, forum postings, telephone calls, and text messages to steal user information. The structure of phishing content is similar to the original content and trick users to access the content in order to obtain their sensitive data. The primary objective of phishing is to gain certain personal information for financial gain or use of identity theft. Phishing attacks are causing severe economic damage around the world. Moreover, most phishing attacks target financial/payment institutions and webmail, according to the Anti-Phishing Working Group (APWG) latest Phishing pattern studies.

In order to receive confidential data, criminals develop unauthorized replicas of a real website and email, typically from a financial institution or other organization dealing with financial data. This e-mail is rendered using a legitimate company's logos and slogans. The design and structure of HTML allow copying of images or an entire website. Also, it is one of the factors for the rapid growth of Internet as a communication medium, and enables the misuse of brands, trademarks and other company identifiers that customers rely on as authentication mechanisms. To trap users, Phisher sends "spooled" mails to as many people as possible. When these e-mails are opened, the customers tend to be diverted from the legitimate entity to a spoofed website.

There is a significant chance of exploitation of user information. For these reasons, phishing in modern society is highly urgent, challenging, and overly critical. There have been several recent studies against phishing based on the characteristics of a domain, such as website URLs, website content, incorporating both the website URLs and content, the source code of the website and the screenshot of the website. However, there is a lack of useful anti-phishing tools to detect malicious URL in an organization to protect its users. In the event of malicious code being implanted on the website, hackers may steal user information and install malware, which poses a serious risk to cybersecurity and user privacy. Malicious URLs on the Internet can be easily identified by analyzing it through Machine Learning (ML) technique. The conventional URL detection approach is based on a blacklist (set of malicious URLs) obtained by user reports or manual opinions. On the one hand, the blacklist is used to verify an URL and on the other hand the URL in the blacklist is updated, frequently. However, the numbers of malicious URLs not on the blacklist are increasing significantly. For instance, cybercriminals can use a Domain Generation Algorithm (DGA) to circumvent the blacklist by creating new malicious URLs. Thus, an exhaustive blacklist of malicious URLs is almost impossible to identify the malicious URLs. Thusnew malicious URLs cannot be identified with the existing approaches. Researchers suggested methods based on the learning of computer to identify malicious URLs to resolve the limitations of the system based on the blacklist.

Malicious URL detection is considered a binary classification task with two-class predictions: malicious and benign. The training ofthe ML method consists of finding the best mapping between the d-dimensional vector space and the output variable. This strategy has a strong generalization capacity to find unknown malicious URLs compared to the blacklist approach.

Recurrent Neural Network (RNN)—Long Short-Term Memory (LSTM) is one of the ML techniques that presents a solution for the complex real—time problems. LSTM allow RNN to

store inputs for a larger period. It is similar to the concept of storage in computer. In addition, each feature will be processed according to the uniform distribution. The combination of RNN and LSTM enables to extract a lot of information from a minimum set of data. Therefore, it supports phishing detection system to identify a malicious site in a shorter duration.

In comparison to most previous approaches, researchers focus on identifying malicious URLs from the massive set of URLs. Therefore, the study proposes Recurrent Neural Network (RNN) based URL detection approach. The objectives of the study are as follows:

1. To develop a novel approach to detect malicious URL and alert users.
2. To apply ML techniques in the proposed approach in order to analyze the real time URLs and produce effective results.
3. To implement the concept of RNN, which is a familiar ML technique that has the capability to handle huge amount of data.

The year 2020 saw peoples' life being completely dependent on technology due to the global pandemic. Since digitalization became significant in this scenario, cyber criminals went on an internet crime spree. Recent reports and researches point to an increased number of security breaches that costs the victims a huge sum of money or disclosure of confidential data. Phishing is a cybercrime that employs both social engineering and technical subterfuge in order to steal personal identity data or financial account credentials of victims. In phishing, attackers counterfeit trusted websites and misdirect people to these websites, where they are tricked into sharing usernames, passwords, banking or credit card details and other sensitive credentials. These phishing URLs may be sent to the consumers through email, instant message or text message. According to the FBI crime report 2020, phishing was the most common type of cyber-attack in 2020 and phishing incidents nearly doubled from 114,702 in 2019 to 241,342 in 2020. The Verizon 2020 Data Breach Investigation Report states that 22% of data breaches in 2020 involved phishing.

The number of phishing attacks as observed by the Anti- Phishing Work Group (APWG) grew through 2020, doubling over the course of the year. In the 4th quarter of 2020, it was found that phishing attacks against financial institutions were the most prevalent. Phishing attacks against SaaS and Webmail sites were down and attacks against E-commerce sites escalated, while attacks against media companies decreased slightly from 12.6% to 11.8%. In light of the prevailing pandemic situation, there have been many phishing attacks that exploit the global focus on Covid-19. According to WHO, many hackers and cyber scammers are sending fraudulent emails and WhatsApp messages to people, taking advantage of the coronavirus disease. These attacks are coming in the form of fake job offers, fabricated messages from health organizations, covid vaccine themed phishing and brand impersonation.

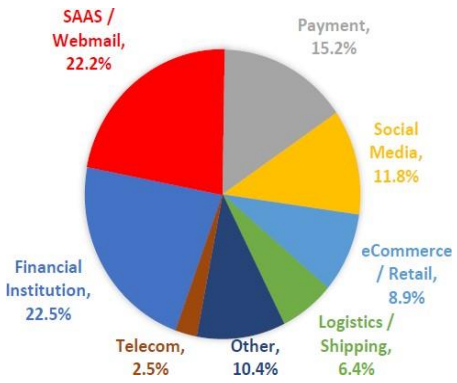


Fig. 2. Most targeted industries, 4Q 2020

Phishing (Fig.1) is a cyber-attack. The goal is to retrievepersonal information of the recipient by making them believe that the message is something they want or need i.e., request from the bank etc. Phishing employs two phishing techniques malware and deceptive based phishing. The attack can occur in two ways either by receiving suspicious email that led to fraud site or by users accessing links that go directlyto a phishing website. In general, two approaches are employed in identifying the phishing sites. The first one is based on blacklist. It worksby comparing the requested URL with those in that list butthis approach has a drawback that is it cannot identify a new fraud site which has been created within a fraction ofsecond. The latter approach is heuristic based approach. Inthis approach several features are collected from various website to classify it into either phishing or legitimate.

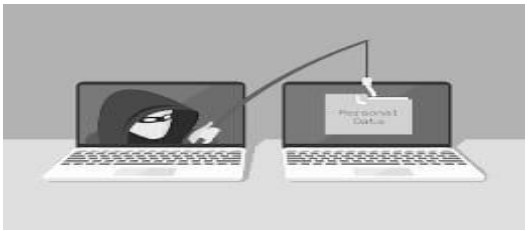


Fig.1. Phishing

My paper resolves all the drawbacks of the existing system. This project employs machine learning techniquefor predic- tion task and supervised learning algorithm namely random forest technique for exploring results. Supervised learning is a type of machine learning technique in which labelled training data are used to trainthe machine and on the basis of that data it predicts the output. Random forest is used for classification and is a machine learning algorithm that belongs to supervised learning technique. For the successful detection of phishing site, it should detect in less time and with high accuracy.

Prediction and prevention of phishing attack is very crucial step towards safeguarding online transaction. Theaim is to develop a model to safeguard users from phishing attacks. This can be done using unique features of phishing websites. The goal of our paper is to developa web application which notifies the user when it detectsa phishing site.

CHAPTER 2:

INTRODUCTION TO MACHINE LEARNING IN PENTESTING

Currently, machine learning techniques are some of the hottest trends in information technology. They impact every aspect of our lives, and they affect every industry and field. Machine learning is a cyber weapon for information security professionals. In this book, readers will not only explore the fundamentals behind machine learning techniques, but will also learn the secrets to building a fully functional machine learning security system. We will not stop at building defensive layers; we will illustrate how to build offensive tools to attack and bypass security defenses. By the end of this book, you will be able to bypass machine learning security systems and use the models constructed in penetration testing (pentesting) missions.

In this chapter, we will cover:

- Machine learning models and algorithms
- Performance evaluation metrics
- Dimensionality reduction
- Ensemble learning
- Machine learning development environments and Python libraries
- Machine learning in penetration testing – promises and challenges

Technical requirements

In this chapter, we are going to build a development environment. Therefore, we are going to install the following Python machine learning libraries:

- NumPy
- SciPy
- Tensor Flow
- Keras
- pandas
- MatplotLib
- scikit-learn
- NLTK
- Theano

2.1 Artificial intelligence and machine learning

Making a machine think like a human is one of the oldest dreams. Machine learning techniques are used to help make predictions based on experiences and data.

Machine learning models and algorithms

In order to teach machines how to solve a large number of problems by themselves, we need to consider the different machine learning models. As you know, we need to feed the model with data; that is why machine learning models are divided, based on datasets entered (input), into four major categories: supervised learning, semi-supervised learning, unsupervised learning, and reinforcement. In this section, we are going to describe each model in a detailed way, in addition to exploring the most well-known algorithms used in every machine learning model. Before building machine learning systems, we need to know how things work underneath the surface.

Supervised

We talk about supervised machine learning when we have both the input variables and the output variables. In this case, we need to map the function (or pattern) between the two parties. The following are some of the most often used supervised machine learning algorithms.

Bayesian classifiers

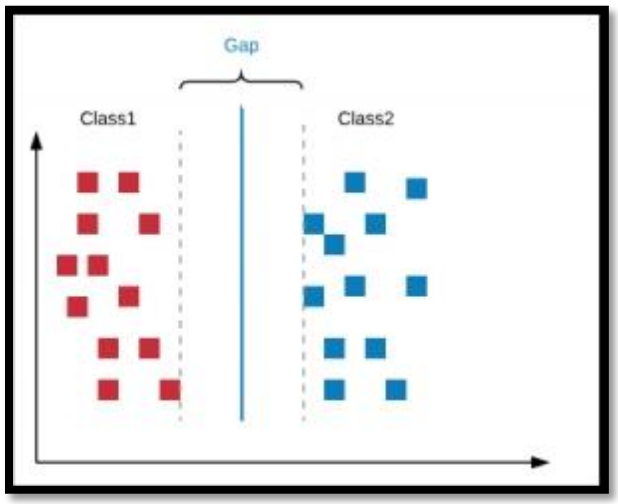
According to the *Cambridge English Dictionary*, bias is the action of supporting or opposing a particular person or thing in an unfair way, allowing personal opinions to influence your judgment. Bayesian machine learning refers to having a prior belief, and updating it later by using data. Mathematically, it is based on the Bayes formula:

$$P(c|x) = \frac{P(x|c) \times P(c)}{P(x)}$$

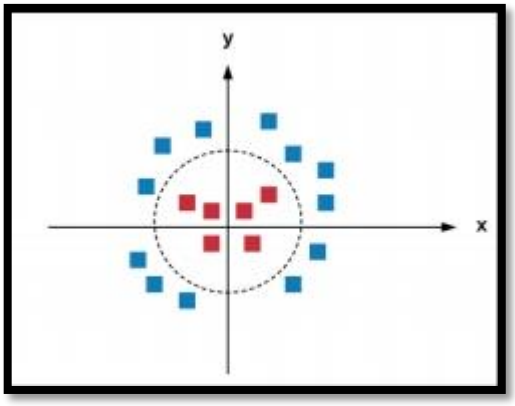
One of the simplest Bayesian problems is randomly tossing a coin and trying to predict whether the output will be heads or tails. That is why we can identify Bayesian methodology as being probabilistic. Naive Bayes is very useful when you are using a small amount of data.

Support vector machines

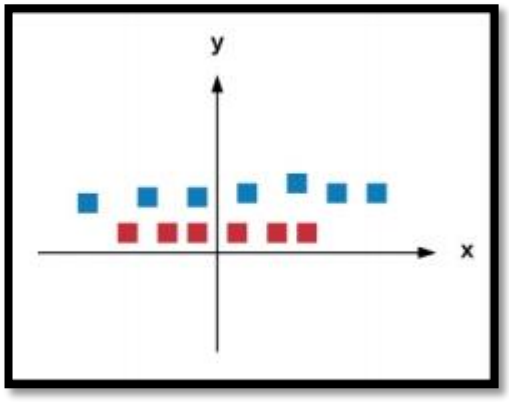
A **support vector machine** (**SVM**) is a supervised machine learning model that works by identifying a hyperplane between represented data. The data can be represented in a multidimensional space. Thus, SVMs are widely used in classification models. In an SVM, the hyperplane that best separates the different classes will be used. In some cases, when we have different hyperplanes that separate different classes, identification of the correct one will be performed thanks to something called a **margin**, or a **gap**. The margin is the nearest distance between the hyperplanes and the data positions. You can take a look at the following representation to check for the margin:



The hyperplane with the highest gap will be selected. If we choose the hyperplane with the shortest margin, we might face misclassification problems later. Don't be distracted by the previous graph; the hyperplane will not always be linear. Consider a case like the following:



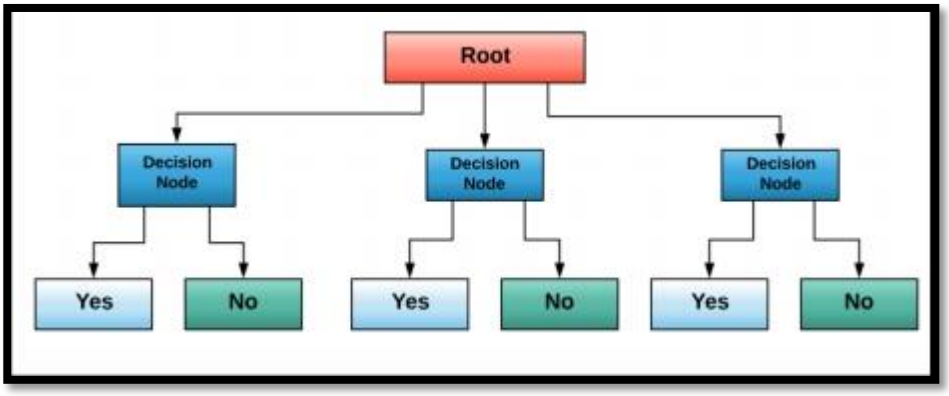
In the preceding situation, we can add a new axis, called the z axis, and apply a transformation using a kernel trick called a kernel function, where z=x^2+y^2. If you apply the transformation, the new graph will be as follows:



Now, we can identify the right hyperplane. The transformation is called a **kernel**. In the real world, finding a hyperplane is very hard. Thus, two important parameters, called regularization and gamma, play a huge role in the determination of the right hyperplane, and in every SVM classifier to obtain better accuracy in nonlinear hyperplane situations.

Decision trees

Decision trees are supervised learning algorithms used in decision making by representing data as trees upside-down with their roots at the top. The following is a graphical representation of a decision tree:



Data is represented thanks to the Iterative Dichotomiser 3 algorithm. Decision trees used in classification and regression problems are called CARTs. They were introduced by Leo Breiman.

Semi-supervised

Semi-supervised learning is an area between the two previously discussed models. In other words, if you are in a situation where you are using a small amount of labeled data in addition to unlabeled data, then you are performing semi-supervised learning. Semi- supervised learning is widely used in real-world applications, such as speech analysis, protein sequence classification, and web content classification. There are many semi- supervised methods, including generative models, low-density separation, and graph- based methods (discrete Markov Random Fields, manifold regularization, and mincut).

Unsupervised

In unsupervised learning, we don't have clear information about the output of the models. The following are some well-known unsupervised machine learning algorithms.

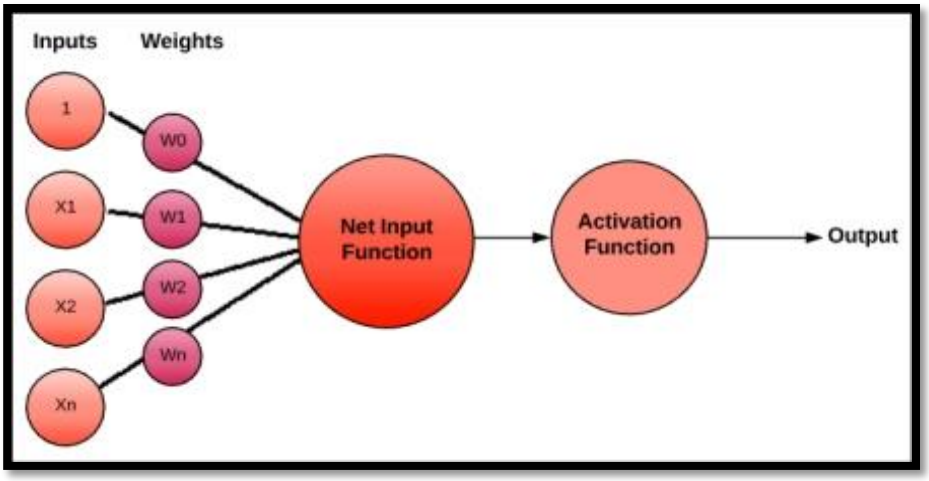
Artificial neural networks

Artificial networks are some of the hottest applications in artificial intelligence, especially machine learning. The main aim of artificial neural networks is building models that can learn like a human mind; in other words, we try to mimic the human mind. That is why, in order to learn how to build neural network systems, we need to have a clear understanding of how a human mind actually works. The human mind is an amazing entity. The mind is composed and wired by neurons. Neurons are responsible for transferring and processing information.

We all know that the human mind can perform a lot of tasks, like hearing, seeing, tasting, and many other complicated tasks. So logically, one might think that the mind is composed of many different areas, with each area responsible for a specific task, thanks to a specific algorithm. But this is totally wrong. According to research, all of the different parts of the human mind function thanks to one algorithm, not different algorithms. This hypothesis is called **the one algorithm hypothesis**.

Now we know that the mind works by using one algorithm. But what is this algorithm? How is it used? How is information processed with it?

To answer the preceding questions, we need to look at the logical representation of a neuron. The artificial representation of a human neuron is called a **perceptron**. A perceptron is represented by the following graph:



There are many **Activation Functions** used. You can view them as logical gates:

- **Step function**: A predefined threshold value.
- **Sigmoid function**:

$$f(x) = \frac{1}{1 + e^{-x}}$$

- **Tanh function**:

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

- **ReL.u function**:

$$f(x) = \begin{cases} 0 for x < 0 \\ x for x >= 0 \end{cases}$$

Many fully connected perceptrons comprise what we call a **Multi-Layer Perceptron** (**MLP**) network. A typical neural network contains the following:

- An input layer
- Hidden layers
- Output layers

We will discuss the term **deep learning** once we have more than three hidden layers. There are many types of deep learning networks used in the world:

- **Convolutional neural networks** (**CNNs**)
- **Recursive neural networks** (**RNNs**)
- **Long short-term memory** (**LSTM**)
- Shallow neural networks
- **Autoencoders** (**AEs**)
- Restricted Boltzmann machines

Don't worry; we will discuss the preceding algorithms in detail in future chapters.

To build deep learning models, we follow five steps, suggested by Dr. Jason Brownlee. The five steps are as follows:

1. Network definition
2. Network compiling
3. Network fitting
4. Network evaluation
5. Prediction

Linear regression

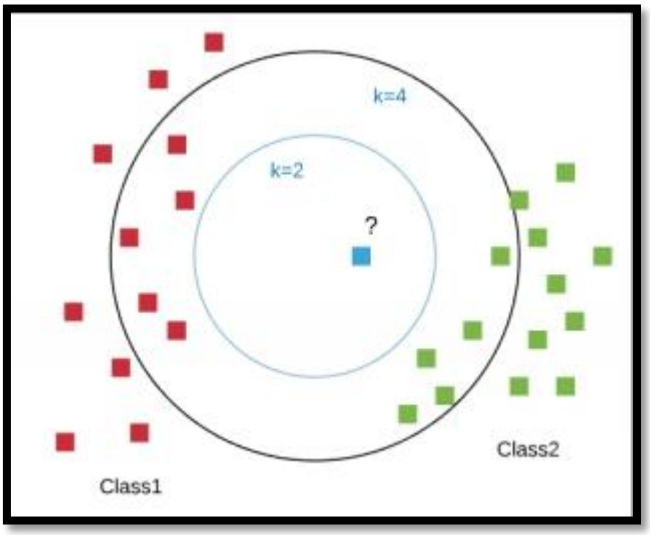
Linear regression is a statistical and machine learning technique. It is widely used to understand the relationship between inputs and outputs. We use linear regression when we have numerical values.

Logistic regression

Logistic regression is also a statistical and machine learning technique, used as a binary classifier - in other words, when the outputs are classes (yes/no, true/false, 0/1, and so on).

Clustering with k-means

k-Nearest Neighbors (**kNN**) is a well-known clustering method. It is based on finding similarities in data points, or what we call the feature similarity. Thus, this algorithm is simple, and is widely used to solve many classification problems, like recommendation systems, anomaly detection, credit ratings, and so on.However, it requires a high amount of memory. While it is a supervised learning model, it should be fed by labeled data, and the outputs are known. We only need to map the function that relates the two parties. A kNN algorithm is non-parametric. Data is represented as feature vectors. You can see it as a mathematical representation:



The classification is done like a vote; to know the class of the data selected, you must first compute the distance between the selected item and the other, training item. But how can we calculate these distances?

Generally, we have two major methods for calculating. We can use the Euclidean distance:

$$E(x,y) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2}$$

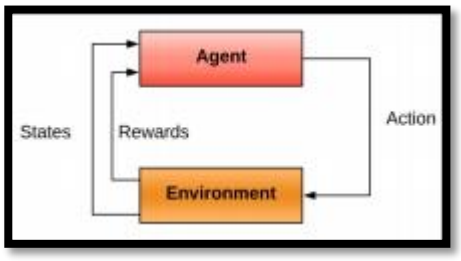
Or, we can use the cosine similarity:

$$Similarity = \cos(\varnothing) = \frac{A.B}{\|A\| \times \|B\|}$$

The second step is choosing *k* the nearest distances (*k* can be picked arbitrarily). Finally, we conduct a vote, based on a confidence level. In other words, the data will be assigned to the class with the largest probability.

Reinforcement

In the reinforcement machine learning model, the agent is in interaction with its environment, so it learns from experience, by collecting data during the process; the goal is optimizing what we call a long term **reward**. You can view it as a game with a scoring system. The following graph illustrates a reinforcement model:



Performance evaluation

Evaluation is a key step in every methodological operation. After building a product or a system, especially a machine learning model, we need to have a clear vision about its performance, to make sure that it will act as intended later on. In order to evaluate a machine learning performance, we need to use well-defined parameters and insights. To compute the different evaluation metrics, we need to use four important parameters:

- True positive
- False positive
- True negative
- False negative

The notations for the preceding parameters are as follows:

- *tp*: True positive
- *fp*: False positive
- *tn*: True negative
- *fn*: False negative

There are many machine learning evaluation metrics, such as the following:

- **Precision**: Precision, or positive predictive value, is the ratio of positive samples that are correctly classified divided by the total number of positive classified samples:

$$precision = \frac{tp}{tp + fp}$$

- **Recall**: Recall, or the true positive rate, is the ratio of true positive classifications divided by the total number of positive samples in the dataset:

$$Recall = \frac{tp}{tp + fn}$$

- **F-Score**: The F-score, or F-measure, is a measure that combines the precision and recall in one harmonic formula:

$$F - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- **Accuracy**: Accuracy is the ratio of the total correctly classified samples divided by the total number of samples. This measure is not sufficient by itself, because it is used when we have an equal number of classes.

- **Confusion matrix**: The confusion matrix is a graphical representation of the performance of a given machine learning model. It summarizes the performance of each class in a classification problem.

Dimensionality reduction

Dimensionality reduction is used to reduce the dimensionality of a dataset. It is really helpful in cases where the problem becomes intractable, when the number of variables increases. By using the term dimensionality, we are referring to the features. One of the basic reduction techniques is feature engineering.

Generally, we have many dimensionality reduction algorithms:

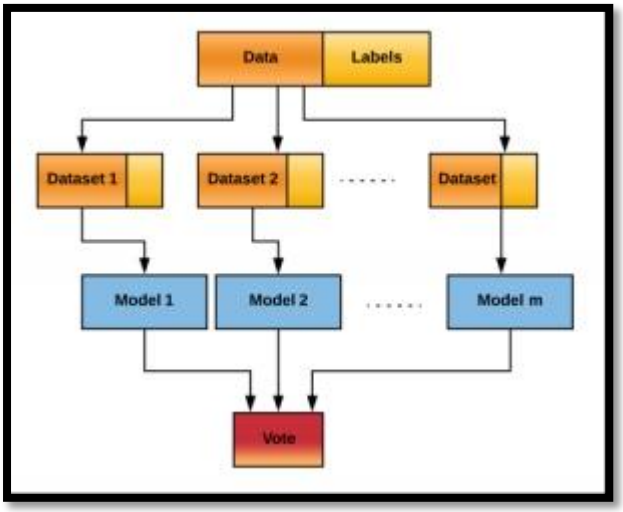
- **Low variance filter**: Dropping variables that have low variance, compared to others.
- **High correlation filter**: This identifies the variables with high correlation, by using pearson or polychoric, and selects one of them using the **Variance Inflation Factor** (**VIF**).
- **Backward feature elimination**: This is done by computing the **sum of square of error** (**SSE**) after eliminating each variable *n* times.
- **Linear Discriminant Analysis** (**LDA**): This reduces the number of dimensions, *n*, from the original to the number of classes—1number of features.
- **Principal Component Analysis** (**PCA**): This is a statistical procedure that transforms variables into a new set of variables (principle components).

Improving classification with ensemble learning

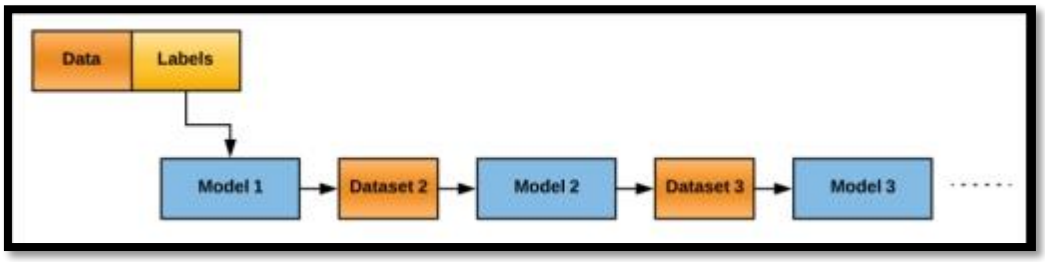
In many cases, when you build a machine learning model, you receive low accuracy and low results. In order to get good results, we can use ensemble learning techniques. This can be done by combining many machine learning techniques into one predictive model.

We can categorize ensemble learning techniques into two categories:

- **Parallel ensemble methods—** The following graph illustrates how parallel ensemble learning works:



- **Sequential ensemble methods—** The following graph illustrates how sequential ensemble learning works:



The following are the three most used ensemble learning techniques:

- **Bootstrap aggregating** (**bagging**): This involves building separate models and combining them by using model averaging techniques, like weighted average and majority vote.
- **Boosting**: This is a sequential ensemble learning technique. Gradient boosting is one of the most used boosting techniques.
- **Stacking**: This is like boosting, but it uses a new model to combine submodels.

2.2 Machine learning development environments and Python libraries

At this point, we have acquired knowledge about the fundamentals behind the most used machine learning algorithms. Starting with this section, we will go deeper, walking through a hands-on learning experience to build machine learning-based security projects. We are not going to stop there; throughout the next chapters, we will learn how malicious attackers can bypass intelligent security systems. Now, let's put what we have learned so far into practice. If you are reading this book, you probably have some experience with Python. Good for you, because you have a foundation for learning how to build machine learning security systems.

I bet you are wondering, why Python? This is a great question. According to the latest research, Python is one of the most, if not *the* most, used programming languages in data science, especially

machine learning. The most well-known machine learning libraries are for Python. Let's discover the Python libraries and utilities required to build a machine learning model.

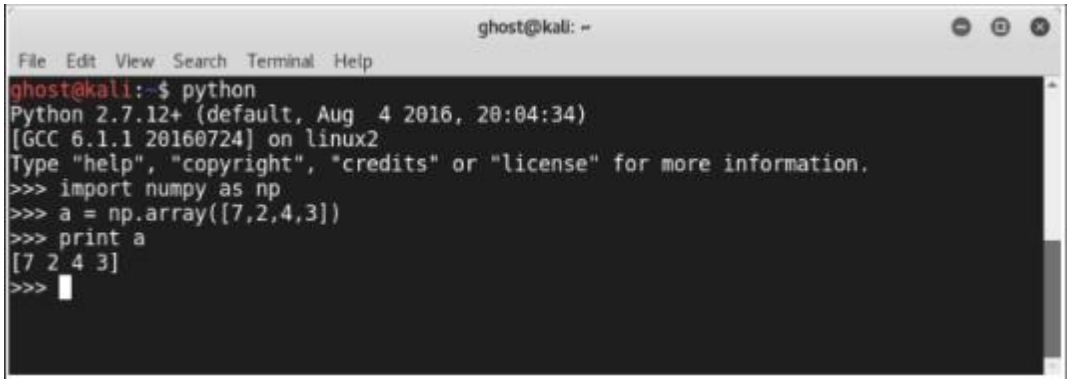
NumPy

The numerical Python library is one of the most used libraries in mathematics and logical operations on arrays. It is loaded with many linear algebra functionalities, which are very useful in machine learning. And, of course, it is open source, and is supported by many operating systems.

To install NumPy, use the pip utility by typing the following command:

```
#pip install numpy
```

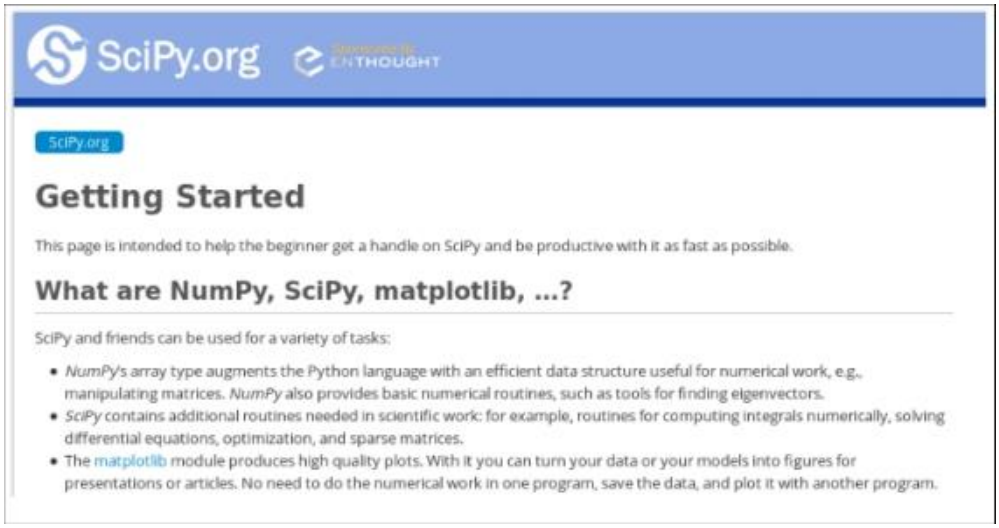
Now, you can start using it by importing it. The following script is a simple array printing example:



In addition, you can use a lot of mathematical functions, like cosine, sine, and so on.

SciPy

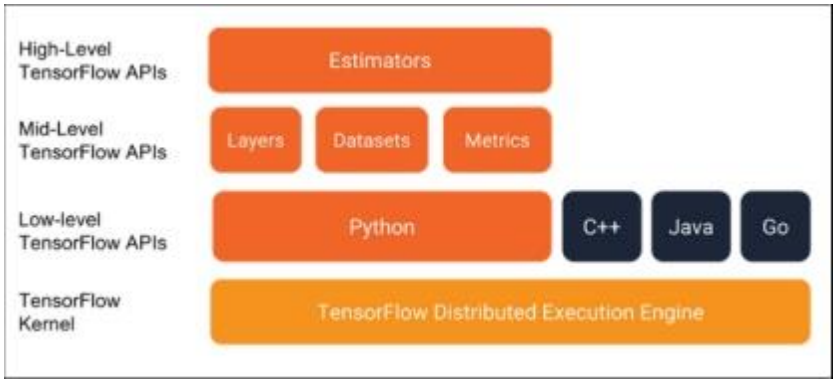
Scientific Python (**SciPy**) is like NumPy—an amazing Python package, loaded with a large number of scientific functions and utilities. For more details, you can visit https://www.scipy.org/getting-started.html:



TensorFlow

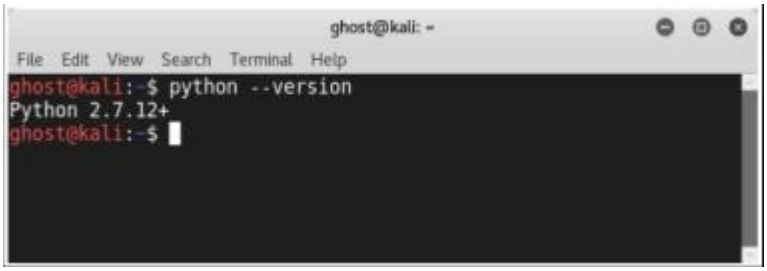
If you have been into machine learning for a while, you will have heard of TensorFlow, or have even used it to build a machine learning model or to feed artificial neural networks. It is an amazing open source project, developed essentially and supported by Google:

The following is the main architecture of TensorFlow, according to the official website:



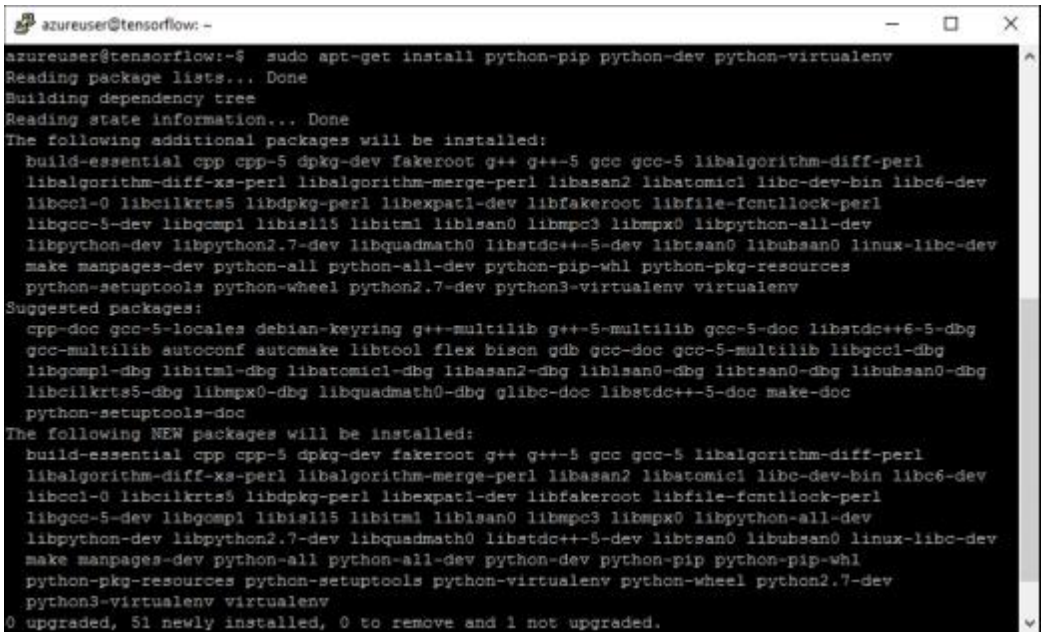
If it is your first time using TensorFlow, it is highly recommended to visit the project's official website at https://www.tensorflow.org/get_started/. Let's install it on our machine, and discover some of its functionalities. There are many possibilities for installing it; you can use native PIP, Docker, Anaconda, or Virtualenv.

Let's suppose that we are going to install it on an Ubuntu machine (it also supports the other operating systems). First, check your Python version with the python --version command:

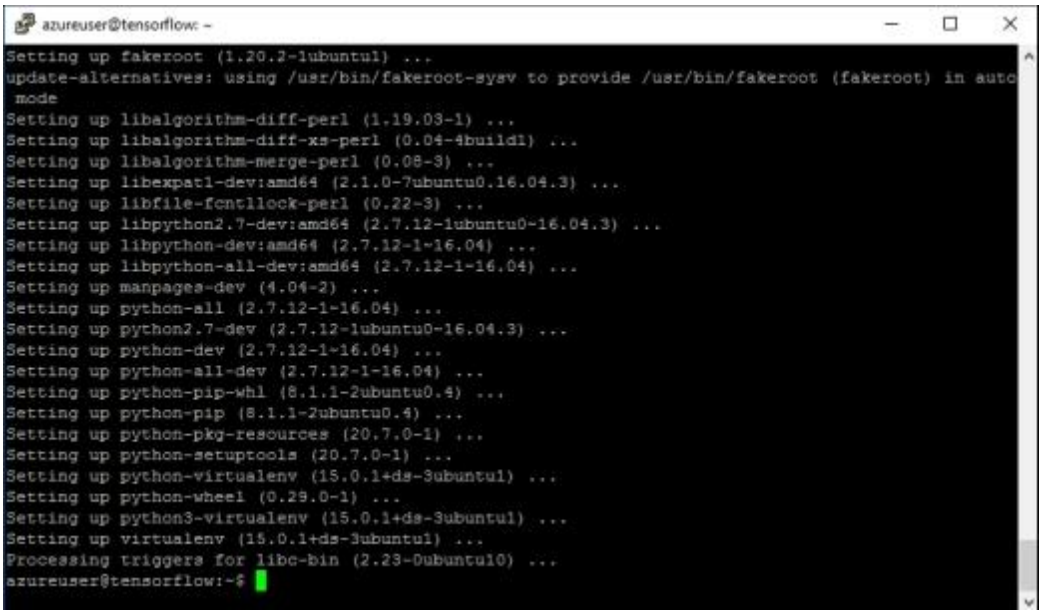


Install PIP and Virtualenv using the following command:

sudo apt-get install python-pip python-dev python-virtualenv



Now, the packages are installed

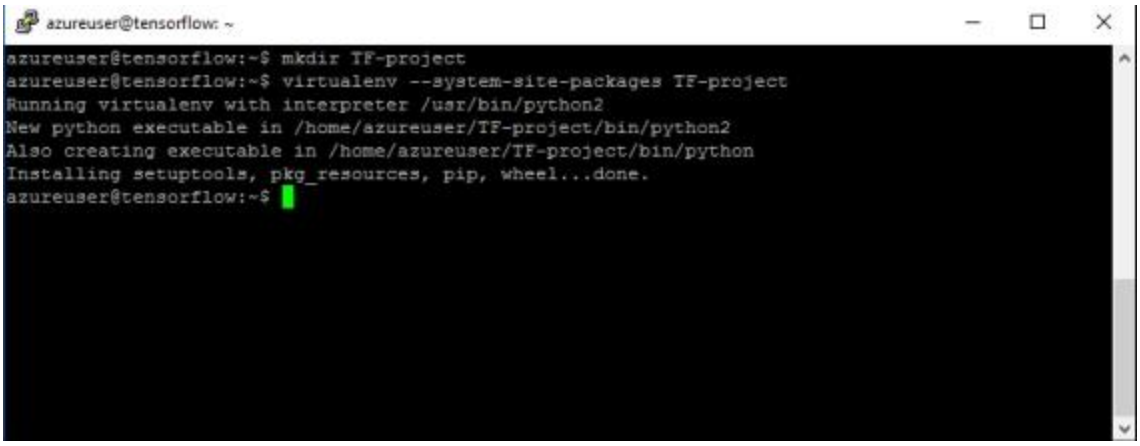


Create a new repository using the mkdir command:

#mkdir TF-project

Create a new Virtualenv by typing the following command:

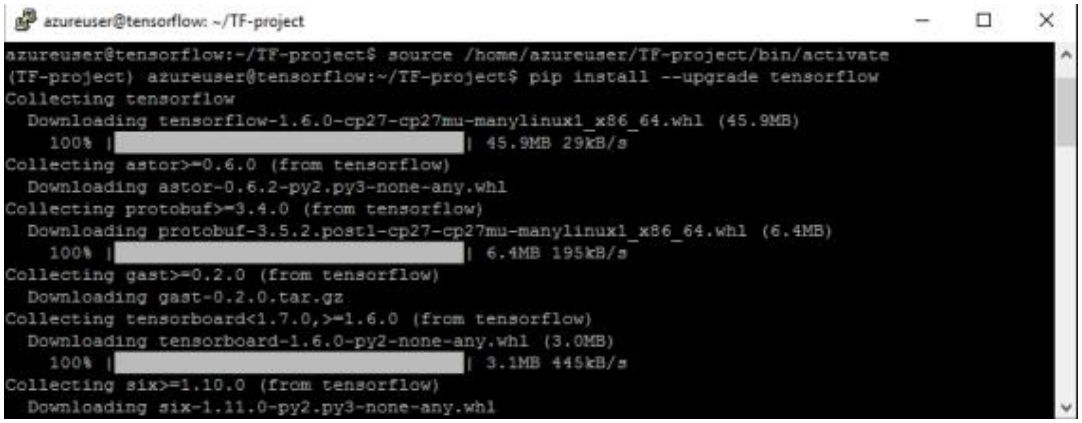
virtualenv --system-site-packages TF-project



Then, type the following command:

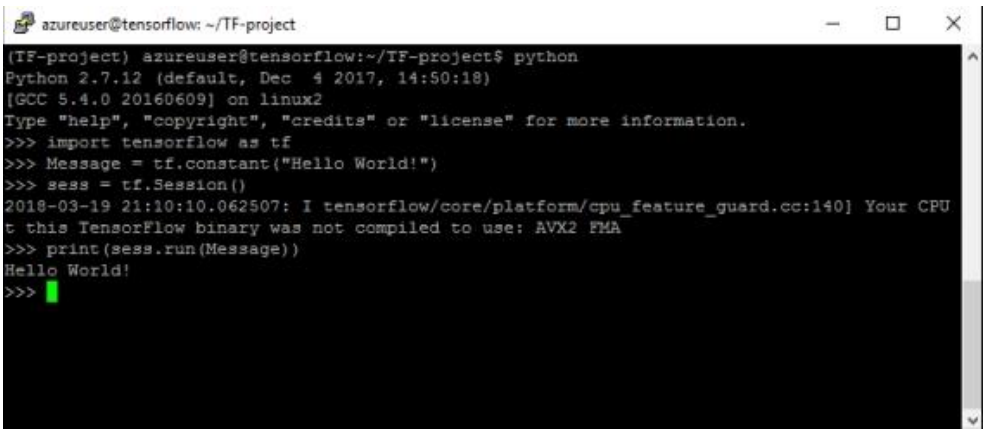
source <Directory_Here>/bin/activate

Upgrade TensorFlow by using the pip install -upgrade tensorflow command:



>>> import tensorflow as tf
>>> Message = tf.constant("Hello, world!")
>>> sess = tf.Session()
>>> print(sess.run(Message))

The following are the full steps to display a Hello World! message:



Keras

Keras is a widely used Python library for building deep learning models. It is so easy, because it is built on top of TensorFlow. The best way to build deep learning models is to follow the previously discussed steps:

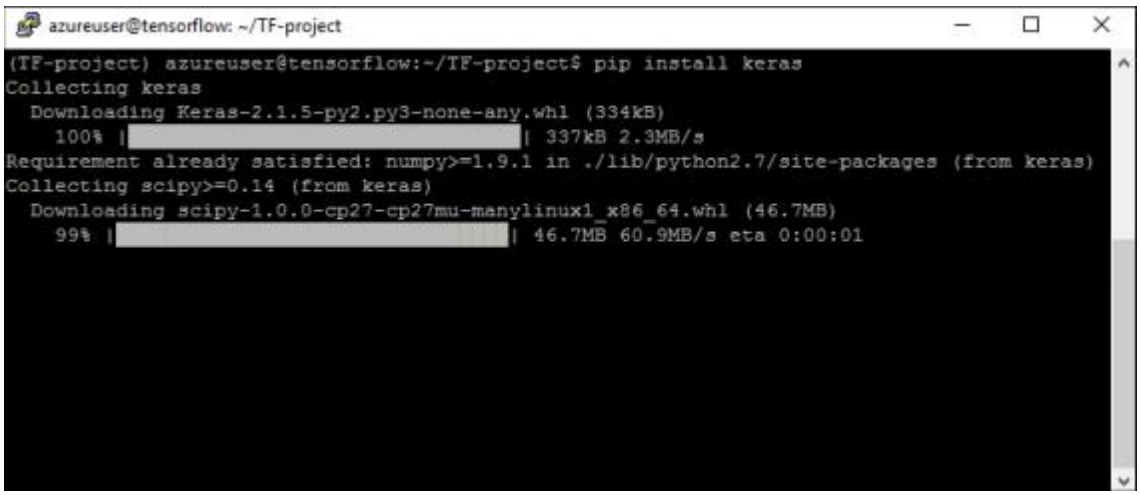
1. Loading data
2. Defining the model
3. Compiling the model
4. Fitting
5. Evaluation
6. Prediction

Before building the models, please ensure that SciPy and NumPy are preconfigured. To check, open the Python command-line interface and type, for example, the following command, to check the NumPy version:

```
>>>print numpy.__version__
```

To install Keras, just use the PIP utility:

```
$ pip install keras
```



And of course to check the version, type the following command:

```
>>> print keras.__version__
```

To import from Keras, use the following:

```
from keras import [what_to_use]
from keras.models import Sequential
from keras.layers import Dense
```

Now, we need to load data:

```
dataset = numpy.loadtxt("DATASET_HERE", delimiter=",")
```

```
I = dataset[:,0:8]
O = dataset[:,8]
#the data is splitted into Inputs (I) and Outputs (O)
```

You can use any publicly available dataset. Next, we need to create the model:

```
model = Sequential()
# N = number of neurons
# V = number of variable
model.add(Dense(N, input_dim=V, activation='relu'))
# S = number of neurons in the 2nd layer
model.add(Dense(S, activation='relu'))
model.add(Dense(1, activation='sigmoid')) # 1 output
```

Now, we need to compile the model:

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

And we need to fit the model:

```
model.fit(I, O, epochs=E, batch_size=B)
```

As discussed previously, evaluation is a key step in machine learning; so, to evaluate our model, we use:

```
scores = model.evaluate(I, O)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

To make a prediction, add the following line:

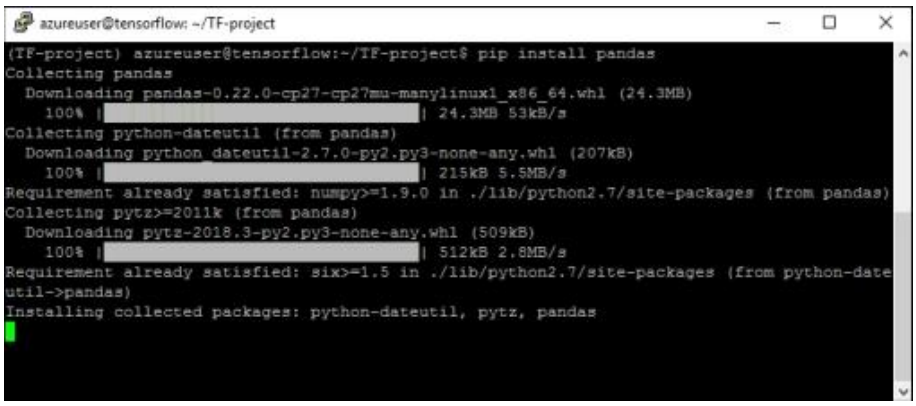
```
predictions = model.predict(Some_Input_Here)
```

pandas

pandas is an open source Python library, known for its high performance; it was developed by Wes McKinney. It quickly manipulates data. That is why it is widely used in many fields in academia and commercial activities. Like the previous packages, it is supported by many operating systems.

To install it on an Ubuntu machine, type the following command:

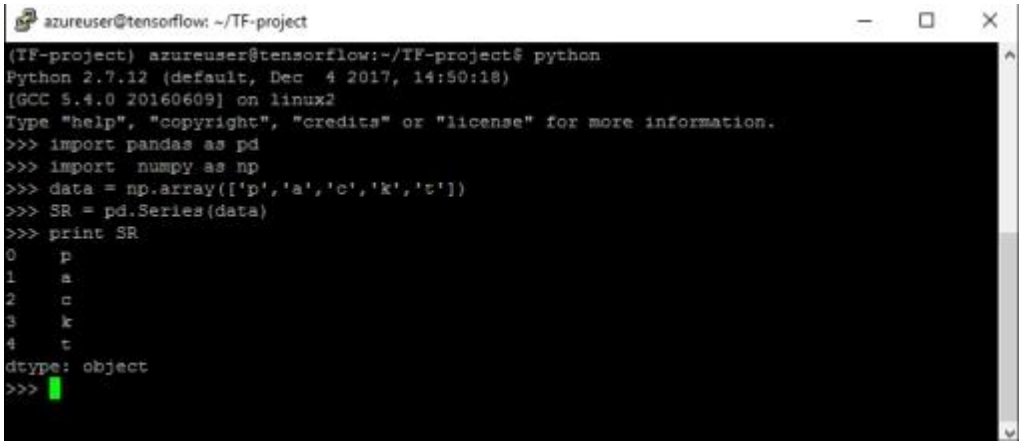
sudo apt-get install python-pandas



Basically, it manipulates three major data structures - data frames, series, and panels:

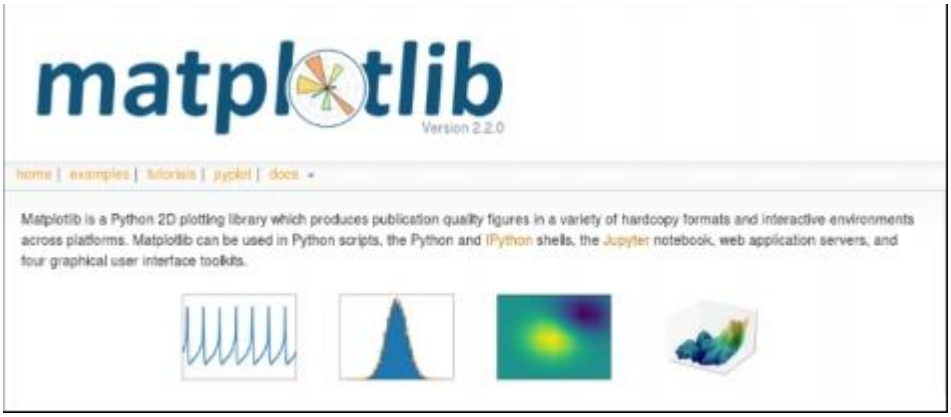
```
>>> import pandas as pd
>>> import numpy as np
>>> data = np.array(['p','a','c','k','t'])
>>> SR = pd.Series(data)
>>> print SR
```

I resumed all of the previous lines in this screenshot:



Matplotlib

As you know, visualization plays a huge role in gaining insights from data, and is also very important in machine learning. Matplotlib is a visualization library used for plotting by data scientists. You can get a clearer understanding by visiting its official website at https://matplotlib.org:



To install it on an Ubuntu machine, use the following command:

```
sudo apt-get install python3-matplotlib
```

```
azureuser@tensorflow: ~/TF-project
(TF-project) azureuser@tensorflow:~/TF-project$  sudo apt-get install python3-matplotlib
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  blt fontconfig-config fonts-lyx javascript-common libblas-common libblas3
  libfontconfig1 libgfortran3 libjbig0 libjpeg-turbo8 libjpeg8 libjs-jquery
  libjs-jquery-ui liblapack3 liblcms2-2 libtcl8.6 libtiff5 libtk8.6 libwebp5 libwebpmux1
  libxft2 libxrender1 libxss1 python-matplotlib-data python3-cycler python3-dateutil
  python3-numpy python3-pil python3-pyparsing python3-tk python3-tz tk8.6-blt2.5
  ttf-bitstream-vera x11-common
Suggested packages:
  blt-demo apache2 | lighttpd | httpd libjs-jquery-ui-docs liblcms2-utils tcl8.6 tk8.6
  dvipng ffmpeg gir1.2-gtk-3.0 ghostscript inkscape ipython3 librsvg2-common
  python-matplotlib-doc python3-cairocffi python3-gi-cairo python3-gobject python3-nose
  python3-pyqt4 python3-scipy python3-sip python3-tornado texlive-extra-utils
  texlive-latex-extra ttf-staypuft gfortran python-numpy-doc python3-dev
  python3-numpy-dbg python-pil-doc python3-pil-dbg tix python3-tk-dbg
The following NEW packages will be installed:
  blt fontconfig-config fonts-lyx javascript-common libblas-common libblas3
```

To import the required packages, use import:

```
import matplotlib.pyplot as plt
import numpy as np
```

Use this example to prepare the data:

```
x = np.linspace(0, 20, 50)
```

To plot it, add this line:

```
plt.plot(x, x, label='linear')
```

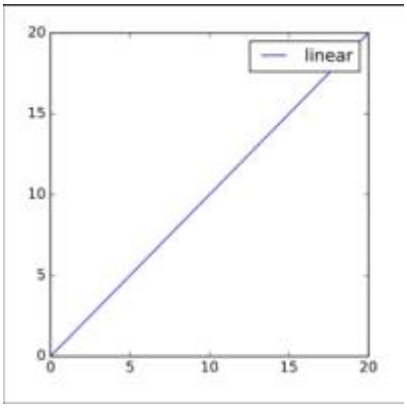
To add a legend, use the following:

```
plt.legend()
```

Now, let's show the plot:

```
plt.show()
```

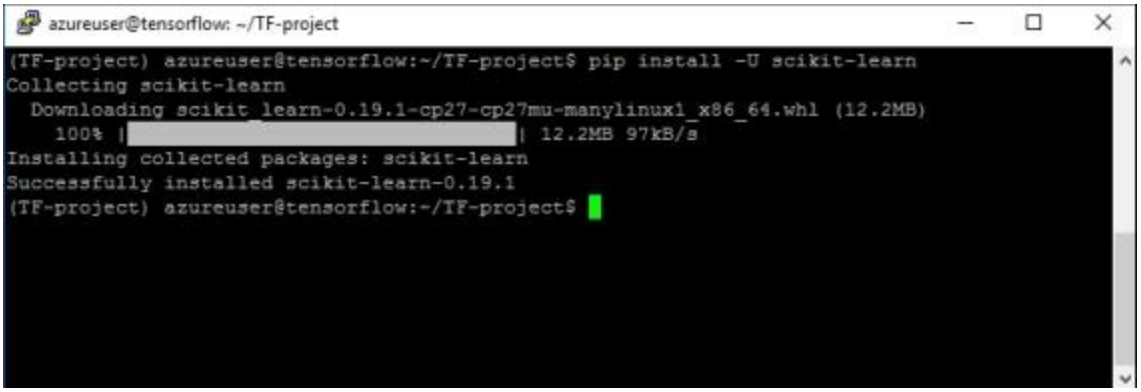
Voila! This is our plot:



scikit-learn

I highly recommend this amazing Python library. scikit-learn is fully loaded, with various capabilities, including machine learning features. The official website of scikit-learn is http://scikit-learn.org/. To download it, use PIP, as previously discussed:

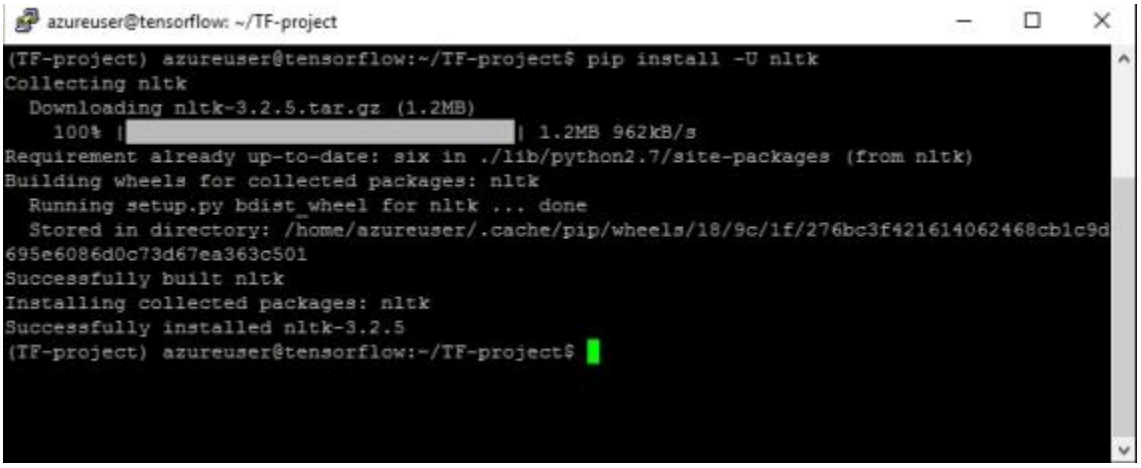
pip install -U scikit-learn



NLTK

Natural language processing is one of the most used applications in machine learning projects. NLTK is a Python package that helps developers and data scientists manage and manipulate large quantities of text. NLTK can be installed by using the following command:

pip install -U nltk

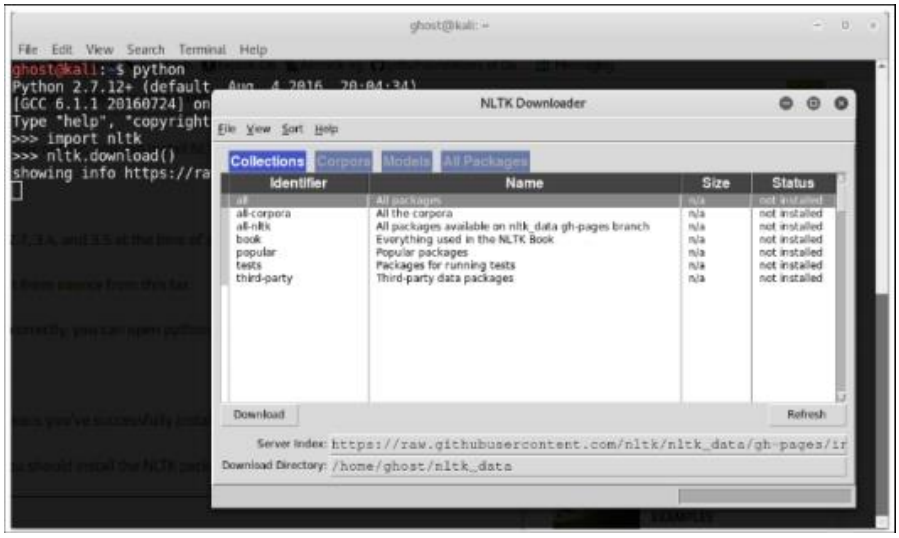


Now, import nltk:

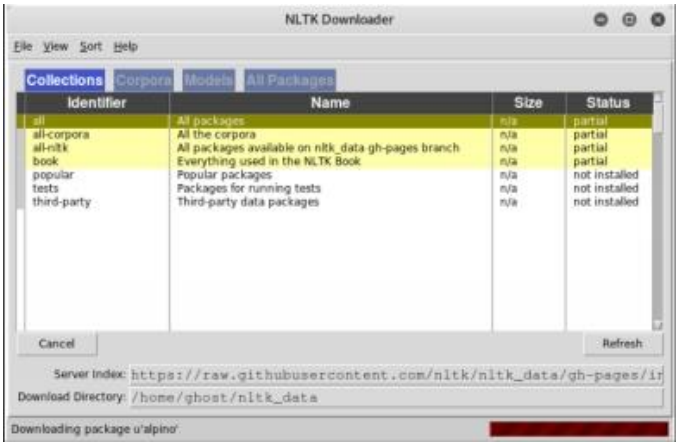
>>> import nltk

Install nltk packages with:

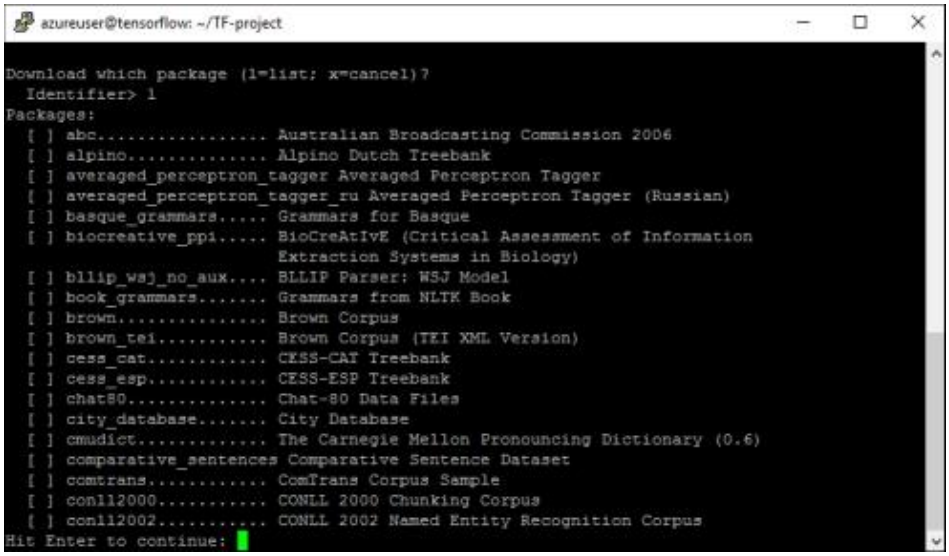
> nltk.download()



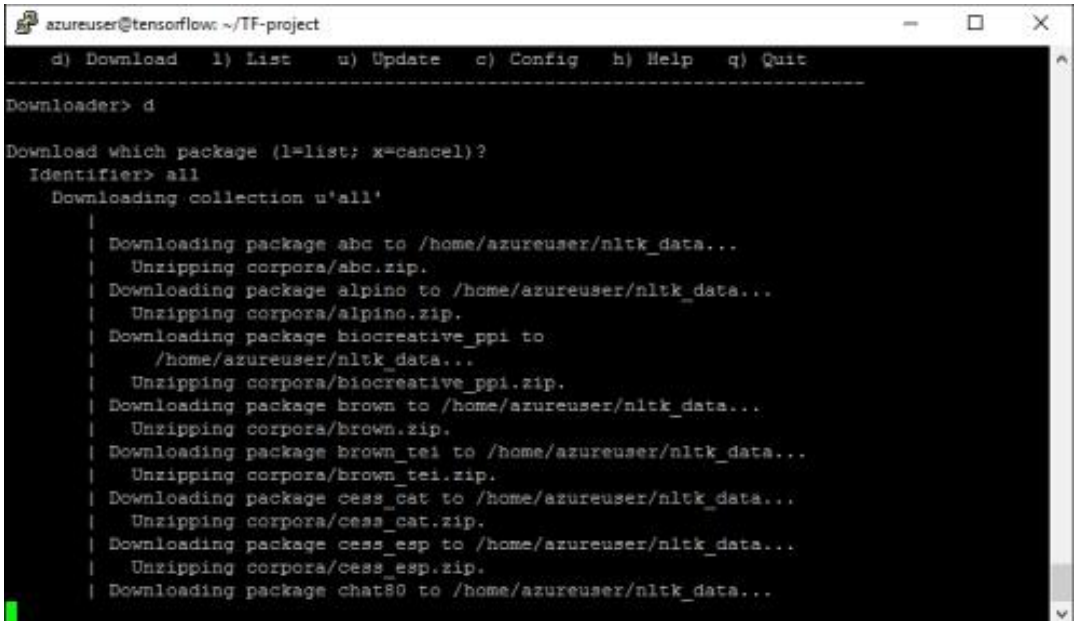
You can install all of the packages:



If you are using a command-line environment, you just need to follow the steps:



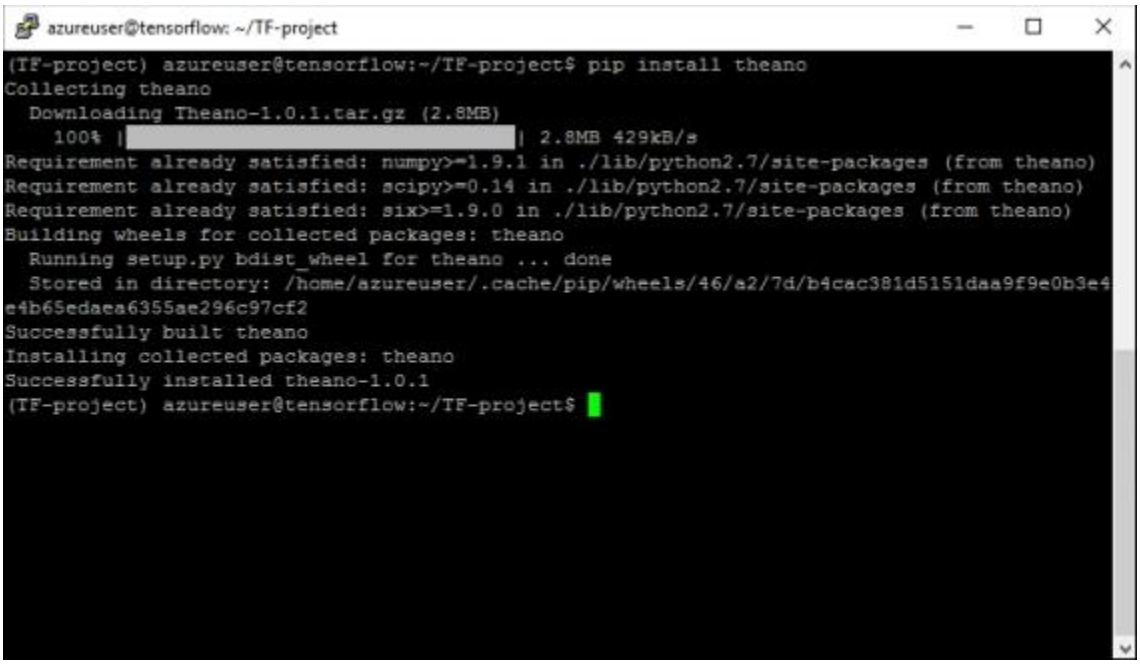
If you hit all, you will download all of the packages:



Theano

Optimization and speed are two key factors to building a machine learning model. Theano is a Python package that optimizes implementations and gives you the ability to take advantage of the GPU. To install it, use the following command:

```
pip install theano
```



To import all Theano modules, type:

```
>>> from theano import *
```

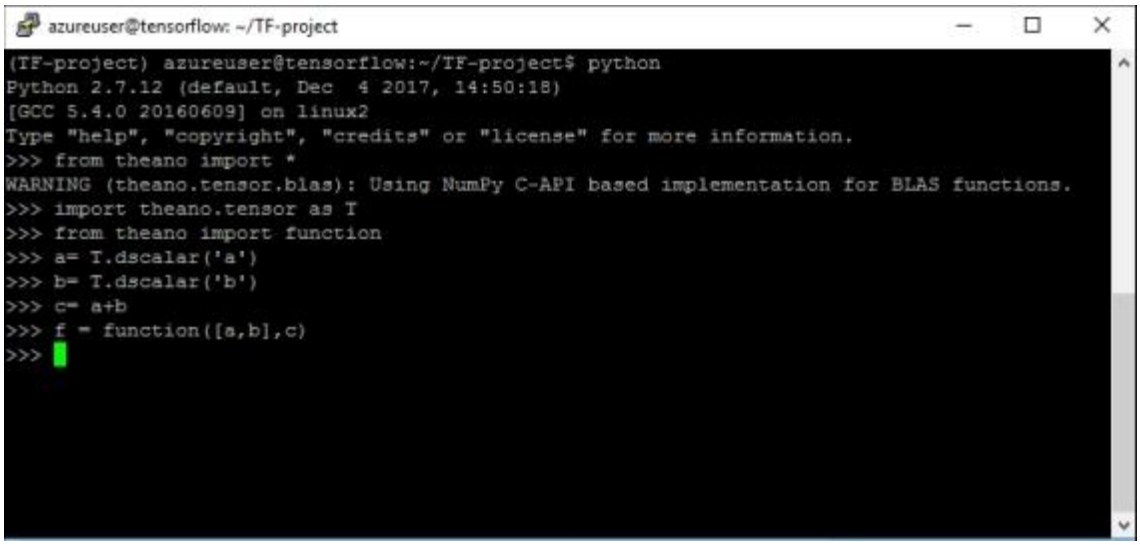
Here, we imported a sub-package called tensor:

```
>>> import theano.tensor as T
```

Let's suppose that we want to add two numbers:

```
>>> from theano import function
>>> a = T.dscalar('a')
>>> b = T.dscalar('b')
>>> c = a + b
>>> f = function([a, b], c)
```

The following are the full steps:



By now, we have acquired the fundamental skills to install and use the most common Python libraries used in machine learning projects. I assume that you have already installed all of the previous packages on your machine. In the subsequent chapters, we are going to use most of these packages to build fully working information security machine learning projects.

2.3 Machine learning in penetration testing - promises and challenges

Machine learning is now a necessary aspect of every modern project. Combining mathematics and cutting-edge optimization techniques and tools can provide amazing results. Applying machine learning and analytics to information security is a step forward in defending against advanced real-world attacks and threats.

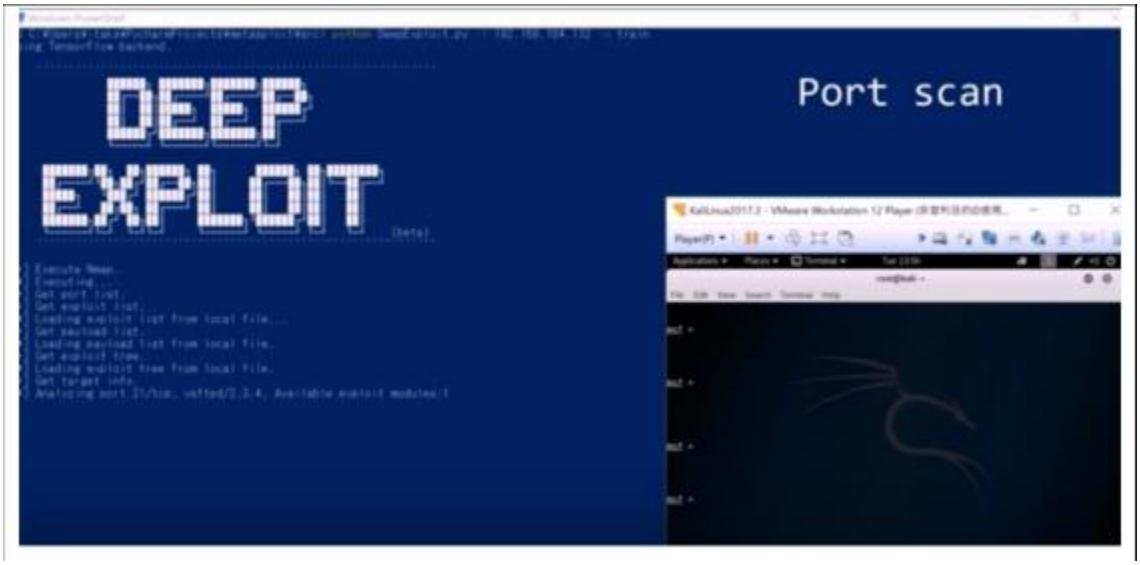
Hackers are always trying to use new, sophisticated techniques to attack modern organizations. Thus, as security professionals, we need to keep ourselves updated and deploy the required safeguards to protect assets. Many researchers have shown thousands of proposals to build defensive systems based on machine learning techniques. For example, the following are some information security models:

- **Supervised learning**:
  - Network traffic profiling
  - Spam filtering
  - Malware detection
- **Semi-supervised learning**:
  - Network anomaly detection
  - C2 detection
- **Unsupervised learning**:
  - User behavior analytics
  - Insider threat detection
  - Malware family identification

As you can see, there are great applications to help protect the valuable assets of modern organizations. But generally, black hat hackers do not use classic techniques anymore. Nowadays, the use of machine learning techniques is shifting from defensive techniques to offensive systems. We are moving from a defensive to an offensive position. In fact, building defensive layers with artificial intelligence and machine learning alone is not enough; having an understanding of how to leverage those techniques to perform ferocious attacks is needed, and should be added to your technical skills when performing penetration testing missions. Adding offensive machine learning tools to your pentesting arsenal is very useful when it comes to simulating cutting-edge attacks. While a lot of these offensive applications are still for research purposes, we will try to build our own projects, to get a glimpse of how attackers are building offensive tools and cyber weapons to attack modern companies. Maybe you can use them later, in your penetration testing operations.

Deep Exploit

Many great publicly available tools appeared lately that use machine learning capabilities to leverage penetration testing to another level. One of these tools is Deep Exploit. It was presented at black hat conference 2018. It is a fully automated penetration test tool linked with metasploit. This great tool uses uses reinforcement learning (self-learning).

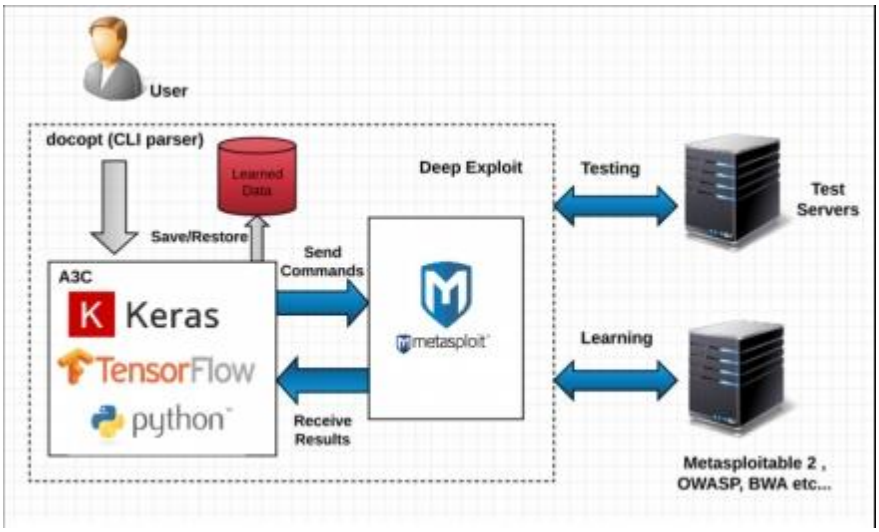


It is able to perform the following tasks:

- Intelligence gathering
- Threat modeling
- Vulnerability analysis
- Exploitation
- Post-exploitation
- Reporting

To download Deep Exploit visit its official GitHub repository: https://github.com/13o- bbr-bbq/machine_learning_security/tree/master/DeepExploit.

It is consists of a machine learning model (A3C) and metasploit. This is a high level overview of Deep Exploit architecture:



The required environment to make Deep Exploit works properly is the following:

- Kali Linux 2017.3 (guest OS on VMWare)
  - Memory: 8.0GB
  - Metasploit framework 4.16.15-dev
- Windows 10 Home 64-bit (Host OS)
  - CPU: Intel(R) Core(TM) i7-6500U 2.50GHz
  - Memory: 16.0GB
  - Python 3.6.1 (Anaconda3)
  - TensorFlow 1.4.0
  - Keras 2.1.2

Summary

Now we have learned the most commonly used machine learning techniques; before diving into practical labs, we need to acquire a fair understanding of how these models actually work. Our practical experience will start from the next chapter.

After reading this chapter, I assume that we can build our own development environment. The second chapter will show us what it takes to defend against advanced, computer- based, social engineering attacks, and we will learn how to build a smart phishing detector. Like in every chapter, we will start by learning the techniques behind the attacks, and we will walk through the practical steps in order to build a phishing detecting system.

CHAPTER 3:

PHISHING DOMAIN DETECTION

Social engineering is one of the most dangerous threats facing every individual and modern organization. Phishing is a well-known, computer-based, social engineering technique. Attackers use disguised email addresses as a weapon to target large companies. With the huge number of phishing emails received every day, companies are not able to detect all of them. That is why new techniques and safeguards are needed to defend against phishing. This chapter will present the steps required to build three different machine learning-based projects to detect phishing attempts, using cutting-edge Python machine learning libraries.

In this chapter, we will cover:

- A social engineering overview
- The steps for social engineering penetration testing
- Building a real-time phishing attack detector using different machine learning models:
    - ❖ Phishing detection with logistic regression
    - ❖ Phishing detection with decision trees
    - ❖ Spam email detection with **natural language processing** (**NLP**)

Technical requirements

In this chapter, we are going to use the following Python libraries:

- scikit-learn Python ($\geq$ 2.7 or $\geq$ 3.3)
- NumPy ($\geq$ 1.8.2)
- NLTK

3.1 Social engineering overview

Social engineering, by definition, is the psychological manipulation of a person to get useful and sensitive information from them, which can later be used to compromise a system. In other words, criminals use social engineering to gain confidential information from people, by taking advantage of human behavior.

3.1.1 Social Engineering Engagement Framework

The **Social Engineering Engagement Framework** (**SEEF**) is a framework developed by Dominique C. Brack and Alexander Bahmram. It summarizes years of experience in information security and defending against social engineering. The stakeholders of the framework are organizations, governments, and individuals (personals). Social engineering engagement management goes through three steps:

1. **Pre-engagement process**: Preparing the social engineering operation
2. **During-engagement process**: The engagement occurs
3. **Post-engagement process**: Delivering a report

There are many social engineering techniques used by criminals:

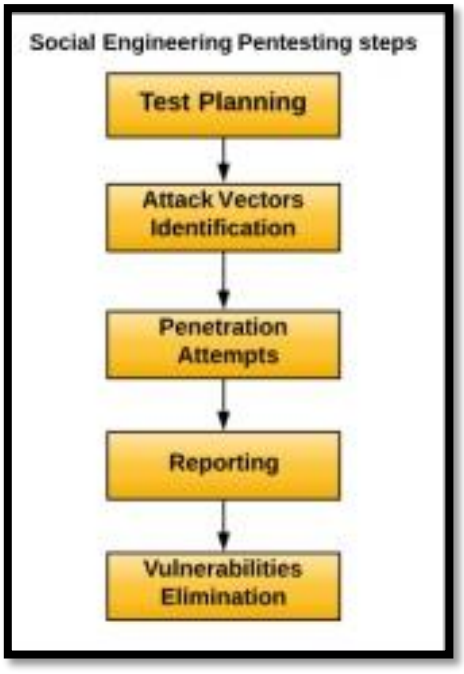
- **Baiting**: Convincing the victim to reveal information, promising him a reward or a gift.
- **Impersonation**: Pretending to be someone else.
- **Dumpster diving**: Collecting valuable information (papers with addresses, emails, and so on) from dumpsters.
- **Shoulder surfing**: Spying on other peoples' machines from behind them, while they are typing.
- **Phishing**: This is the most often used technique; it occurs when an attacker, masquerading as a trusted entity, dupes a victim into opening an email, instant message, or text message.

3.2 Steps of social engineering penetration testing

Penetration testing simulates a black hat hacker attack in order to evaluate the security posture of a company for deploying the required safeguard. Penetration testing is a methodological process, and it goes through well-defined steps. There are many types of penetration testing:

- White box pentesting
- Black box pentesting
- Grey box pentesting

To perform a social engineering penetration test, you need to follow the following steps:



3.3 Building real-time phishing attack detectors using different machine learning models

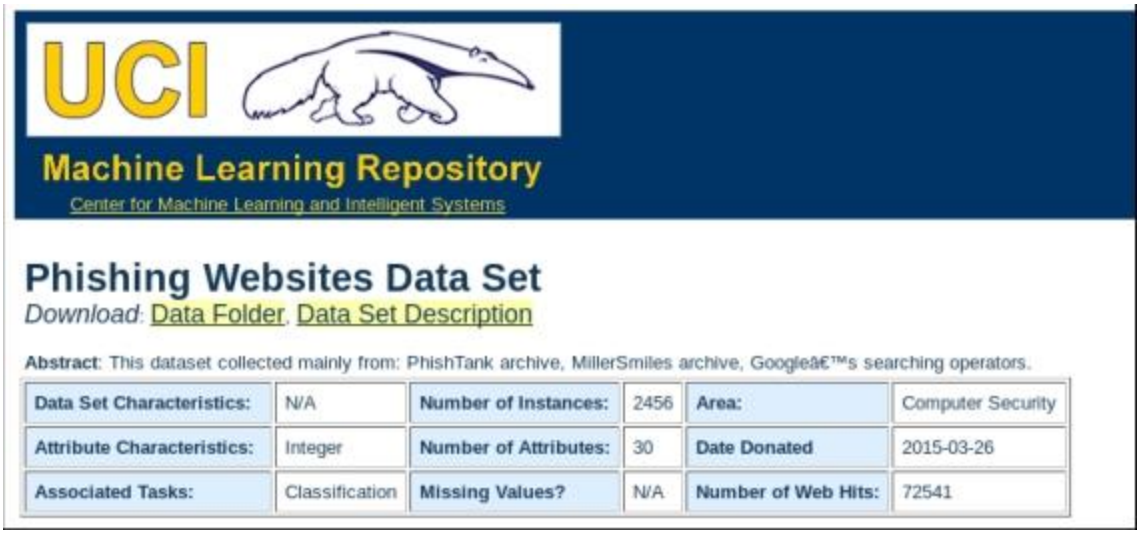
In the next sections, we are going to learn how to build machine learning phishing detectors. We will cover the following two methods:

- Phishing detection with logistic regression
- Phishing detection with decision trees

3.3.1 Phishing detection with logistic regression

In this section, we are going to build a phishing detector from scratch with a logistic regression algorithm. Logistic regression is a well-known statistical technique used to make binomial predictions (two classes).

Like in every machine learning project, we will need data to feed our machine learning model. For our model, we are going to use the UCI Machine Learning Repository (Phishing Websites Data Set). You can check it out at https://archive.ics.uci.edu/ml/datasets/ Phishing+Websites:



UCI Machine Learning Repository

Center for Machine Learning and Intelligent Systems

Phishing Websites Data Set

Download: Data Folder, Data Set Description

Abstract: This dataset collected mainly from: PhishTank archive, MillerSmiles archive, Googleâ€™s searching operators.

Data Set Characteristics:	N/A	**Number of Instances:**	2456	**Area:**	Computer Security
Attribute Characteristics:	Integer	**Number of Attributes:**	30	**Date Donated**	2015-03-26
Associated Tasks:	Classification	**Missing Values?**	N/A	**Number of Web Hits:**	72541

The dataset is provided as an arff file

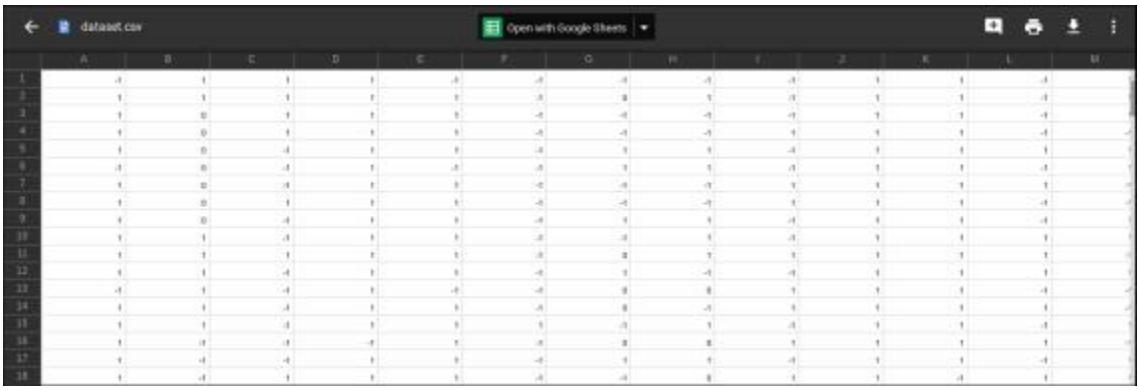
```
@relation phishing

@attribute having_IP_Address  { -1,1 }
@attribute URL_Length   { 1,0,-1 }
@attribute Shortining_Service { 1,-1 }
@attribute having_At_Symbol   { 1,-1 }
@attribute double_slash_redirecting { -1,1 }
@attribute Prefix_Suffix  { -1,1 }
@attribute having_Sub_Domain  { -1,0,1 }
@attribute SSLfinal_State  { -1,1,0 }
@attribute Domain_registeration_length { -1,1 }
@attribute Favicon { 1,-1 }
@attribute port { 1,-1 }
@attribute HTTPS_token { -1,1 }
@attribute Request_URL  { 1,-1 }
@attribute URL_of_Anchor { -1,0,1 }
@attribute Links_in_tags { 1,-1,0 }
@attribute SFH  { -1,1,0 }
@attribute Submitting_to_email { -1,1 }
@attribute Abnormal_URL { -1,1 }
@attribute Redirect  { 0,1 }
@attribute on_mouseover  { 1,-1 }
@attribute RightClick  { 1,-1 }
@attribute popUpWidnow  { 1,-1 }
@attribute Iframe { 1,-1 }
@attribute age_of_domain  { -1,1 }
@attribute DNSRecord   { -1,1 }
@attribute web_traffic  { -1,0,1 }
@attribute Page_Rank { -1,1 }
@attribute Google_Index { 1,-1 }
@attribute Links_pointing_to_page { 1,0,-1 }
@attribute Statistical_report { -1,1 }
@attribute Result  { -1,1 }
```

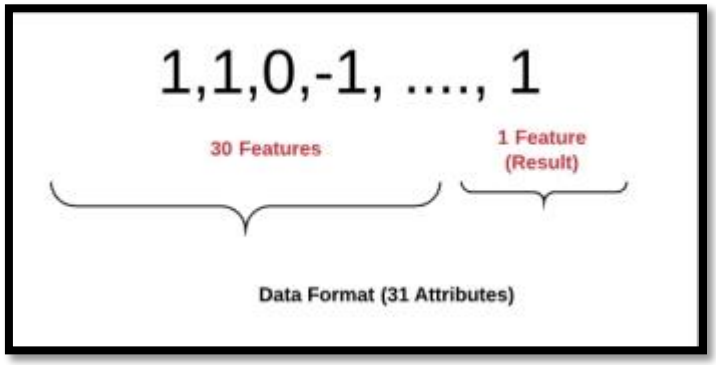
The following is a snapshot from the dataset:

```
@data
-1,1,1,1,-1,-1,-1,-1,-1,1,1,-1,1,-1,1,-1,-1,-1,0,1,1,1,1,-1,-1,-1,-1,1,1,-1,-1
1,1,1,1,1,-1,0,1,-1,1,1,-1,1,1,0,-1,-1,1,1,1,0,1,1,1,1,-1,-1,0,-1,1,1,1,-1
1,0,1,1,1,-1,-1,-1,-1,1,1,1,-1,1,1,0,-1,-1,-1,-1,-1,0,1,1,1,1,1,1,-1,1,-1,1,0,-1,-1
1,0,1,1,1,-1,-1,-1,1,1,1,1,-1,-1,0,0,-1,1,1,1,0,1,1,1,1,-1,-1,1,-1,1,-1,1,-1,1,-1
1,0,-1,1,1,-1,1,1,-1,1,1,1,1,1,0,0,-1,1,1,1,0,-1,1,-1,1,-1,-1,-1,0,-1,1,1,1,1,1
-1,0,-1,1,-1,-1,1,1,1,-1,1,1,-1,1,1,0,0,-1,-1,-1,-1,0,1,1,1,1,1,1,1,1,-1,1,1,-1,-1,1
1,0,-1,1,1,-1,-1,-1,1,1,1,1,1,1,-1,1,0,-1,-1,-1,-1,0,1,1,1,1,1,1,-1,-1,-1,1,1,0,-1,-1
1,0,1,1,1,1,-1,-1,-1,1,1,1,1,-1,-1,1,0,-1,-1,1,1,1,0,1,1,1,1,1,-1,-1,0,-1,1,1,0,1,-1
1,0,-1,1,1,-1,1,1,-1,1,1,1,-1,1,1,0,1,-1,1,1,1,0,1,1,1,1,1,-1,1,1,1,1,0,1,1
1,1,-1,1,1,1,-1,-1,1,-1,1,1,1,1,1,1,0,1,-1,1,1,1,0,1,1,1,1,1,1,-1,0,-1,1,1,0,1,-1
1,1,1,1,1,1,-1,0,1,1,1,1,1,1,-1,1,0,0,-1,-1,-1,1,0,1,1,1,1,1,-1,1,1,1,1,1,-1,-1,1,1
1,1,-1,1,1,1,-1,1,-1,-1,1,1,1,1,1,-1,-1,1,-1,-1,-1,-1,0,1,1,1,1,1,-1,-1,-1,-1,1,1,0,-1,-1
-1,1,-1,1,1,-1,-1,-1,0,0,1,1,1,1,-1,-1,-1,1,-1,1,1,1,0,-1,1,1,-1,1,1,1,-1,-1,-1,1,1,0,1,-1
1,1,-1,1,1,1,-1,0,-1,1,1,1,1,1,-1,-1,1,-1,-1,-1,1,1,1,0,1,1,1,1,1,-1,-1,0,-1,1,1,1,1,-1
1,1,-1,1,1,1,1,-1,1,-1,1,1,1,1,-1,1,1,1,1,1,0,1,1,1,1,1,1,1,-1,1,1,1,1,1,1,1,1,1,1,1
1,-1,-1,-1,1,-1,0,0,1,1,1,1,1,-1,-1,0,-1,1,1,0,1,1,1,1,1,1,-1,-1,-1,1,1,0,1,-1
1,-1,-1,1,1,-1,1,1,-1,1,1,1,-1,1,1,0,-1,-1,-1,-1,0,1,1,1,1,1,1,-1,0,-1,1,1,-1,-1
1,-1,1,1,1,1,-1,-1,0,1,1,-1,1,1,1,0,-1,-1,-1,-1,0,1,1,1,1,1,-1,1,1,1,-1,1,1,-1,-1
1,1,1,1,1,1,-1,-1,1,1,-1,1,1,1,1,1,-1,0,-1,-1,-1,0,1,1,1,1,1,1,-1,1,1,1,-1,1,-1,1
1,1,1,1,1,1,-1,-1,1,-1,1,1,1,1,1,1,0,0,-1,-1,-1,0,-1,-1,-1,-1,1,-1,0,-1,1,0,-1,1
1,0,-1,1,1,-1,0,1,-1,1,1,1,1,1,0,0,-1,-1,-1,0,-1,1,-1,1,-1,1,1,-1,1,-1,-1,1
1,0,1,1,1,-1,0,1,1,1,1,-1,-1,1,0,-1,-1,-1,-1,0,1,1,1,1,1,-1,1,-1,-1,1,0,-1,1
1,1,1,1,1,1,-1,-1,-1,-1,1,1,1,-1,1,0,0,-1,1,1,0,1,1,1,1,1,1,1,-1,1,-1,1,1,1
1,1,1,1,1,1,-1,1,0,-1,1,1,1,1,1,0,0,-1,1,1,0,1,1,1,1,1,1,1,-1,1,-1,1,1
1,-1,-1,-1,1,-1,1,1,-1,1,1,-1,-1,0,0,-1,1,1,0,1,1,1,1,1,1,-1,-1,1,0,1,1
1,-1,1,1,1,1,-1,0,1,-1,1,1,1,1,1,1,0,-1,1,1,0,1,1,1,1,1,-1,1,1,-1,1,0,1,1
1,-1,1,1,1,1,-1,0,-1,1,1,1,1,-1,-1,1,-1,-1,-1,-1,-1,0,1,1,1,1,1,1,1,0,-1,1,-1,-1,-1
1,-1,-1,1,1,1,1,-1,1,1,1,1,1,1,-1,1,1,0,-1,-1,-1,0,1,1,1,1,1,1,-1,0,-1,1,0,-1,1
1,-1,-1,1,-1,1,-1,1,-1,1,1,1,1,1,1,1,0,-1,1,1,1,1,1,1,1,-1,-1,1,-1,1,-1,1,1
1,-1,1,1,1,1,-1,-1,1,-1,1,1,1,1,1,1,1,0,-1,1,1,0,1,1,1,1,-1,1,1,1,1,1,0,1,1
1,-1,1,1,1,1,-1,-1,1,-1,1,1,-1,1,1,0,1,-1,1,1,1,0,1,1,1,1,-1,-1,-1,1,1,0,1,1
1,-1,1,1,1,1,-1,-1,1,-1,-1,1,-1,1,1,0,-1,-1,-1,-1,0,-1,1,-1,-1,1,-1,1,1,1,1,0,-1,1
1,-1,1,1,1,1,1,1,1,-1,1,1,1,1,1,1,1,1,-1,-1,0,1,1,1,1,1,1,1,-1,-1,1,-1,-1,1
1,0,1,1,1,-1,-1,1,-1,1,1,1,1,1,0,1,-1,-1,-1,0,1,1,1,1,1,1,1,-1,1,0,-1,1
```

For better manipulation, we have organized the dataset into a csv file:



As you probably noticed from the attributes, each line of the dataset is represented in the following format – *{30 Attributes (having_IP_Address URL_Length, abnormal_URL and so on)} + {1 Attribute (Result)}*:



For our model, we are going to import two machine learning libraries, NumPy and scikit- learn, which we already installed in Chapter 1, *Introduction to Machine Learning in Pentesting*.

Let's open the Python environment and load the required libraries:

```
>>> import numpy as np
>>> from sklearn import *
>>> from sklearn.linear_model import LogisticRegression
>>> from sklearn.metrics import accuracy_score
```

Next, load the data:

```
training_data = np.genfromtxt('dataset.csv', delimiter=',', dtype=np.int32)
```

Identify the inputs (all of the attributes, except for the last one) and the outputs (the last attribute):

```
>>> inputs = training_data[:,:-1]
>>> outputs = training_data[:, -1]
```

In the previous chapter, we discussed how we need to divide the dataset into training data and testing data:

```
training_inputs = inputs[:2000]
training_outputs = outputs[:2000]
testing_inputs = inputs[2000:]
testing_outputs = outputs[2000:]
```

Create the scikit-learn logistic regression classifier:

```
classifier = LogisticRegression()
```

Train the classifier:

```
classifier.fit(training_inputs, training_outputs)
```

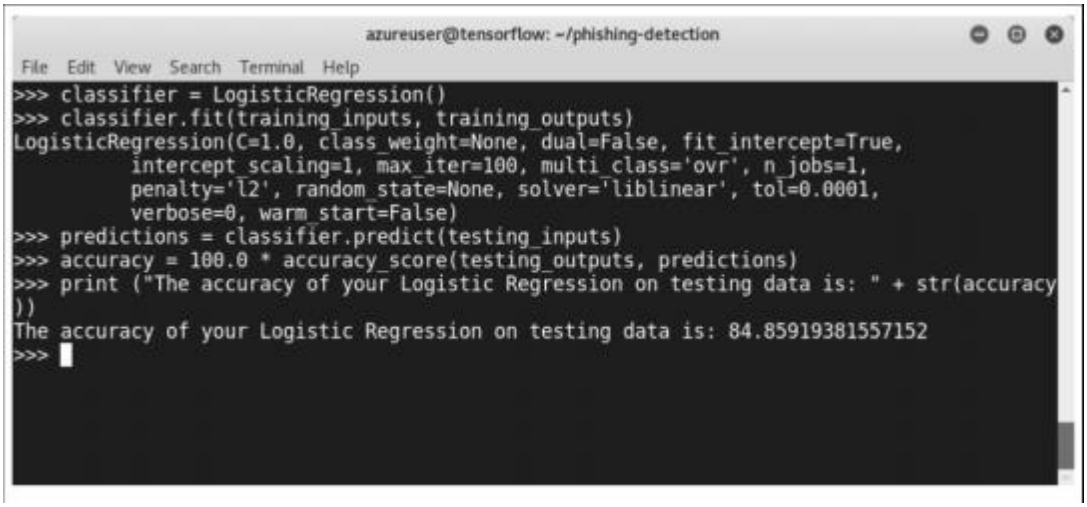
Make predictions:

```
predictions = classifier.predict(testing_inputs)
```

Let's print out the accuracy of our phishing detector model:

```
accuracy = 100.0 * accuracy_score(testing_outputs, predictions)

print ("The accuracy of your Logistic Regression on testing data
is: " +  str(accuracy))
```



The accuracy of our model is approximately 85%. This is a good accuracy, since our model detected 85 phishing URLs out of 100. But let's try to make an even better model with decision trees, using the same data.

3.3.2 Phishing detection with decision trees

To build the second model, we are going to use the same machine learning libraries, so there is no need to import them again. However, we are going to import the decision tree classifier from sklearn:

```
>>> from sklearn import tree
```

Create the tree.DecisionTreeClassifier() scikit-learn classifier:

```
classifier = tree.DecisionTreeClassifier()
```

Train the model:

```
classifier.fit(training_inputs, training_outputs)
```

Compute the predictions:

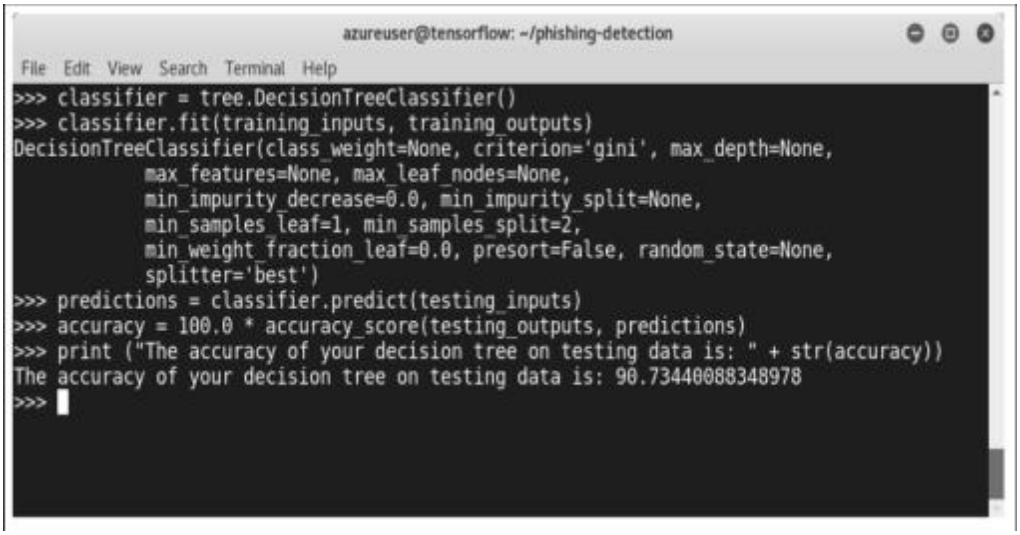
```
predictions = classifier.predict(testing_inputs)
```

Calculate the accuracy:

```
accuracy = 100.0 * accuracy_score(testing_outputs, predictions)
```

Then, print out the results:

```
print ("The accuracy of your decision tree on testing data is: " + str(accuracy))
```



The accuracy of the second model is approximately 90.4%, which is a great result, compared to the first model. We have now learned how to build two phishing detectors, using two machine learning techniques.

3.4 NLP in-depth overview

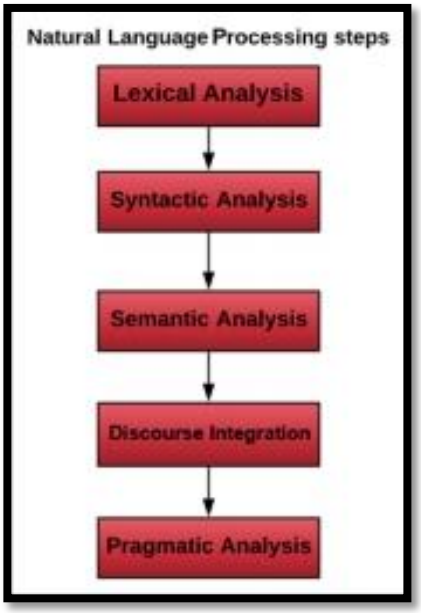
NLP is the art of analyzing and understanding human languages by machines. According to many studies, more than 75% of the used data is unstructured. Unstructured data does not have a predefined data model or not organized in a predefined manner. Emails, tweets, daily messages and even our recorded speeches are forms of unstructured data. NLP is a way for machines to analyze, understand, and derive meaning from natural language. NLP is widely used in many fields and applications, such as:

- Real-time translation
- Automatic summarization
- Sentiment analysis
- Speech recognition
- Build chatbots

Generally, there are two different components of NLP:

- **Natural Language Understanding (NLU)**: This refers to mapping input into a useful representation.
- **Natural Language Generation (NLG)**: This refers to transforming internal representations into useful representations. In other words, it is transforming data into written or spoken narrative. Written analysis for business intelligence dashboards is one of NLG applications.

Every NLP project goes through five steps. To build an NLP project the first step is identifying and analyzing the structure of words. This step involves dividing the data into paragraphs, sentences, and words. Later we analyze the words in the sentences and relationships among them. The third step involves checking the text for meaningfulness. Then, analyzing the meaning of consecutive sentences. Finally, we finish the project by the pragmatic analysis.



3.4.1 Open source NLP libraries

There are many open source Python libraries that provide the structures required to build real-world NLP applications, such as:

- Apache OpenNLP
- GATE NLP library
- Stanford NLP
- And, of course, **Natural Language Toolkit** (**NLTK**)

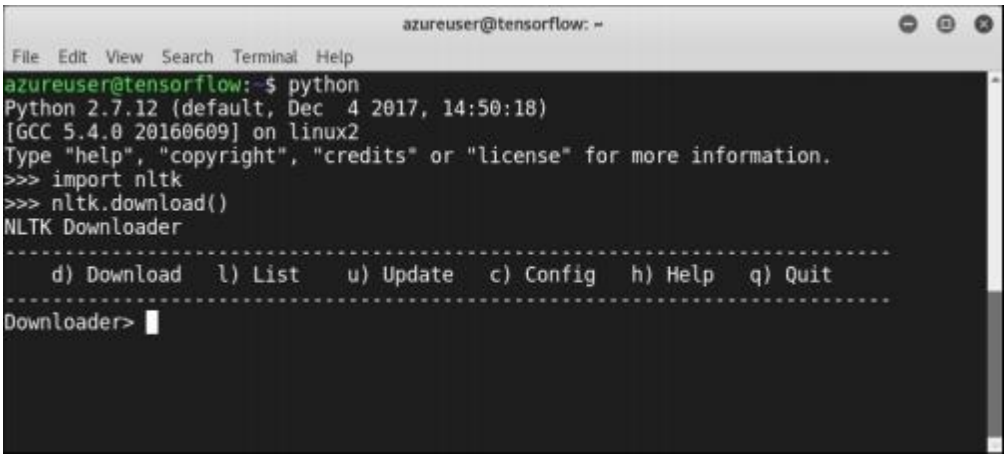
In the previous chapter, we learned how to install many open source machine learning Python libraries, including the NLTK. Let's fire up our Linux machine and try some hands- on techniques.

Open the Python terminal and import nltk:

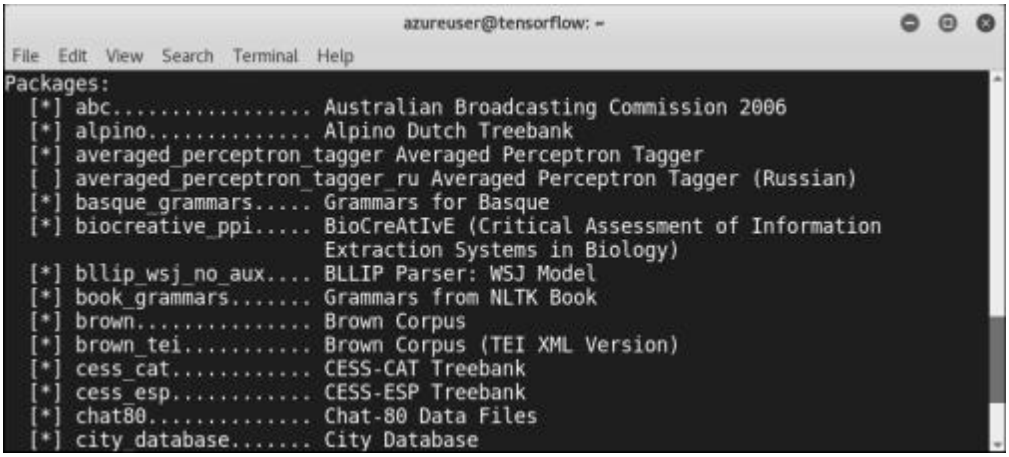
```
>>> import nltk
```

Download a book type, as follows:

```
>>> nltk.download()
```

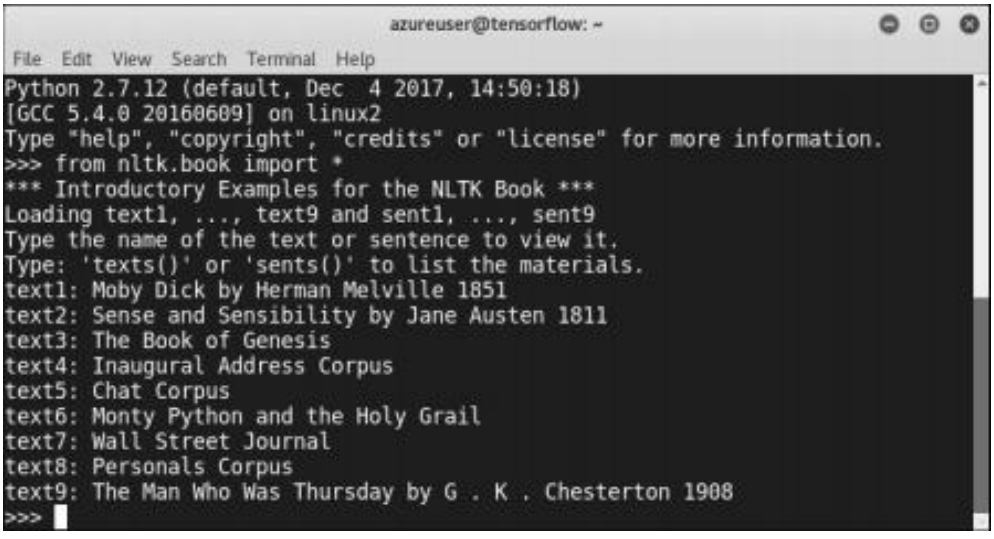


If you want to list the available resources that we already downloaded in the previous chapter, type l:



You can also type:

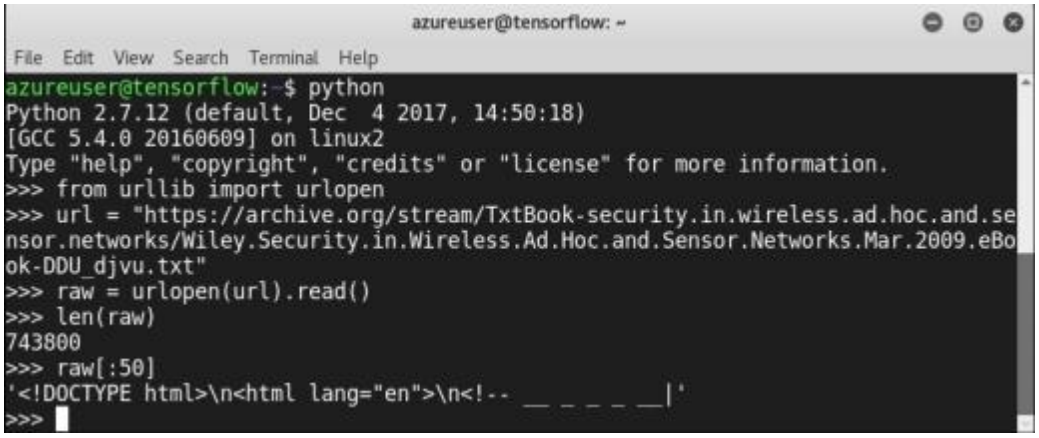
>> **from nltk.book import ***



To get text from a link, it is recommended to use the urllib module to crawl a website:

```
>>> from urllib import urlopen
>>> url = http://www.URL_HERE/file.txt
```

As a demonstration, we are going to load a text called Security.in.Wireless.Ad.Hoc.and.Sensor.Networks:



We crawled the text file, and used len to check its length and raw[:50] to display some content. As you can see from the screenshot, the text contains a lot of symbols that are useless for our projects. To get only what we need, we use **tokenization**:

```
>>> tokens = nltk.word_tokenize(raw)

>>> len(tokens)

>>> tokens[:10]
```

To summarize what we learned in the previous section, we saw how to download a web page, tokenize the text, and normalize the words.

3.4.2 Spam detection with NLTK

Now it is time to build our spam detector using the NLTK. The principle of this type of classifier is simple; we need to detect the words used by spammers. We are going to build a spam/non-spam binary classifier using Python and the nltk library, to detect whether or not an email is spam. First, we need to import the library as usual:

```
>>> import nltk
```

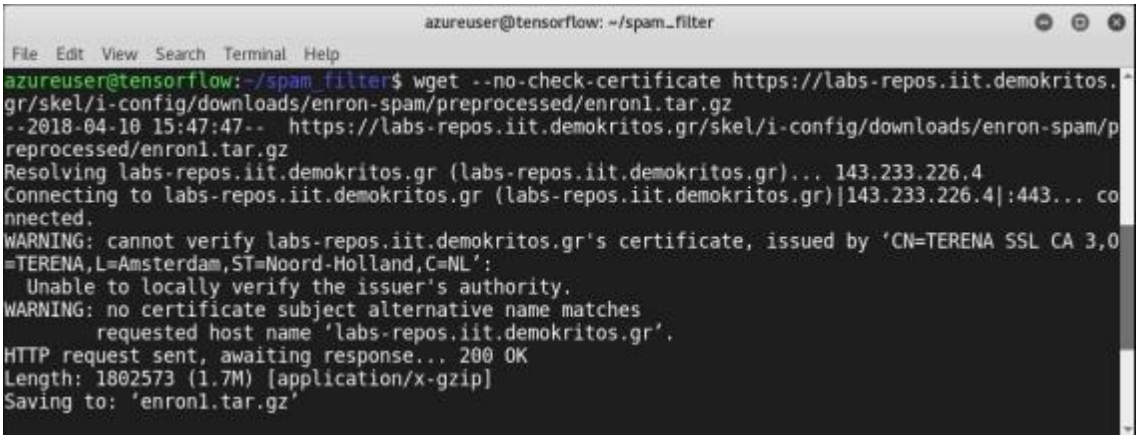
We need to load data and feed our model with an emails dataset. To achieve that, we can use the dataset delivered by the **Internet CONtent FIltering Group**. You can visit the website at https://labs-repos.iit.demokritos.gr/skel/i-config/:

Basically, the website provides four datasets:

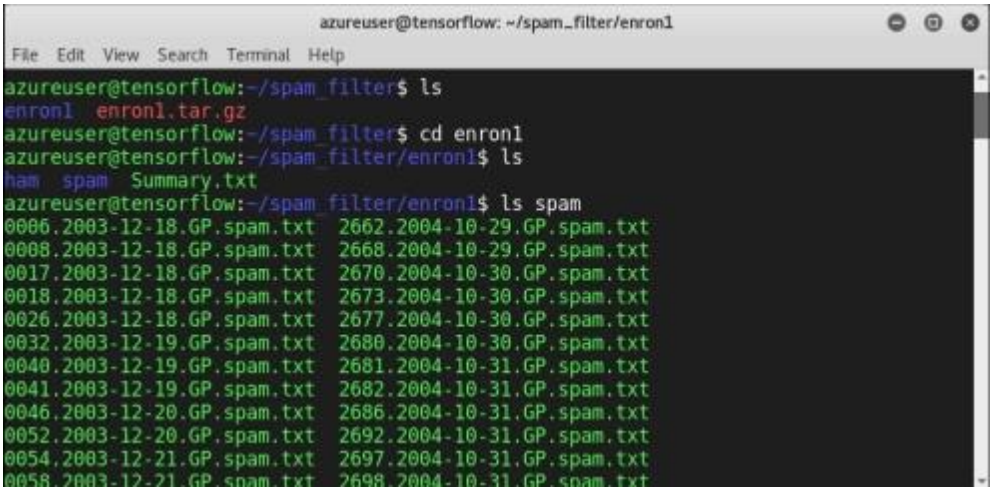
- Ling-spam
- PU1
- PU123A
- Enron-spam

For our project, we are going to use the Enron-spam dataset:

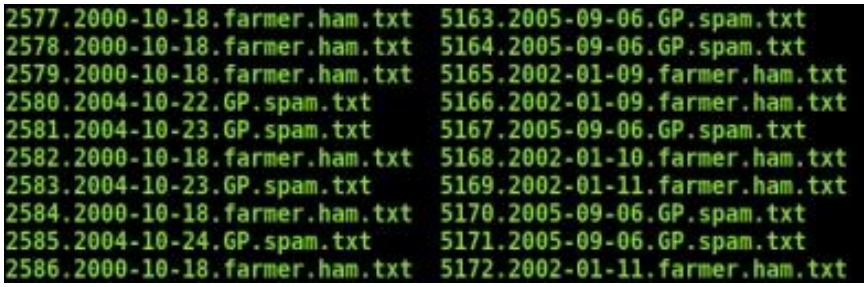
Let's download the dataset using the wget command:



Extract the tar.gz file by using the tar -xzf enron1.tar.gz command:



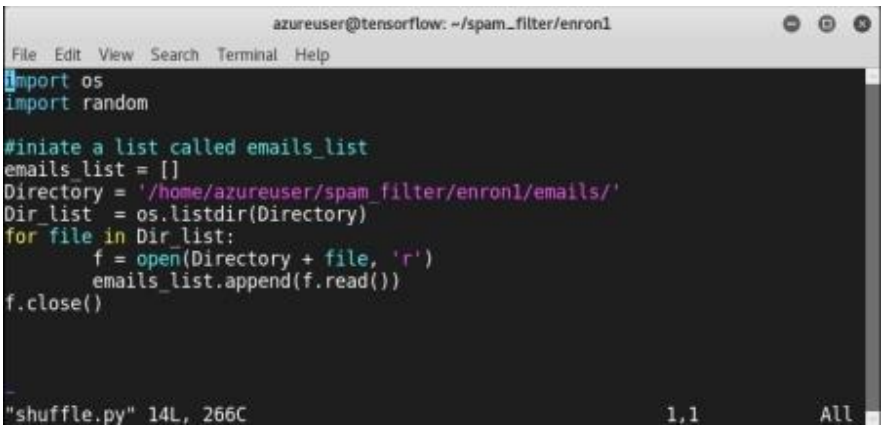
Shuffle the cp spam/* emails && cp ham/* emails object:



To shuffle the emails, let's write a small Python script, Shuffle.py, to do the job:

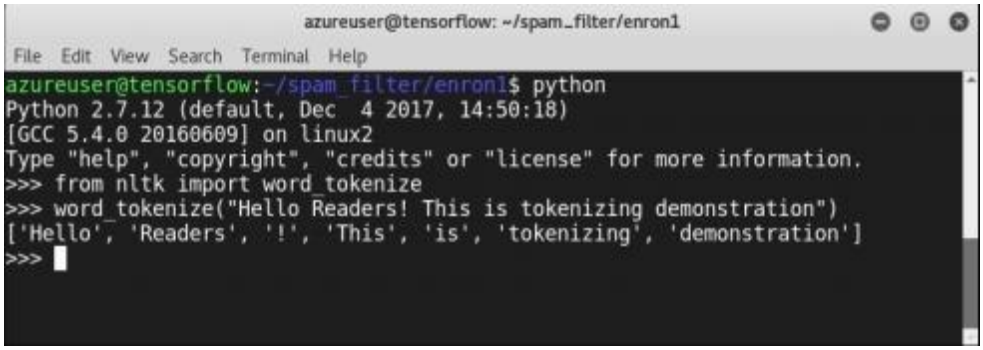
```
import os
import random
#initiate a list called emails_list
emails_list = []
Directory = '/home/azureuser/spam_filter/enron1/emails/'
Dir_list = os.listdir(Directory)
for file in Dir_list:
        f = open(Directory + file, 'r')
        emails_list.append(f.read())
f.close()
```

Just change the directory variable, and it will shuffle the files:



After preparing the dataset, you should be aware that, as we learned previously, we need to tokenize the emails:

```
>> from nltk import word_tokenize
```

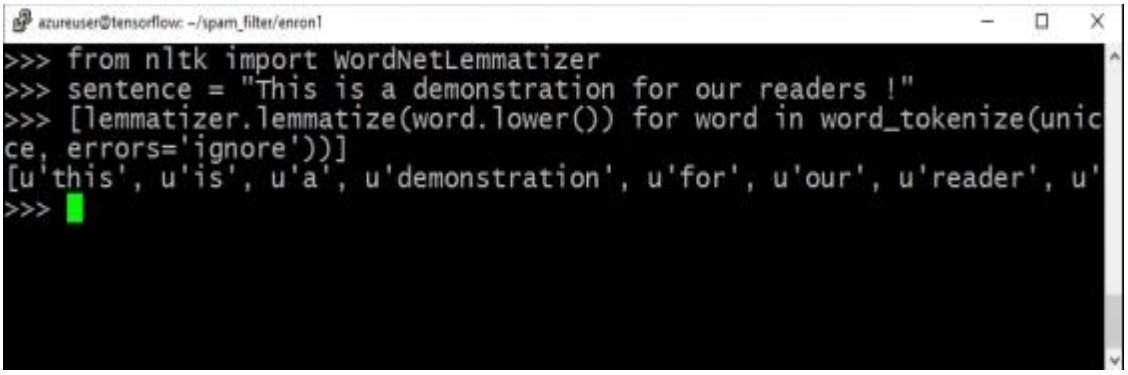


Also, we need to perform another step, called lemmatizing. Lemmatizing connects words that have different forms, like hacker/hackers and is/are. We need to import WordNetLemmatizer:

```
>>> from nltk import WordNetLemmatizer
```

Create a sentence for the demonstration, and print out the result of the lemmatizer:

```
>>> [lemmatizer.lemmatize(word.lower()) for word in
word_tokenize(unicode(sentence, errors='ignore'))]
```



Then, we need to remove stopwords, such as of, is, the, and so on:

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
```

To process the email, a function called Process must be created, to lemmatize and tokenize our dataset:

```
def Process(data):
        lemmatizer = WordNetLemmatizer()
        return [lemmatizer.lemmatize(word.lower()) for word in
word_tokenize (unicode (sentence,    errors='ignore'))]
```

The second step is feature extraction, by reading the emails' words:

```
from collections import Counter
```

```
def Features_Extraction(text, setting):
    if setting=='bow':
# Bow means  bag-of-words
        return {word: count for word, count in Counter(Process(text)).items() if
        not word in stop}
    else:
        return {word: True for word in Process(text) if not word in stop}
```

Extract the features:

```
features = [(Features_Extraction(email, 'bow'), label) for (email, label) in emails]
```

Now, let's define training the model Python function:

```
def training_Model (Features, samples):

    Size = int(len(Features) * samples)

    training , testing = Features[:Size], Features[Size:]

    print ('Training = ' + str(len(training)) + ' emails')

    print ('Testing = ' + str(len(testing)) + ' emails')
```

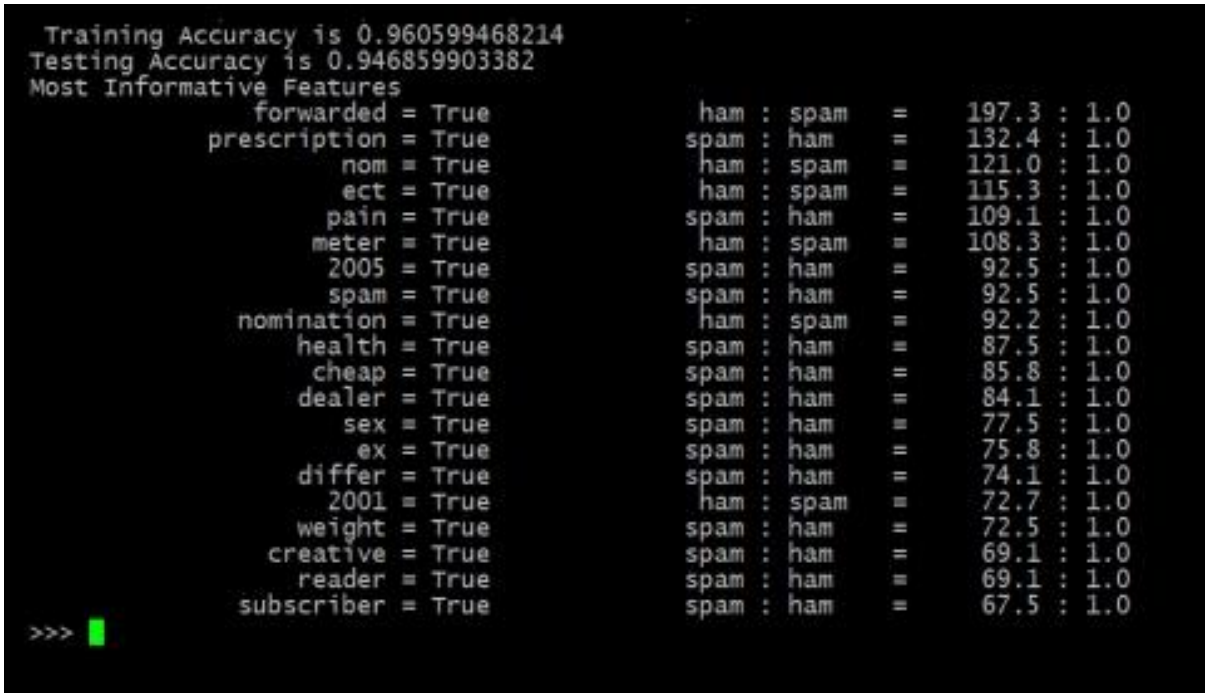
As a classification algorithm, we are going to use NaiveBayesClassifier:

```
from nltk import NaiveBayesClassifier, classify

classifier = NaiveBayesClassifier.train(training)
```

Finally, we define the evaluation Python function:

```
def evaluate(training, tesing, classifier):
    print ('Training Accuracy is ' + str(classify.accuracy(classifier, train_set)))
    print ('Testing Accuracy i ' + str(classify.accuracy(classifier, test_set)))
```



Summary

In this chapter, we learned to detect phishing attempts by building three different projects from scratch. First, we discovered how to develop a phishing detector using two different machine learning techniques, thanks to cutting-edge Python machine learning libraries. The third project was a spam filter, based on NLP and Naive Bayes classification. In the next chapter, we will build various projects to detect malware, using different techniques and Python machine learning libraries.

CHAPTER 4:

ML BASED PHISHING WEBSITE FEATURE CLASSIFICATION

4.1 INTRODUCTION

Phishing, one form of cyber-attacks, continues to be a growing concern not only to cyber security specialists but alsoto e-business users and owners. The severity of such cyber attack vector is continuously growing with the exponential increase in digital information generation and the increased reliance of people and business on cyber space. The Anti- Phishing Working Group (APWG) has seen rapid growth in the number of unique phishing websites detected from 2014 to 2016 [19]. The average annual growth rate is 97.36% andis expected to continue to grow. Estimates of annual direct financial loss to the US economy caused by phishing activitiesrange from $61 million to $3 billion [49].

To mitigate the increasing damage caused by phishing, a broad range of anti-phishing mechanisms have been proposed over the past two decades. These anti-phishing techniques can be categorized into three broad groups [12]: (1) Detective

Zuochao Dou is with the Electrical and Computer Engineering Department, New Jersey Institute of Technology, Newark, USA, E-mail: zd36@njit.edu.

Issa Khalil is with the Qatar Computing Research Institute (QCRI), Hamad bin Khalifa University (HBKU), Doha, Qatar. E-mail: ikhalil@hbku.edu.qa.

Abdallah Khreishah is with the Electrical and Computer Engineering Department, New Jersey Institute of Technology, Newark, USA, E-mail: abdallah@njit.edu.

Ala Al-Fuqaha is with the NEST Research Lab, College of Engineering & Applied Sciences Computer Science Department, Western Michigan Univer- sity, Kalamazoo, USA, E-mail: ala.al-fuqaha@wmich.edu.

Mohsen Guizani is with the Electrical and Computer Engineering Depart- ment, University of Idaho, Moscow, USA, E-mail: mguizani@uidaho.edu.

solutions (e.g., website filtering); (2) Preventive solutions(e.g., strong authentication [54], [85], [34], [32], [33], [43],
[53]); and (3) Corrective solutions (e.g., Site takedown [58],[57]). In this paper, we focus on detective solutions. Morespecifically, we look at software-based phishing detection schemes that are specialized in identifying and classifyingphishing websites. This class of approaches is arguably moreimportant than other approaches because it helps in reducing human errors. Preventative and corrective solutions take adifferent approach, but if the user behind the keyboard hasbeen successfully tricked by a phishing attempt, and willingly submitted sensitive information, then no firewall, encryption software, certificates, or authentication mechanism can help in preventing the attack from materializing [49]. Software-basedphishing detection also delivers improved results comparedto detection by user education (e.g., [60], [61], [98]) becausephishing attacks normally aim at exploiting human weaknesses[59]. For example, a study of phishing detection using usereducation [97] shows a 29% false negative rate (FNR) for thebest performance, while the software based approaches that aresurveyed by the same study have FNR in the range of 0.1%to 10%. For this reason, we focus our study on software basedphishing detection systems, and the term "phishing detection"will refer only to this form of detection in the rest of the paper. Although the research area of phishing detection and classi- fication is relatively rich, there is a lack of systematic analysisof the requirements, the capabilities, and the shortcomings of the existing anti-phishing techniques. For example, websitesthat offer identification and

classification of phishing as a service have been popular in recent years, however, those services leverage different evaluation datasets from various sources at different time periods to validate their outcomes. Albeit those schemes may have similar performance results (e.g., in terms of false positive rate, true positive rate, etc.), it is difficult to compare their performance because of the variation in the evaluation datasets employed. Consequently, a systematic assessment of the datasets used to validate phishing detection approaches is desired, as well as necessary, in order to provide a foundation for comprehensive comparisons among different phishing detection schemes, and ultimately, select the best in practice.

In this work, we complement the existing survey papers on phishing detection, including [49], [59], and [103], by providing a broad systematic analysis of software based anti-phishing approaches. In [103], the authors focus on studying, analyzing, and classifying the most significant and novel detection techniques, and pointed out the advantages and disadvantages of each approach. On the other hand, we present a more comprehensive systematic review of phishing detection schemes, not only from the perspective of detection algo-rithms, but also from a broader perspective that covers other important aspects including the phishing detection life cycle, taxonomy of phishing detection schemes, evaluation datasets, detection features, and evaluation metrics and strategies. The work in [49] focuses more on the attack side of phishing. More specifically, it presents details about phishing attacks including the anatomy of such attacks, why people fall in phishing attacks and how bad phishing is. However, it only provides a high level analysis of the state-of-the-art phishing countermeasures. In order to provide a systematic review of the phishing detection research, we first present the necessary information about the phishing attacks by answering three questions: (1) What is phishing?, (2) How does phishing work? and (3) What is the current status of phishing? Then, we conduct systematic review of phishing detection schemes in a detailed and comprehensive manner. Finally, Khonji et al. [59] present a literature survey about anti-phishing solutions (e.g., user training, email filtering and website detection, etc.), including their classification, detection techniques and evalu- ation metrics. Compared to [59], we focus on the software based phishing website detection schemes, which are proved to be the most effective anti-phishing solutions and are not systematically studied in [59].

In a nutshell, the objective of this paper is to provide a systematic understanding of existing phishing detection studies and provide a comprehensive way to evaluate phishing detection approaches from different perspectives in order to guide future developments and validations of new or upgraded anti-phishing techniques.

We summarize our contributions in this work as follows:

- Compile a comprehensive profile of phishing through its various definitions, detailed ecosystem (i.e., in terms of phishing life cycle, actors involved and their operations, etc.), and the state-of-the-art phishing trends.
- Present a systematic review of the software based phish- ing detection schemes from different perspectives includ- ing the life cycle, taxonomy, evaluation datasets, detection features, detection techniques and evaluation metrics.
- Introduce a novel feature, Network Round Trip Time (*NRTT*), for efficient and real time detection of phishing attacks.
- Provide detailed takeaway lessons for researchers and practitioners in the area of phishing

detection that we believe will help guide the development of effective phishing detection schemes.

TABLE I: MOST POPULAR DEFINITIONS OF PHISHING

	Definition
PhishTank [81]	Phishing is a fraudulent attempt, usually made through email, to steal your personal information.
APWG [44]	Phishing is a criminal mechanism employing both social engineering and technical subterfuge to steal consumers' personal identity data & financial account credentials.
Xiang et al. [118]	Phishing is a form of identity theft, in which criminals build replicas of target Web sites and lure unsuspecting victims to disclose their sensitive information like passwords, personal identification numbers (PINs), etc..
Whittaker et al. [108]	A phishing page is any web page that, without permission, alleges to act on behalf of a third party with the intention of confusing viewers into performing an action with which the viewer would only trust a true agent of the third party.
Khonji et al. [59]	Phishing is a type of computer attack that communicates socially engineered messages to humans via electronic communication channels in order to persuade them to perform certain actions for the attacker's benefit.
Ramesh et al. [90]	Phishing is a fraudulent act to acquire sensitive information from unsuspecting users by masking as a trustworthy entity in an electronic commerce.

TABLE II: TARGETS AND STRATEGIES OF PHISHING

	Target	**Strategy**
PhishTank [81]	Personal information	Social engineering
APWG [44]	Identity data, Financial account credentials	Social engineering
Xiang et al. [118]	Sensitive information	Not specified
Whittaker et al. [108]	Not specified	Not specified
Khonji et al. [59]	Not specified	Social engineering
Rameshe et al. [90]	Sensitive information	Not specified

4.1.1 Motivation behind doing this project

Motivation:

The motivation behind the project on phishing website features classification using machine learning is to develop a robust and automated system that can accurately identify and classify phishing websites. Traditional rule-based approaches for phishing detection often struggle to keep up with the ever-changing techniques employed by attackers. Machine learning algorithms, on the other hand, have the potential to learn from patterns and features in data, enabling the detection of previously unseen phishing websites.

By leveraging machine learning techniques, we aim to create a system that can analyze various features of websites and accurately classify them as either legitimate or phishing. This classification can be used by individuals, organizations, and security systems to prevent users from falling victim to phishing attacks and to enhance overall cybersecurity.

The benefits of using machine learning for phishing website classification include:

1. Automation: Machine learning algorithms can automatically analyze large amounts of website data, saving time and effort compared to manual inspections.

2. Adaptability: Machine learning models can adapt to new types of phishing attacks by learning from new patterns and features in the data.

3. Scalability: Machine learning approaches can handle a large number of websites, making them suitable for real-time detection in web browsers or network traffic.

4. Accuracy: By leveraging the power of machine learning, we can aim for high accuracy in classifying phishing websites, minimizing false positives and false negatives.

Overall, the project aims to contribute to the field of cybersecurity by developing a reliable and efficient system for detecting and classifying phishing websites using machine learning techniques.

4.1.2 Objective of the project

The objective of the project on phishing website features classification using machine learning is to develop an accurate and reliable system that can effectively detect and classify phishing websites. The main goals of the project include:

1. Phishing Website Detection: Develop a machine learning model capable of accurately identifying and distinguishing between legitimate websites and phishing websites. The model

should be able to analyze various features and patterns in website data to make informed predictions.

2. Feature Extraction: Identify and extract relevant features from website data that can provide valuable insights for distinguishing between legitimate and phishing websites. These features may include URL structure, domain information, SSL certificates, content analysis, and other indicators of phishing activity.

3. Machine Learning Algorithm Selection: Explore and evaluate different machine learning algorithms to identify the most suitable ones for the task of phishing website classification. This may involve comparing and testing algorithms such as decision trees, random forests, support vector machines, neural networks, or ensemble methods.

4. Model Training and Evaluation: Train the selected machine learning model using a labeled dataset consisting of both legitimate and phishing websites. Employ appropriate evaluation metrics to assess the model's performance, such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC).

5. Performance Optimization: Investigate techniques to optimize the model's performance, such as hyperparameter tuning, feature selection, or ensemble methods. The goal is to achieve the highest possible accuracy and reliability in classifying phishing websites.

6. Comparative Analysis: Conduct a comparative analysis of the proposed machine learning approach with existing phishing detection methods, including rule-based systems or other machine learning approaches. Assess the advantages, limitations, and overall effectiveness of the proposed system in terms of accuracy, efficiency, and scalability.

7. Practical Application: Demonstrate the practical application of the developed system by integrating it into real-world scenarios, such as web browsers, email clients, or network security systems. Evaluate its performance in real-time environments and assess its usability and effectiveness in detecting and preventing phishing attacks.

The ultimate objective of the project is to contribute to the field of cybersecurity by providing a robust and automated system that can accurately classify phishing websites, thereby enhancing the protection of individuals, organizations, and online systems against phishing threats.

4.1.3 Background of Phishing

Phishing is a prevalent form of cybercrime that involves tricking individuals into revealing sensitive information such as passwords, credit card numbers, or social security numbers. Phishing attacks typically occur through fraudulent websites that mimic legitimate ones, making it difficult for users to distinguish between the two. As the sophistication of phishing attacks continues to evolve, there is a growing need for effective methods to detect and classify phishing websites.

A. State-of-the-art Phishing Attacks

In this section, we first present the various definitions of phishing, then we introduce some statistics about phishingbetween January 2010 and June 2016. Finally, we describethe phishing ecosystem.

1) What is Phishing?*:* There is no consensus on how phishing should be defined. Different phishing definitions lead to different research directions and approaches (e.g., email filtering or website detection). It is important to clearly identify the target of any phishing detection approach to avoid confusion about its applicability in different scenarios. The target and scope of phishing detection approaches canbe analyzed from the definition of phishing which has been adopted by such approaches. Therefore, presenting a back- ground on the different definitions of phishing can help the readers understand the scope and the capabilities of different approaches. Table I summarizes the popular definitions ofphishing. On one hand, the definitions of PhishTank [81], APWG [19], Xiang et al. [118], Tameshe et al. [90] cover the majority of cases in which phishers aim at stealing sensitive personal information such as authentication credentials. Table II shows the comparison of those phishing definitions basedon phishing target and phishing strategy. The most dominant phishing strategies are social engineering (e.g., through fraudu-lent emails) and technical subterfuge (e.g., malware infection).However, sophisticated techniques (e.g., pharming [52]) are also used to harvest users' personal information from the Internet. On the other hand, the definitions of Whittaker etal. [108] and Khonji et al. [59] do not limit the attacker'starget (e.g., sensitive personal information). They describe the phishing strategy (e.g., phishing website or socially engineeredmessages) without stating a specific phishing target (e.g., only state the attackers' benefit). To sum up, the definitionof Whittaker et al.[108] is the most general among those reviewed, while APWG [19] defines the most commonly usedphishing attacks in a specific manner.

2) How Does Phishing Work?*:* In this section, we intro- duce the ecosystem of phishing in terms of phishing process, actors involved, their actions and interactions.

(i) Phishing Process: In a generic/traditional phishing scenario (i.e., mass-email phishing campaigns), an attackerhosts a fake website, and presents users of a web servicewith convincing emails containing a link to the fake website. When a user of the web service opens the link and entersher sensitive data, data is collected by the server hosting the fake website. As shown in Figure 1, Mihai and Giurea [78] suggest that a generic phishing process can be identified in five steps: (1) Reconnaissance: Phishers look for famous web service brands with a broad customer base; (2) Weaponization:Phishers design the phishing websites and social engineer on email spam; (3) Distribution: Phishers deliver emails tothe victims; (4) Exploitation: Phishers exploit weaknesses of humans to lure the victims into phishing traps via socially engineered emails. (5) Exfiltration: Phishers collect sensitive data from the phishing databases.

Unlike generic phishing attacks, spear phishing targets par- ticular individuals or organizations. [24] [104] [86]. Spear phishing attacks typically extract sensitive data from their victims by attaching a type of malware to emails or in the phishing website. Industry statistics indicate that spear phishing attacks have a success rate of 19%, while the success rate of generic phishing attacks is less than 5% [86].

For the purpose of this paper, we will not consider email fil-tering (e.g., [120], [21], [68]) as a phishing detection method. Our focus is on detection of website phishing for both generic and spear phishing attacks.

(ii) ***Phishing Actors:*** There are six actors involved in a typical phishing life cycle (see Figure 2), as defined in the following paragraphs:

- **Phisher**: Individuals or organizations that conduct phish- ing attacks in order to obtain a certain type of benefit, such as financial gain, identity hiding (e.g., refers to the situation in which phishers do not use the stolen identities directly, but rather sell them to interested criminals and cyber attackers.), fame and notoriety, etc. [59][105].
  Web service provider: Companies that provide a certain type of service (e.g., email, social network, e-banking, on-line shopping, etc.) on the Internet (usually through a website).
- **Web service subscriber**: Customers who subscribe to web services provided by the web service provider. Sub- scribers are the potential targets of traditional phishing attacks.
- **Web hosting provider**: Companies that provide website hosting services to web service companies.
- **Anti-phishing institutes**: Institutes that support those tackling the phishing menace and provide advice on anti-phishing controls and information on current trends [4].
- **Spear phishing targets**: Specific individuals or compa- nies targeted by phishers.

Each actor involved in the phishing process has different actions and reactions (summarized in Table III). Phishers try to use sophisticated techniques to evade phishing detection approaches (e.g., DNS poisoning [11]). In addition, there is a growing trend in which phishers have decoupled the process of phishing website hosting from the process of sending phishing emails in order to evade the anti-phishing solutions (Han, Kheir, & Balzarotti [45]).

Web service providers usually announce blacklists of phish-ing websites and recommend users to use strong authentication schemes (e.g., [17], [55], [34], [32], [33]). Additionally, web service subscribers highly depend on browser filters (e.g., Google Safe Browser [50]) and other third party anti-phishing toolbars (e.g., Netcraft [3]) to detect and block phishing attempts.

The role of web hosting providers is rather ambiguous in the phishing process. Reputable providers usually enforce strict "Terms of Use" and avail certain anti-phishing solutions (e.g., brand monitoring [12]). Due to financial constraints, many free-to-use web hosting providers may not be able to afford deploying good anti-phishing security measures, which leaves their customers not only vulnerable, but even worse, attractive targets for phishing.

TABLE III: OPERATIONS OF DIFFERENT PLAYERS INVOLVED IN PHISHING

	Basic Operations	Advanced Operations
A	1. Data collection 2. Website development 3.Email engineering	1. Evasion of anti-phishing techniques
B	1.Blacklist announcement	1. User behavior detection 2. Strong authentications
C	1. Human detection 2. Browser filter	1.Phishing detection toolbar
D	1.Policy enforcement	1.Brand monitoring
E	1. Phishing data analysis 2. Anti-Phishing solutions	1. Law enforcement 2. Government coalition
F	1.Employee training	1. Email filtering 2. Phishing detection software

A: Phisher; B: Web service provider; C: Web service subscriber; D: Web hosting provider; E: Anti-Phishing institute; F: Spear Phishing targets

Anti-phishing institutes collect and analyze phishing data (e.g., suspicious websites reported by users) from various sources (e.g., users' reports via anti-phishing toolbars), and provide anti-phishing suggestions and solutions (e.g., up-to- date phishing website blacklist, phishing detection toolbars, etc.). In addition, they may also cooperate with government agencies such as public security and law enforcement to detect and prevent cyber attacks [4].

3) ***What is the Current State of Phishing?****:* According to phishing activity trends reports published by APWG [19] from Jan. 2010 to Jun. 2016 (shown in Figure 3), the number of unique phishing websites established per month increased significantly since 2015 (i.e., the average number for 2016is 2.93 times the average from prior years). It is clear that phishers profited from this type of cyber-attacks, which result in financial loss for both web subscribers and business owners.Therefore, agile techniques to mitigate phishing will continue to be a pressing need.

Phishing attacks tend to empoly advanced techniques tolure web service users into their rogue websites. Using the database from Trend Micro web reputation technology, Pajares [83] reports the number of phishing sites that use HTTPSconnections increased significantly in 2014 compared to 2010 (shown in Figure 4). Attackers become more cautious and attentive when designing phishing websites to evade existing phishing detection methods [1]. Some phishing groups arecapable and desire to perform more advanced phishing attacks. Avalanche (commonly known as the Avalanche Gang) is a criminal syndicate involved in phishing attacks [109]. In 2010, APWG reported that Avalanche was responsible for two-thirds of all phishing attacks in the second half of 2009, describingit as "one of the most sophisticated and damaging on theInternet" and "the world's most prolific phishing gang" [10]. Ithas been discovered that Avalanche uses different techniques to evade the anti-phishing mechanisms.

In addition, more and more sophisticated techniques arebeing used to implement phishing attacks. For example, the pharming attack, a refined version of phishing attacks, is designed to steal users' credentials by redirecting them tofraudulent websites using DNS-based techniques [41], [58]. Many computer security experts predict that the use of pharm-ing attacks will continue to grow as more criminals embrace these techniques [52].

Fig. 1: Illustration of the phishing process

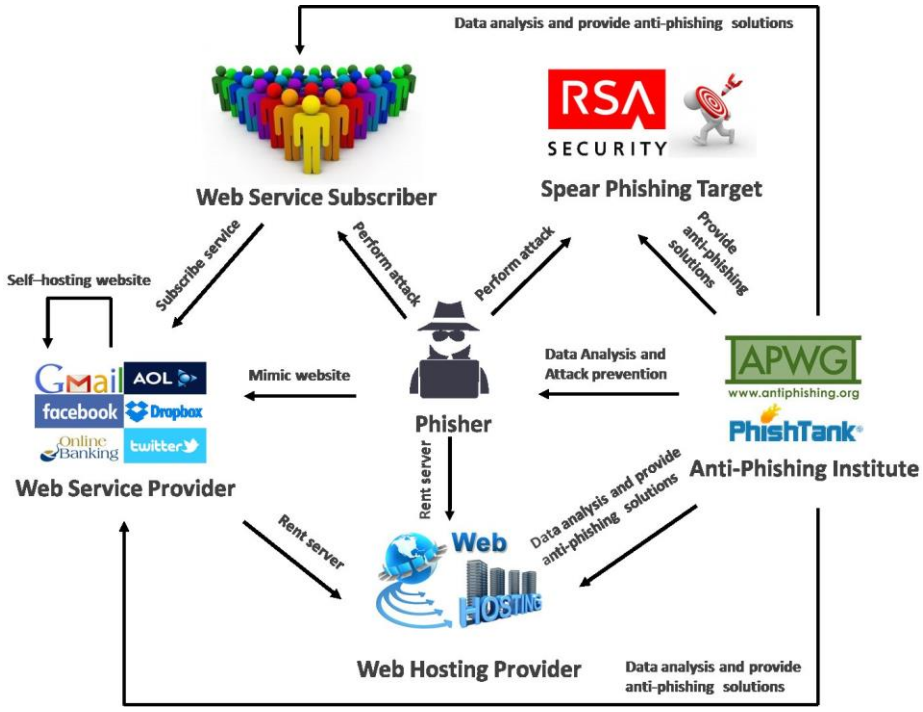


Fig. 2: Players involved in the phishing process.

A. Life cycle of phishing detection

As mentioned in Section I, we do not incorporate phishing detection approaches that rely on user education due to their poor performance. In addition, we do not cover phishing detection methods that perform email filtering because it isa different detection theme that warrants a separate compre- hensive study on its own. we reemphasize here that our focus is on the area of software-based phishing detection which aimsat detecting or blocking phishing websites.

The life-cycle of software-based phishing detection is il- lustrated in Figure 5. Starting from the initial inputs, the detection scheme extracts phishing detection features (or calledheuristics, as detailed in Section IV-B) and/or blacklists from various sources (e.g., URL related information, trusted third party, WHOIS server, etc.) via different feature mining ap- proaches (e.g., search engines, target identification algorithms, etc.). Then, it applies different data mining algorithms and/or proposes various detection strategies to the engineered featuresto achieve its objectives (e.g., identifying phishing links, blocking phishing websites, etc.). To evaluate the performanceof phishing detection schemes, various evaluation datasets are collected from different sources (e.g., PhishTank, Yahoo directory, etc.). Finally, leveraging the collected datasets and following various validation strategies (e.g., cross validation), the proposed scheme is evaluated based on multiple metrics (e.g., False Positive Rate, False Negative Rate, etc.).

In the coming sections, following the life cycle of software-based phishing detection schemes, we present a comprehen- sive study of the phishing detection research from 5 differ-ent perspectives, namely, classification of phishing detection techniques, validation datasets, detection features, detection techniques and detection criteria.

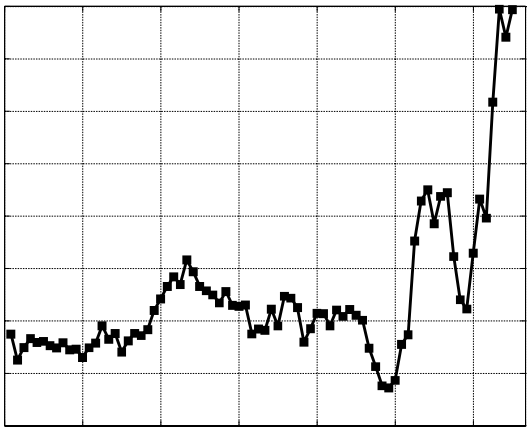


Fig. 3: The number of unique phishing sites per month from Jan. 2010 to Jun. 2016

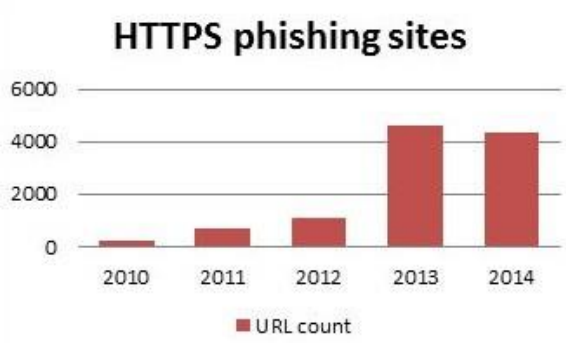


Fig. 4: The number of phishing sites that use HTTPS. Re-printed from [83]

4.2 PHISHING DETECTION SCHEMES: TAXONOMY

In phishing literature, software-based phishing detection schemes are usually categorized into heuristic and blacklist based schemes [49] [59]. Heuristic-based approaches examine contents of the web pages including: (1) surface level content (e.g., the URL); (2) textual content (e.g., terms or words that appear on a given web page); (3) visual content (e.g., the layout, and the block regions etc.) [122]. These methods can detect phishing attacks as soon as they are launched but also introduce relatively high false positive rates (*FPR*). Blacklist-based approaches have a higher level of accuracy. However, they do not defend against zero-hour attacks [49] [115]. Com- binations of heuristic and blacklist based approaches provide more robust and flexible defense against phishing attacks than either one on a standalone basis.

In this paper, we classify phishing detection approaches as either public phishing detection toolbars or academic phishing detection/classification schemes. Phishing detection toolbars use blacklists and/or selected heuristics to identify phishing websites. There is usually little information about what heuris-tics these toolbars use and how they are used. Academic phishing detection solutions are similar to phishing detec-tion toolbars, but usually apply more complex technologies and are usually not available/feasible for public use. Most academic phishing classification schemes apply combinations of heuristics features into various data mining algorithms to enhance the classification accuracy. Table IV summarizes the differences between phishing detection toolbars and academic phishing detection/classification schemes. Note, the "scheme details" column in Table IV estimates the amount of publicly available details about detection schemes, such as detection methodology, data mining algorithms, and datasets.

Furthermore, based on the heuristic/blacklist classification, we further classify the academic phishing detection approaches into more specific and fine-grained sub-categories, namely, (1) heuristic: URL based methods; (2) heuristic: page content based methods; (3) heuristic: visual similarity based methods; heuristic: other methods; (5) blacklist based methods; (6) hybrid methods. Details about each category are introduced in Section IV-B.

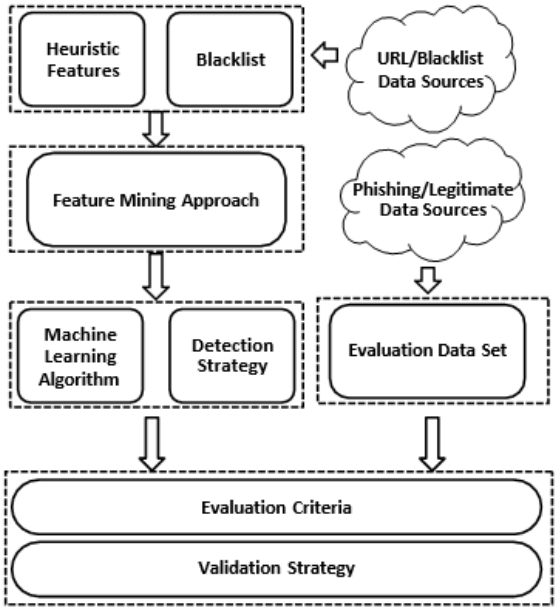


Fig. 5: Life Cycle of typical phishing detection schemes

A. *Public phishing detection toolbars*

Many freely available anti-phishing toolbars offer detection and blocking services against Internet phishing attacks. These toolbars typically come in the form of web browser extensions (i.e., default extensions or third party extensions) that warn users about a suspicious phishing site after clicking on its URL.

Publicly available anti-phishing toolbars are either embed- ded in the browser as default extensions (e.g., Microsoft SmartScreen Filter [112]) or can be downloaded from third party websites (e.g., Netcraft [3]). They both display security warnings on screen when certain actions are triggered in the browser. These security warnings can be classified into two types [59]:

Passive warnings: Passive warnings display various in- formation (e.g., user ratings, site suggestions, etc.) about the website that is currently being visited but do not block the content of the website, as depicted in Figure 6.

- **Active warnings**: Active warnings display warning in-formation about the website a user is trying to visit and block the content of the website, as depicted in Figure 7. Many studies have shown that the majority of web ser- vice users ignore security warnings provided by anti-phishing toolbars [31] [35] [116]. Furthermore, Egelman et. al. [36] found that active warnings are much more effective than passive warnings (79% of participants paid attention to active warnings while only 13% participants paid attention to passive warnings). Table V summarizes the information gathered about the state-of-the-art anti-phishing toolbars. In the following paragraphs, we discuss the details of those toolbars:

Google Safe Browsering: It uses a browser to check URLs against Google's constantly updated blacklist of unsafe web resources (e.g., phishing websites) [50] and provides active warnings to the end users. According to Google Safe Browsing's website, for different platform and threat types, it examines pages against the safe browsing lists. It also issues reminders before users access risky links.

McAfee SiteAdvisor: This is a web application that reports on the identity of websites by scanning them for potential mal- ware and spam [111]. The detection result is decided according to a combination of heuristics and manual verification, such as the age and country of the domain registration, the number of links to other known-good sites, third-party cookies, and user reviews [30]. In addition, it provides passive warnings.

Netcraft Anti-Phishing Toolbar:Provides Internet security services including anti-fraud and anti-phishing services, ap- plication testing and PCI scanning [113]. According to its website, Netcraft's toolbar screens and identifies the deceiving contents in URLs. It also ensures that the navigational controls (e.g., toolbar and address bar) are activated in order to prevent pop-up windows (particularly for Firefox). In addition, it shows the geographic information of the hosting location of the sites and analyzes fraudulent URLs (e.g., the real citibank.com or barclays.co.uk sites have little possibility to be located in the former Soviet Union [3]).

SpoofGuard: A heuristics-based anti-phishing toolbar de- veloped for Internet Explorer with passive warnings. The heuristics used include (1) Domain name check: examines if the domain name for the attempted URL matches recent entries; (2) URL Check: checks if the username, the port number, as well as the domain name, are suspicious; (3) Email Check: determines whether the current URL directs to the browser via email; (4) Password

Field Check: determines if theinput fields of type "password" are located in the document; (4) Link Check: searches for risky links in the body of the document; (6) Image Check: analyzes the images of the new site vs. the previous sites; (7) Password Tracking: prevents theuser from typing the same username and password for multiplesites [63].

TABLE IV: SUMMARY OF DIFFERENCES BETWEEN PHISHING DETECTION TOOLBARS AND ACADEMIC PHISHING DETECTION/CLASSIFICATION SCHEMES

	Public usage availability	Data Mining Algorithms	Exact Heuristics	Blacklist	Scheme Details	Goal
Publicly available Phishing detection toolbars	Yes	Very little	Very little	Yes	Little	**Detect and block phishing websites**
Academic phishing detection/classification schemes	Very little	Yes	Yes	Little	Yes	**Detect or classify phishing websites**

TABLE V: INFORMATION ABOUT SELECTED STATE-OF-THE-ART ANTI-PHISHING TOOLBARS

	Technology	Warning Type	Platform
Google Safe Browsing	BlacklistHeuristics(Suspected)	Active	Internet Explorer, Firefox and Chrome
McAfee SiteAdvisor	Heuristics, Blacklist(Manual verification)	Passive	Internet Explorer, Firefox and Chrome
Netcraft Anti-Phishing Toolbar	BlacklistHeuristics	Active	Internet Explorer, Firefox and Chrome
SpoofGuard	Heuristics	Passive	Internet Explorer
Microsoft SmartScreen Filter	Blacklist	Active	Internet Explorer
EarthLink Toolbar	Blacklist(User ratings and manual verification), Heuristics (Unknown)	Passive	Internet Explorer and Firefox
eBay Toolbar	BlacklistHeuristics	Passive	Internet Explorer
GeoTrust TrustWatch Toolbar	Blacklist Third-party reputation services and certificate authorities	Passive	Internet Explorer
WOT (Web of Trust)	Blacklist(User ratings, Third-party information)	Active	Internet Explorer, Firefox and Chrome

Microsoft SmartScreen Filter: A blacklist-based phishing and malware filter implemented in several Microsoft browsers,including Internet Explorer and Microsoft Edge [112]. When browsing the site, SmartScreen helps monitor and identify the possibility of visiting a suspicious page. If so, it issues an active warning before next step is taken, as well as soliciting feedback from users. SmartScreen also maintains a list ofreported phishing and software sites. It screens the list to check if a match is found. In that case, it issues a warning message while blocking the site for user's safety. In addition, security checks are also performed when the user starts a download from the site. Moreover, SmartScreen compares the download to a list of existing downloads by other users. A warning is issued if it's a brand new download.

EarthLink Toolbar: Helps to protect the user from on-line scams by displaying a security

rating (i.e., passive warning)for all the websites the user visited previously. Additionally,it alerts the user if he tries to access a previously known fraudulent website. It appears to rely on a combination of heuristics, user ratings, and manual verification [30].

eBay Toolbar: Helps the buyers and sellers with real time alerts and keeps users safe from spoofing and fraudulent attacks by detecting fake sites via a combination of heuristics and blacklists through passive warnings [30].

GeoTrust TrustWatch Toolbar: Provides website veri- fication service that alerts the users to potentially unsafe, or phishing web sites based on the information of severalthird-party reputation services and certificate authorities via passive warnings [42]. TrustWatch notifies the users that the website has passed the verification scan based on a list of disreputable sites. It would also recommend additional caution when inputting sensitive information to the website. Further- more, it blocks the initial attempt when visiting potentially unsafe websites and warns users in case of a risk in revealing information to the site.

Web of Trust (WOT): A browser extension that tells the user which websites he can trust via active warnings[42]. It ensues the user's Internet safety from scams, malware, rogue web stores and dangerous links based on community ratings and reviews.

B. Academic phishing detection/classification schemes

Unlike the public anti-phishing toolbars, which aim at pro- viding real-time warnings about the legitimacy of visited web-sites, academic phishing detection and classification schemes normally focus on improving the detection accuracy and reducing the number of false alerts by employing sophisticatedtechnologies and various machine learning algorithms. Table VI shows the time-based (from 2005 to 2016) developmentof 41 selected academic phishing detection/classification ap- proaches. In order to choose the most representative studies,in this paper, we comply with the following criteria based on state-of-the-art literature:

- **Pioneering**: Research that introduces new ideas or meth-ods to the literature.
- **Attention**: Research that receives more attentions interms of the number of citations.
- **Completeness**: Research that presents their work follow-ing the entire life cycle of phishing detection in depth.

Based on the proposed criteria, all of the 41 selected works are introduced in the following sections and twelverepresentative studies are chosen as examples to illustrate the detailed detection methodology in each category. They are listed in Table VI and introduced below.

Visual similarity based methods: Chen et. al. [27] describe a novel heuristic anti-phishing system that explicitly employs gestalt and decision theory concepts to model perceptual similarity. More specifically, they apply logistic regression algorithm to a set of normalized page content features. The proposed scheme can achieve 100% true positive rate and0.74% false positive rate.

Fig. 6: Passive warnings from Netcraft anti-phishing toolbar. Reprinted from: http://toolbar.netcraft.com/.

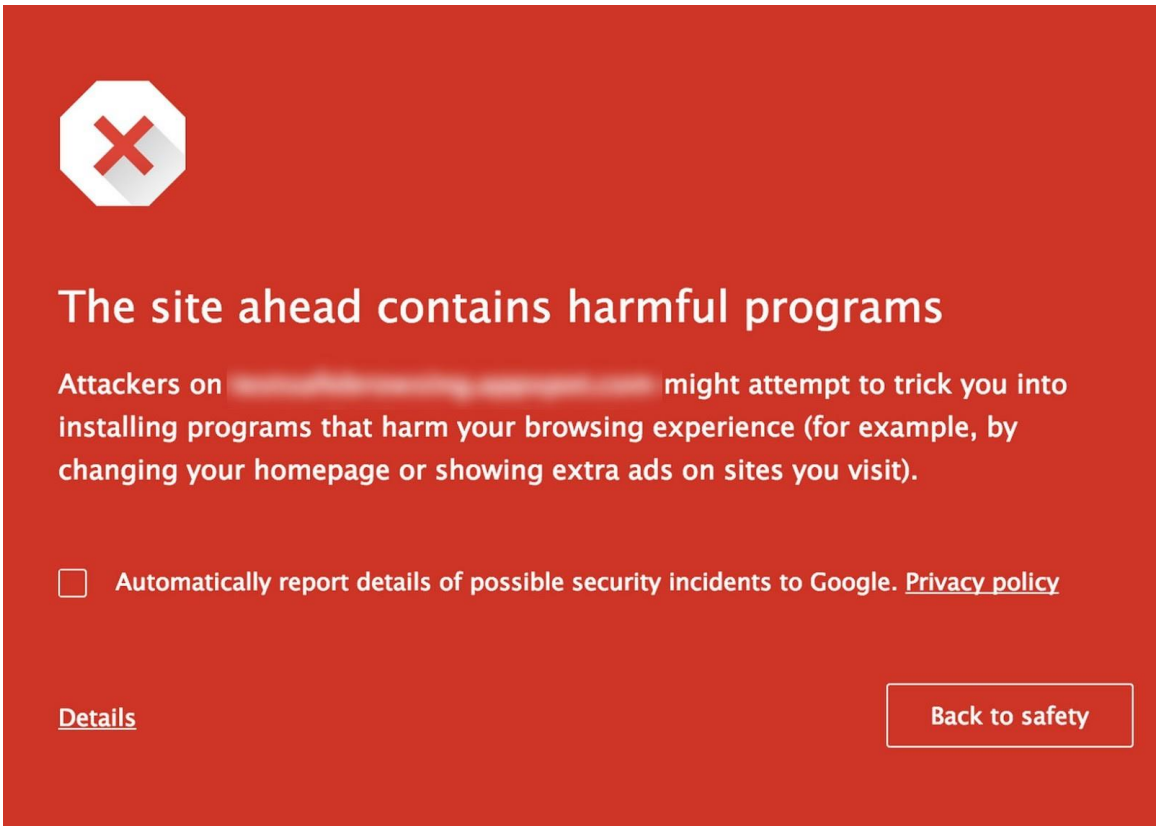


Fig. 7: Active warning from Google Safe Browsering. Reprinted from: https://googleblog.blogspot.com/2015/03/protecting-people-across-web-with.html.

TABLE VI: TIME-LINE BASED DEVELOPMENT OF PHISHING DETECTION SCHEMES FROM 2005 TO 2016

Research	Year	Represented(Y/N)
EMD based visual similarity for detection of phishing webpages [39]	2005	
Phishing web page detection [106]	2005	
Anomaly based web phishing page detection [84]	2006	
Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD) [38]	2006	Y
Cantina: a content-based approach to detecting phishing web sites [124]	2007	Y
A framework for detection and measurement of phishing attacks [40]	2007	Y
Anti-phishing based on automated individual white-list [23]	2008	
A phishing sites blacklist generator [96]	2008	
Visual similarity-based phishing detection [77]	2008	
B-apt: Bayesian anti-phishing toolbar [69]	2008	
Phishzoo: An automated web phishing detection approach based on profiling and fuzzy matching [15]	2009	
Fighting phishing with discriminative keypoint features [25]	2009	
Beyond blacklists: Learning to detect malicious web sites from suspicious URLs [72]	2009	Y
A hybrid phish detection approach by identity discovery and keywords retrieval [119]	2009	
Identifying suspicious URLs: An application of large-scale online learning [73]	2009	Y
Visual similarity-based phishing detection without victim site information [46]	2009	
Automatic detection of phishing target from phishing webpage [70]	2010	
Phishnet: predictive blacklisting to detect phishing attacks [119]	2010	Y
Large-scale automatic classification of phishing pages [108]	2010	Y
Lexical feature based phishing URL detection using online learning [22]	2010	
Detecting visually similar web pages: Application to phishing detection [26]	2010	
Intelligent phishing detection system for e-banking using fuzzy data mining [13]	2010	
Using domain top-page similarity feature in machine learning-based web phishing detection [94]	2010	
Textual and visual content based anti-phishing: A bayesian approach [122]	2011	
Cantina+: A feature-rich machine learning framework for detecting phishing web sites [118]	2011	Y
PhishDef: URL names say it all [66]	2011	
Design and evaluation of a real-time URL spam filtering service [102]	2011	Y
Antiphishing through phishing target discovery [107]	2012	
Using visual website similarity for phishing detection and reporting [76]	2012	
PhishAri: Automatic realtime phishing detection on twitter [16]	2012	
Phishing Website detection using latent Dirichlet allocation and AdaBoost [89]	2012	
Phishing detection plug-in toolbar using intelligent Fuzzy-classification mining techniques [14]	2013	
Phishstorm: Detecting phishing with streaming analytics [74]	2014	
An efficacious method for detecting phishing webpages through target domain identification [90]	2014	Y
An anti-phishing system employing diffused information [27]	2014	Y
Predicting phishing websites based on self-structuring neural network [80]	2014	
Examination of data, rule generation and detection of phishing URL using online logistic regression [37]	2014	
Feature extraction and classification phishing websites based on URL [20]	2015	
New rule-based phishing detection method [79]	2016	
Know Your Phish: Novel Techniques for Detecting Phishing Sites and their Targets [75]	2016	Y
PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder [101]	2016	

The most representative work in this category is done byFu et. al. [38]. They propose an effective phishing website detection approach via visual similarity assessment based on Earth Mover's Distance (EMD) [47].

The detection process contains two phases, namely, generating signature of web pages and computing visual similarity score from EMD.

The web page processing phase (i.e., generate the signature)contains three steps: (1) obtain the

image of a web page from its URL using Graphic Device Interface (GDI) API;
(2) perform image normalization (the normalized image size is 100 x 100, and Lanczos algorithm [93] is used to resize the image); (3) transform the web page image by a visual signature. The signature is comprised of the image color tuple
using the [Alpha, Red, Green, and Blue] (ARGB) scheme and the centroid of its position in the image.

The second step is to compute the EMD between the visual similarity signatures of the two web pages (legitimate site and phishing site). Firstly, the normalized Euclidean distance of the degraded ARGB colors and the centroids are computed. Then the two distances are added up with their corresponding weights (i.e., p and q, $p + q = 1$). The normalized feature distance between ϕ_i and ϕ_j is defined as:

$$d_{ij} = ND_{feature}(\varphi_i, \varphi_j) = p * ND_{color}(dc_i; dc_j)$$
$$+q * ND_{centroid}(C_{dc_i}; C_{dc_j})$$

where $\varphi_i =< dc_i, C_{dc_i} >$, $dc =< dA; dR; dG; dB >$ is the color tuple, and C_{dc} is the centroid value. Suppose we have signature $S_{s,a}$ and signature $S_{s,b}$, the EMD between $S_{s,a}$ and $S_{s,b}$ can be calculated as:

$$EMD\{S_{s,a}, S_{s,b}\} = \frac{\sum f_{ij} * d_{ij}}{\sum f_{ij}}$$

where f_{ij} is the flow matrix calculated through linear programming [47]. Note that if EMD=0, the two images are identical, if EMD=1, they are completely different.

Finally, the EMD-based visual similarity of two images is defined as:

$$VS\{S_{s,a}, S_{s,b}\} = 1 - [EMD\{S_{s,a}, S_{s,b}\}]^{\alpha}$$

where $\alpha \in (0, +\infty)$ is an amplification factor that limits the skewness of the visual similarity for the distributed in the (0,1) range.

Large-scale experiments with 10,281 suspected web pages are carried out and the proposed scheme achieves 0.71% false positive rate and 89% true positive rate.

Similar works based on visual similarly include [39] [106] [77] [70] [15] [25] [46] [26] [94] [122] [76].

Page content based methods: Zhang et. al. [124] propose CANTINA, a novel content-based approach for detecting phishing web sites based on the Term Frequency/Inverse Document Frequency (TF-IDF) information retrieval metric. In addition, using some heuristics, the false positive rate is reduced. Generally, CANTINA works as follows:

1) CANTINA calculates the TF-IDF scores of each termof the content in the given website.
2) CANTINA generates a lexical signature by taking the five terms with highest TF-IDF weights.
3) CANTINA sends the lexical signature to a search engine(i.e., in their case, Google Search).
4) If the domain name of the current website matches the domain name of the top *N* search results, it is consideredto be a legitimate website. Otherwise, it is concluded to be a phishing

site. Note that, the value of *N* affects the false positives.

CANTINA with TF-IDF alone results in a relatively high false positive rate. Therefore, several heuristics are used to reduce the false positive rate, including:

- Age of Domain: it examines the age of the domain name. If the page has been registered for more than 12 months, the heuristic returns +1 (i.e., legitimate), otherwise it returns -1 (phishing).
- Known Images: it examines whether a page contains inconsistent well-known logos.
- Suspicious URL: it examines if the URL contains an "@" or a "-" in the domain name.
- Suspicious Links: for each link in the webpage, it per- forms the above three URL checks.
- IP Address: it examines if the URL contains an IP address.
- Dots in URL: it examines the number of dots in the URL.
- Forms: it examines if a web page contains any HTML text entry form requesting sensitive personal data (e.g., password).

In addition, CANTINA uses a simple forward linear model to make the decision

$$S = f(w_i * h_i)$$

where h_i is the result of each heuristic, w_i is the weight of each heuristic, and f is a simple threshold function.

$$f(x) = 1 \quad if \quad x > 0, \quad f(x) = -1 \quad if \quad x <= 0.$$

Here, 1 means legitimate site and -1 means a phishing site.

The proposed scheme could achieve 97% true positive rate while maintaining 1% false positive rate.

In 2011, Xiang et. al. [118] extended the work of Zhang et. al. [124] by proposing CANTINA+ which is claimed to be the most comprehensive feature-based approach in the literature. It exploits the HTML Document Object Model (DOM), search engines and third party services with machine

learning techniques to detect phishing. It has been shown to achieve 0.4% false positive rate and over 92% true positive rate.

Similar works based on page content include [119] [89] [14].

URL based methods: Garera et. al. [40] claim that it is of- ten possible to tell whether or not a URL belongs to a phishing attack without requiring any knowledge of the corresponding page data. By applying several selected features (i.e., page rank, domain name white list and URL based features) into logistic regression learning algorithm, the proposed scheme is efficient and has a high accuracy.

The most representative work in this category is done by Ma et. al. [72]. They propose a phishing detection approach to automatically classify URLs based on different data mining algorithms across both lexical and host based URL features.

The lexical features selected in this method include the length of the hostname, the length of the entire URL, as well as the number of dots in the URL. In addition, the authors create a binary feature for each token in the hostname (delimited by ".") and in the path URL (strings delimited by "/", "?', ".", "=","-" and " "). The host-based features contain: (1) IP address properties (e.g., is the IP address in a blacklist?); (2) WHOIS properties (e.g., the date of registration, update, and expira- tion); (3) Domain name properties (e.g., the time-to-live (TTL) value for the DNS records associated with the hostname); (4) Geographic properties

(e.g., the continent/country/city that the IP address belongs to).

All the features of the URL are encoded into high dimen- sional feature vectors and then different types of classifiers areapplied to them. Here are some examples of the classifiers:

- *Naive Bayes*: Let x denote the feature vectors and y $\in$ 0, 1 denote the label of the website, with $y = 1$ for malicious and $y = 0$ for legitimate ones. $P(x\ y)$ denotes the conditional probability of the feature vector givenits label. Then, assuming that malicious and legitimate websites are equally probable, the posterior probability that the feature vector x belongs to a malicious URL is computed as:

$$P(y = 1|x) = \frac{P(x|y = 1)}{P(x|y = 1) + P(x|y = 0)}$$

  Finally, the right hand side of the equation is thresholded to predict the binary label of the feature vector x.

- *Support Vector Machine (SVM)*: The decision using SVMs is expressed in terms of a kernel function $K(x, x')$that computes the similarity between two feature vectors and non-negative coefficients α_i that indicate which train- ing examples lie close to the decision boundary. SVMs classify new examples by computing their distance to the decision boundary:

$$h(x) = \sum_{1}^{n} \alpha_i(2y_i - 1)K(x_i, x)$$

  where $h(x)$ is the threshold to predict a binary label for the feature vector x.

  Logistic Regression: LR classification is based on the distance from a hyperplane decision boundary [71]. The decision function is $\sigma(z) = [1 + e^{-z}]^{-1}$ that converts these distances into probabilities that feature vectors have positive or negative labels. The conditional probability that feature vector x has a label $y = 1$ is:

$$P(y = 1|x) = \sigma(wx + b)$$

  where w (i.e., the weight vector) and b (i.e., the scalar bias) are parameters computed based on the training data.Finally, the right hand side of the equation is thresholded to decide the label of the feature vector x.

The proposed scheme can achieve 0.1% false positive rate and 92.4% true positive rate.

Similar works based on URL related features include [22] [66] [74] [37] [20].

Blacklist based methods: PhishNet [87] is a predictive blacklisting scheme to detect phishing attacks. Traditional blacklist approaches (i.e., exact match with the blacklistedentries) are easy for attackers to evade. Instead, PhishNetuses five heuristics (i.e., top-level domains, IP address, direc- tory structure, query string, brand name) to compute simple combinations of blacklisted sites to discover new phishingsites. Also, it proposes an approximate matching algorithmto determine whether a given URL is a phishing site or not. PhishNet consists of two major components, namely, component I: predicting malicious URLs and component II: approximate matching.

The basic idea of component I is to combine different URL heuristics of known phishing URLs from a blacklist (i.e., PhishTank database) to generate new phishing URLs. These five URL heuristics include: (1) top-level domains (TLDs): by changing the TLDs of known blacklist entries, a list of new URLs can be obtained;

(2) IP address: the predicted new phishing sites are obtained by enumerating all the

combinations of the hostnames and pathnames of the known blacklisted websites with the same IP address; (3) directory structure: the idea is that two URLs sharing a common directory structure (e.g., www.abc.com/online/signin/paypal.htm and www.xyz.com/online/signin/ebay.htm) may have similar sets of file names. Therefore, the predicted new URLs are www.abc.com/online/signin/ebay.htm and www.xyz.com/online/signin/paypal.htm; (4) query string: starting from the observation that some URLs with the exact same directory structure differ only in query string (e.g., www.abc.com/online/signin/ebay?XYZ, and www.xyz.com/online/signin/paypal?ABC), two new URLs, www.abc.com/online/signin/ebay?ABC and www.xyz.com/online/signin/paypal?XYZ, are created; (5) brand name: the intuition here is that phishers often target multiple brand names using the same URL structure method. Therefore, the predicted URLs are obtained by changing the brand names embedded in the known phishing URLs.

After obtaining the whole set of the predicted URLs, Phish-Net first performs a DNS lookup to filter out sites that cannot be resolved. Then it conducts a content similarity check (i.e., using an online tool at http://www.webconfs.com) between the known phishing URLs and the corresponding predicted URLs.

The predicted URL is concluded to be a phishing site if the similarity score exceeds a certain threshold.

The second component performs an approximate match of a given URL to determine whether it is a phishing site or not. It first breaks the input URL into four different entities: IP address, hostname, directory structure and brand name. Then it assesses each entity by matching with the corresponding part of the known phishing URLs to generate an evaluation score. If the score is higher than a certain threshold, it is considered to be a phishing URL.

About 18,000 new phishing URLs are discovered from a set of 6,000 new blacklist entries. The proposed scheme achieves 3% false positive rate and 95% true positive rate.

Similar works based on blacklist/white-list include [23] [96].

Hybrid methods: Whittaker et. al. [108] use a logistic regression classifier to maintain Google's phishing blacklist automatically by examining the URL and the contents of a page. The proposed scheme correctly classifies more than 90% of phishing pages several weeks after training concludes. Marchal et. al. [75] develop a phishing detection system that requires very little training data, which is language- independent, resilient to adaptive attacks and implemented entirely on client-side. The proposed target identification algo-rithm is faster than previous works and can help reduce false positives. The proposed scheme achieves 0.5% false positive rate and 99% true positive rate.

The most representative work in this category is Monarch [102], a real-time system that determines whether the sub- mitted URL is spam or not. The authors deploy a real implementation to demonstrate its scalability, accuracy, and run time performance. Monarch consists of four components:

(1) URL aggregation: it accepts URL submissions from a number of major email providers and Twitter's streaming API;

(2) Feature collection: it visits a URL via Firefox web browser to collect page content; (3) Feature extraction: it transforms the raw data generated from the feature collection component into a feature vector (e.g., transforming URLs into binary features and converting HTML content into a bag of words [110]).

(4) Classification: feature vectors are applied to a proposed distributed logistic regression

classifier for classification. The selected features in [102] are represented by a combination of URL based features, page content based features, whitelist and other features (e.g., routing data), including:

- Initial URL and Landing URL: domain tokens, path tokens, query parameters, number of sub-domains, lengthof domain, length of path, length of URL.
- Redirects: number of redirects, type of redirect.
- Sources and Frames: URL features for each embedded IFrame links and sources links.
- HTML Content: tokens of main HTML, frame HTML,and script content.
- Page Links: URL features for each link, number of links,ratio of internal domains to external domains.
- JavaScript Events: number of user prompts, tokens ofprompts.
- Pop-up Windows: URL features for each window URL.

Plugins: URL features for each plugin URL.

- HTTP Headers: tokens of all field names and values;
- DNS: IP of each host, mailserver domains and IPs,nameserver domains and IPs.
- Geolocation: country code, city code of each IP.
- Routing Data: ASN/BGP prefix for each IP encountered.
- Whitelist: a whitelist of known good domains.

Logistic Regression (LR) with L1-regularization is chosen as the classifier. To predict the class label ($y = 1$ means non-spam, $y = +1$ means spam) of a URL's feature vector x. We train a linear classifier characterized by weight vector w. Given a set of n labeled training points $(x_i; y_i)$, $i = 1 : n$, the training process is to find w that minimizes the following objective function:

$$f(w) = \sum_{1}^{n} log(1 + exp[-y_i(x_i * w_i)]) + \lambda * ||w||_1$$

The first component is the log likelihood of the training data asa function of the weight vector. The second component is the regularization which adds a penalty to the objective function [71].

To perform the learning process over large-scale datasets inreal time, the data is divided into m shares and then processed in a distributed manner (i.e., using Hadoop Spark [121]).

Monarch can achieve an overall accuracy of 91% with 0.87% false positives with a throughput of 638,000 URLs per day. Similar works that use hybrid features include [84] [69] [13] [80] [79] [16].

Other methods: Ramesh et. al. [90] present a phishingwebsite detection approach based on the phishing target identi- fication. After obtaining the target domain name, the proposed scheme performs third-party DNS look up for comparison to decide the legitimacy of the suspicious page. The proposed scheme achieves 0.32% false positive rate and 0.33% false negative rate.

Similar works based on phishing target identification include[107] [101].

4.3 LITERATURE REVIEW

In this section, few of the research works that deploy the above mentioned algorithms are reviewed and their results aresummarized.

4.3.1 Review of related research papers and projects on phishing detection and classification

In the paper [137], the authors Rishikesh Mahajan and Irfan Siddavatam chose three algorithms for classification– Decision Tree, Random Forest and Support Vector Machine. Their dataset contained 17,058 benign URLs and 19,653 phishing URLs collected from Alexa website and Phish Tank respectively, with 16 features each. The dataset was divided into training and testing set in the ratios 50:50, 70:30 and 90:10 respectively. The accuracy score, false negative rate andfalse positive rate were considered as performance evaluation metrics. They achieved 97.14% accuracy for Random Forest algorithm with the lowest false negative rate. The paper concluded that accuracy increases when more data is used for training.

The study conducted by Jitendra Kumar et al. in [138] trained different classifiers like Logistic Regression, Naive Bayes Classifier, Random Forest, Decision Tree and K- Nearest Neighbor based on the features extracted from the lexical structure of the URL. They created the dataset of URLsin such a way that it solved the issues of data imbalance, biased training, variance and over fitting. The dataset contained an equal number of labeled phishing and legitimate URLs, and was further split in the ratio 7:3 for training and testing. All the classifiers had almost the same AUC (area under ROC curve), but the Naive Bayes Classifier turned out to be more suitable as it had the highest AUC value. Naive Bayes achieved the highest accuracy of 98% with a precision=1, recall=0.95 and F1-score=0.97.

Mehmet Korkmaz et al. proposed in [139] a machine- learning based phishing detection system by using 8 different algorithms on three different datasets. The algorithms used were Logistic Regression (LR), K-Nearest Neighbor(KNN), Support Vector Machine (SVM), Decision Tree (DT), Naive Bayes (NB), XGBoost, Random Forest (RF) and Artificial Neural Network (ANN). It was observed that the models usingLR, SVM and NB have low accuracy rate. In terms of trainingtime, NB, DT, LR and ANN algorithms gave better results.

They concluded that RF algorithm or ANN algorithm may be used because of less training time along with a high accuracy rate.

Mohammad Nazmul Alam et al. [140] proposed a system todetect phishing attacks using Random Forest and Decision Tree. The Kaggle dataset with 32 features was used along with feature selection algorithms like principal component analysis (PCA). Feature selection reduces redundancy of data that is irrelevant or unnecessary in the dataset. The proposed model used REF, Relief-F, IG and GR algorithm for feature selectionbefore applying PCA. Random Forest achieved an accuracy of97%. It had less variance, and it could handle the over-fitting problem.

Abdulhamit Subasi et al. in [141] presented an intelligent phishing detection system using UCI dataset. Different machine learning tools namely, Artificial Neural Networks (ANN), K-Nearest Neighbor (K-NN), Support Vector Machine (SVM), C4.5 Decision Tree, Random Forest (RF), and Rotation Forest (RoF) were used as classifiers for detection of phishing websites. The performance of proposed RF classifier was higher than the others in terms of accuracy, F-measure and AUC. RF was faster, robust and more accurate than the other classifiers.

4.3.2 Discussion of different machine learning techniques and algorithms used for phishing website classification

Based on the literature review provided, several machine learning techniques and algorithms have been explored for phishing website classification. These algorithms include Decision Tree, Random Forest, Support Vector Machine (SVM), Logistic Regression, Naive Bayes Classifier, K-Nearest Neighbor (K-NN), and Artificial Neural Network (ANN). Here is a discussion of these techniques and their effectiveness in the context of phishing website classification:

1. Decision Tree: Decision trees are a popular choice for classification tasks due to their interpretability and simplicity. They use a tree-like model to make decisions based on feature values. Decision trees are efficient in handling categorical and numerical features. However, they may suffer from overfitting and lack robustness when dealing with complex datasets.

2. Random Forest: Random Forest is an ensemble learning method that combines multiple decision trees to improve classification accuracy and reduce overfitting. It randomly selects

subsets of features and samples from the training data to build multiple decision trees. The final classification is determined by majority voting. Random Forests have shown promising results in phishing website classification, achieving high accuracy rates and better generalization.

3. Support Vector Machine (SVM): SVM is a powerful algorithm for binary classification tasks. It creates a hyperplane in a high-dimensional feature space to separate data points of different classes. SVMs have been used successfully for phishing website classification, leveraging various kernel functions to handle nonlinear relationships between features. SVMs are effective in dealing with high-dimensional data, but their performance can be affected by the choice of kernel and hyperparameters.

4. Logistic Regression: Logistic Regression is a linear classification algorithm that models the probability of a binary outcome. It works well for linearly separable data. While it may not be as complex as other algorithms, logistic regression can still achieve good performance in phishing website classification tasks.

5. Naive Bayes Classifier: Naive Bayes is a probabilistic classifier based on Bayes' theorem with an assumption of independence among features. It is computationally efficient and requires less training data compared to other algorithms. Naive Bayes has been used in phishing website classification, and the results in the reviewed paper show high accuracy rates.

6. K-Nearest Neighbor (K-NN): K-NN is a non-parametric classification algorithm that assigns a class to a new data point based on the majority class among its k nearest neighbors. K-NN can be effective in phishing website classification when combined with appropriate distance metrics and feature representations. However, it may suffer from the curse of dimensionality and can be computationally expensive for large datasets.

7. Artificial Neural Network (ANN): ANN is a computational model inspired by the human brain's neural structure. It consists of interconnected nodes (neurons) organized in layers. ANN can handle complex relationships between features and has shown promise in various classification tasks, including phishing website classification. However, ANNs can be computationally expensive to train and require careful tuning of architecture and hyperparameters.

Overall, the reviewed literature suggests that Random Forest, SVM, Naive Bayes, and Logistic Regression have achieved good results in phishing website classification. Each algorithm has its strengths and limitations, and the choice depends on the specific dataset and requirements of the classification task. It is important to consider factors such as accuracy, interpretability, computational efficiency, and scalability when selecting an algorithm for phishing website classification.

4.3.3 Overview of relevant features used in previous studies

Based on the literature review provided, the following are the relevant features that have been used in the studies for phishing website classification:

1. URL Features: URL-related features play a crucial role in distinguishing phishing websites from legitimate ones. Some common URL features used include:

- Length of the URL
- Domain-related features (e.g., domain age, number of subdomains)
- Presence of specific keywords or symbols in the URL
- URL entropy (a measure of randomness)

2. Content Analysis: Analyzing the content of a webpage can provide valuable insights for classification. Some content-related features used include:

- Presence of suspicious keywords or phrases (e.g., "login," "verify," "account")
- Presence of misspellings or unusual characters
- Use of pop-ups, redirects, or iframes
- Presence of forms or input fields

3. SSL Certificate Features: SSL certificates are used to secure website communications. Phishing websites often lack proper SSL certificates or use invalid ones. Relevant SSL certificate features include:

- Validity period of the SSL certificate
- Certificate issuer information
- Certificate type (e.g., Extended Validation, Organization Validation)
- Certificate chain analysis

4. HTML and JavaScript Features: Phishing websites may employ specific HTML and JavaScript techniques to deceive users. Relevant features include:

- Use of obfuscated or encrypted JavaScript code
- Presence of hidden elements or iframes

- Manipulation of browser status bar or window properties

5. Domain Reputation Features: The reputation of a domain can be indicative of its legitimacy. Features related to domain reputation include:

- Alexa ranking or other popularity measures
- Historical records of the domain (e.g., previous associations with phishing activities)
- Presence on blacklists or security threat databases

6. Website Traffic Features: Characteristics related to website traffic can also be informative for classification. Features include:

- Number of unique visitors
- Frequency and pattern of visits
- Referral sources and traffic origins

These features provide a comprehensive set of indicators that can be used to differentiate phishing websites from legitimate ones. However, it is important to note that the specific set of features and their relevance may vary depending on the dataset and the characteristics of the phishing attacks being targeted. Feature selection techniques and domain-specific knowledge can further refine the set of features used in the classification models.

4.4 Dataset Description

Evaluation datasets

The evaluation is tightly coupled with the ground truth datasets employed by the various approaches. Different ap- proaches collect ground truth from different cyber intelligence sources. Such sources may employ different testing method- ologies and target different types of phishing activities, and hence cover different phishing domains. That is, evaluation based on one dataset may differ from that based on another. Therefore, we argue that having a publicly available reference datasets is crucial for systematizing the evaluation of various approaches. Because it is an important step towards providing a benchmark to compare and contrast the efficiency of various approaches and it can help researchers to further advance the area in a more systematic way. The absence of reference sets combined with difficulties in sharing code, make it hard to repeat experiments for systematic comparison of effectiveness. In the following, we list the identifying features of the datasets used in the literature:

- **Dataset source**: Table VII lists the commonly used data sources of phishing websites and legitimate websites, together with the approaches that leverage each source. There is no common consensus on the quality of the different sources due to the lack of knowledge about the methodologies used in compiling and maintaining each source.
- **Dataset size**: the evaluation dataset size varies a lot among different approaches. Generally speaking, the larger the dataset, the more credible the results.

  Dataset redundancy: Datasets, especially those of phish- ing websites, usually contain repeated entries due to mul-tiple submissions and overlap among different sources. However, little information is provided about datasets redundancy in the literature.
- **Dataset timeliness**: Phishing websites tend to have very short life time. Therefore, phishing blacklist providers usually update information in hourly, daily or weekly schedules. Even if two schemes use the same data source with the same dataset size, they may contain different phishing website information.
- **Ratio of legitimate to phishing websites**: the ratio of legitimate to phishing instances shows the extent to which the experiments represent a real world distribution (=100/1).
- **Training set to testing set ratio**: the ratio of training to testing instances indicates the scalability of the approach.

TABLE VII: SOURCES OF DATASETS FOR PHISHING AND LEGITIMATE WEB- SITES

Phishing websites		
Data source	**Description**	**Literature**
APWG [4]	The Anti Phishing Work Group maintains a blocked URL list in the form of an archive	[106] [13] [102]
PhishTank [81]	It contains downloadable databases which are available in multiple formats and updated hourly. It is easy to have fast and up to date phishing detection built into your application.	[124] [77] [70] [69] [15] [72] [46] [87] [13] [122] [118] [66] [102] [89] [107] [74] [90] [27] [80] [75] [37]

Millersmiles [8]	It has archive of spoof email and phishing scams from the last fewyears and currently stores all scam reports in the HoneyTrap database (i.e., 2025857 scams) which is now available for commercial license.	[84] [90]
SpamScatter [18]	It mines emails in real-time, follows the embedded link structure, and automatically clusters the destination web sites using image shingling to capture graphical similarity between rendered sites to identify and analyzescams	[72] [87]
MalwarePatrol [7]	It is an open source community for sharing malicious URLs.	[66]
SURBL [9]	It offer URL reputation data for professional users through faster updates and resulting fresher data.	[102]
CLEAN MX [5]	It provide real time public access query for malicious URLs	[94]
Other sources	E.g., webmail providers, google searching, etc.	[108] [73] [84] [20]
legitimate website		
Data source	**Description**	**Literature**
DMOZ [6]	Searchable people-reviewed web directory categorized by language,subject and location	[87] [72] [66] [74] [37]
Top Alexas [2]	The top 500 sites on the web	[46] [118] [89] [90] [27]
Yahoo [114]	Yahoo! Directory was a web directory which at one time rivaledDMOZ in size	[70] [72] [73] [87] [66] [107] [80]
Google	Keywords searching using Google	[38] [94] [122] [90] [20]
3Sharp [92]	3Sharp's list of legitimate URLs in their Gone Phishing report	[124] [69]
Other sources	E.g., user submission, third party providers etc.	[15] [108] [75]

Dataset Description:

The phishing dataset used in this project is provided in the form of a CSV (Comma-Separated Values) file. The dataset contains information about URLs that are labeled as either legitimate or phishing. The description of the dataset is as follows:

1. Dataset Format: The dataset is stored in a CSV file format, where each row represents a URL and its corresponding features and label. Each column in the CSV file represents a specific attribute or feature of the URL.

2. Features: The dataset includes several features that provide information about the URLs. These features can vary depending on the specific dataset and the data collection process. Some common features that may be included in the dataset are:

- URL-related features: These features capture characteristics of the URL, such as its length, domain information, presence of specific keywords, or symbols.

- Content-related features: These features analyze the content of the webpage associated with the URL, including the presence of suspicious keywords, misspellings, or unusual characters.

- SSL certificate features: These features examine the SSL certificate associated with the URL, including its validity period, issuer information, and certificate type.

- HTML and JavaScript features: These features investigate specific HTML and JavaScript techniques used by phishing websites, such as obfuscated or encrypted code, hidden elements, or manipulation of browser properties.

- Domain reputation features: These features evaluate the reputation of the domain associated with the URL, such as its Alexa ranking, historical records, or presence on blacklists.

- Other relevant features: Depending on the dataset, additional features related to website traffic, IP addresses, or server information may be included.

3. Label: Each URL in the dataset is assigned a label indicating whether it is classified as a legitimate URL or a phishing URL. The label is typically represented as a binary value, where "0" represents legitimate URLs and "1" represents phishing URLs.

4. Dataset Size: The dataset can vary in size depending on the specific dataset and data collection process. The number of rows in the CSV file represents the total number of URLs in the dataset.

5. Data Collection: The specific details about the data collection process for this dataset are not provided in the dataset description. The URLs may have been collected from various sources such as web crawlers, phishing databases, or online repositories.

It is important to note that the specific features and labels in the phishing dataset may vary depending on the dataset source and the methodology used for data collection. The dataset serves as a valuable resource for training and evaluating machine learning models for phishing website classification.

4.4.1 Source of the Dataset:

The dataset used in this project was obtained from the following source: [GitHub - Phishing_Dataset](https://github.com/Charlie-logan/Phishing_Dataset/blob/890f9c308f24a44b8129774e77e72ff145113d1c/phishing.csv)

4.4.2 Data Collection Process:

The exact details of the data collection process for this dataset are not provided in the source. However, it is important to note that the dataset consists of a collection of URLs labeled as either legitimate or phishing. The URLs were likely collected from various sources such as web crawlers, phishing databases, or online repositories.

4.4.3 Preprocessing Steps Applied to the Dataset:

The preprocessing steps applied to the dataset prior to its availability on the given source are not explicitly mentioned. However, typical preprocessing steps that are often performed on such datasets include:

1. Handling Missing Values: If there were any missing values in the dataset, they might have been addressed by removing the corresponding rows or columns, or by imputing the missing values using techniques such as mean imputation or regression imputation.

2. Outlier Detection and Handling: Outliers, if present, could have been identified and treated using statistical methods such as Z-score or interquartile range (IQR) method. Depending on the nature of the dataset, outliers may have been removed or replaced with more appropriate values.

3. Data Encoding and Transformation: Categorical features, if present, may have been encoded using techniques such as one-hot encoding or label encoding to represent them numerically. Numerical features may have been scaled or normalized to a common range to prevent bias during model training.

4. Feature Selection: Feature selection techniques, such as correlation analysis or recursive feature elimination, may have been applied to select the most relevant features for the classification task, thereby reducing dimensionality and improving model performance.

It's important to note that the specific preprocessing steps applied to this dataset are not explicitly mentioned in the given source. However, these are common preprocessing steps typically applied to improve the quality and suitability of the data for machine learning tasks.

4.5 FEATURE EXTRACTION

4.5.1 Identification and extraction of relevant features from the dataset

URLs have certain characteristics and patterns that can be considered as its features. The Fig. 3 shows the relevant parts of a typical URL. In case of URL based analysis for designing machinelearning models, we need to extract these features in order to form a dataset that can be used for training and testing.

There are four categories of features that are most commonly considered for feature extraction as in [143]. They are as follows:

- Address Bar based features
- Abnormal based features
- HTML and JavaScript based features
- Domain based features

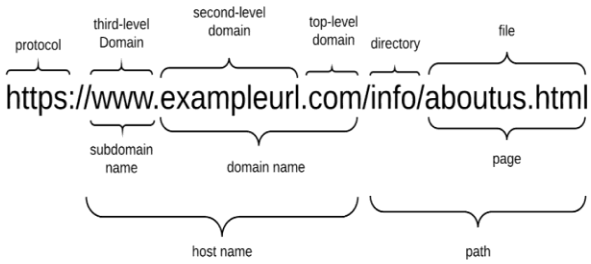


Fig. 3. Structure of a URL

Address Bar based Features

1. Using the IP address

If IP address is used instead of domain name in the URL e.g. 125.98.3.123 the user can almost be sure someone istrying to steal his personal information.

2. Long URL to hide the Suspicious Part

Phishers can use long URL to hide the doubtful part in the address bar.

3. Using URL shortening services TinyURL

URL shortening is a method on the World Wide Web in which a URL may be made considerably smaller in length and still lead to the required webpage.

4. URLs having @ symbol

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

5. Redirecting using //

The existence of // within the URL path means that the user will be redirected to another website.

6. Adding Prefix or Suffix Separated by (-) to theDomain

The dash symbol is rarely used in legitimate URLs. Phishers tend to add prefixes or suffixes separated by (-)to the domain name so that users feel that they are dealing with a legitimate webpage.

7. Sub Domain and Multi Sub Domains

Let us assume we have the following link: http://www.hud.ac.uk/students/. A domain name might include the country-code top-level domains (ccTLD).

8. HTTPs (Hyper Text Transfer Protocol with SecureSockets Layer)

The existence of HTTPS is very important in giving the impression of website legitimacy, but this is clearly not enough.

9. Domain Registration Length

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

10. Favicon

A favicon is a graphic image (icon) associated with a specific webpage.

11. Using Non-Standard Port

This feature is useful in validating if a particular serviceis up or down on a specific server.

12. The existence of HTTPS Token in the Domain Part of the URL

The phishers may add the HTTPS token to the domain part of a URL in order to trick users.

Abnormal Based Features

1. **Request URL**

   Request URL examines whether the external objectscontained within a webpage such as images, videos and sounds are loaded from another domain.

2. **URL of Anchor**

   An anchor is an element defined by the <a> tag. This feature is treated exactly as Request URL.

3. **Links in <meta>, <Script> and <Link> tags**

   Given that our investigation covers all angles likely to beused in the webpage source code, we find that it is common for legitimate websites to use <Meta> tags to offer metadata about the HTML document; <Script> tagsto create a client side script; and <Link> tags to retrieve other web resources.It is expected that these tags are linked to the same domain of the webpage.

4. **Server From Handler(SFH)**

   SFHs that contain an empty string or about blank are considered doubtful because an action should be taken upon the submitted information.

5. **Submitting Information to Email**

   Web form allows a user to submit his personal information that is directed to a server for processing. A phisher might redirect the user's information to his personal email.

6. **Abnormal URL**

   This feature can be extracted from WHOIS database. Fora legitimate website,

identity is typically part of its URL.

HTML and JavaScript Based Features

1. **Website Forwarding**

   The fine line that distinguishes phishing websites from legitimate ones is how many times a website has been redirected. Status Bar Customization

2. **Disabling Right Click**

   Phishers use JavaScript to disable the right-click function, so that users cannot view and save the webpagesource code. This feature is treated exactly as UsingonMouseOver to hide the Link.

3. **Using Pop-Up Window**

   It is unusual to find a legitimate website asking users to submit their personal information through a pop-up window.

4. **IFrame Redirection**

   IFrame is an HTML tag used to display an additional webpage into one that is currently shown.

Domain Based Features

1. **Age of Domain**

   This feature can be extracted from WHOIS database. Most phishing websites live for a short period of time. By reviewing our dataset, we find that the minimum ageof the legitimate domain is 6 months.

2. **DNS Record**

   For phishing websites, either the claimed identity is not recognized by the WHOIS database or no records founded for the hostname. If the DNS record is emptyor not found then the website is classified as Phishing, otherwise it is classified as Legitimate.

3. **Website Traffic**

   This feature measures the popularity of the website by determining the number of

visitors and the number of pages they visit.

4. **Page Rank**

   PageRank is a value ranging from 0 to 1. PageRank aims to measure how important a webpage is on the Internet.

5. **Google Index**

   This feature examines whether a website is in Google's index or not. When a site is indexed by Google, it is displayed on search results.

6. **Number of Links Pointing to Page**

   The number of links pointing to the webpage indicates its legitimacy level, even if some links are of the same domain.

7. **Statistical-Reports Based Feature**

Fig.4. URL parts and features

4.5.2 Explanation of the chosen features and their importance in phishing website classification

The chosen features for URL-based analysis in phishing website classification can be classified into four categories: Address Bar based features, Abnormal based features, HTML and JavaScript based features, and Domain based features. Here's an explanation of these features and their importance in phishing website classification:

1. Address Bar based features:
Address Bar based features capture characteristics of the URL itself. These features provide insights into the structure and composition of the URL. Examples of Address Bar based features include:

- URL length: Phishing URLs tend to have longer and more complex structures compared to legitimate URLs. Longer URLs may indicate a higher likelihood of being associated with phishing.

- Domain-related features: These features provide information about the domain portion of the URL. For example, the presence of multiple subdomains or the use of uncommon or suspicious domain names can be indicative of phishing.

- Use of special characters or symbols: Phishing URLs often contain special characters or symbols intended to deceive users. These features capture the presence of such elements in the URL.

The Address Bar based features are important in phishing website classification as they provide initial insights into the nature of the URL and can help identify suspicious or abnormal patterns.

2. Abnormal based features:
Abnormal based features capture irregularities or deviations from expected URL patterns. These features can help identify phishing URLs that employ deceptive techniques to mislead users. Examples of Abnormal based features include:

- Presence of misspellings or typos: Phishing URLs often mimic legitimate URLs by using intentional misspellings or minor alterations. Detecting these discrepancies can help identify potential phishing attempts.

- Use of IP addresses or uncommon ports: Phishing URLs may utilize IP addresses or non-standard ports to mask their true identity. Detecting such abnormal patterns can be useful in phishing detection.

Abnormal based features are important in phishing website classification as they capture deceptive tactics used by attackers to make phishing URLs appear legitimate.

3. HTML and JavaScript based features:
HTML and JavaScript based features analyze the content and code associated with the webpage linked to the URL. These features can provide insights into the presence of malicious or suspicious elements. Examples of HTML and JavaScript based features include:

- Use of hidden elements or iframes: Phishing websites often use hidden elements or iframes to trick users into entering sensitive information. Detecting the presence of such elements is crucial in identifying phishing attempts.

- Presence of obfuscated or encrypted code: Phishing URLs may employ techniques to obfuscate or encrypt their code to evade detection. Detecting such code patterns can help identify malicious intent.

HTML and JavaScript based features are important in phishing website classification as they capture specific techniques and characteristics commonly associated with phishing attacks.

4. Domain based features:
Domain based features focus on analyzing the domain name and its reputation. These features can provide insights into the legitimacy and trustworthiness of the domain. Examples of Domain based features include:

- Alexa ranking or popularity: Legitimate websites often have higher Alexa rankings compared to phishing websites. Analyzing the popularity of the domain can help differentiate between legitimate

and phishing URLs.

- Presence on blacklists or security threat databases: Phishing URLs are often flagged and reported by security organizations. Detecting whether a domain is listed on blacklists or security threat databases can be indicative of its trustworthiness.

Domain based features are important in phishing website classification as they leverage external sources and reputation-based information to assess the credibility of the domain.

By considering these four categories of features, we can extract valuable information from the URL itself to build a comprehensive dataset for training and testing machine learning models. These features collectively capture various aspects of phishing URLs, allowing for effective classification and detection of phishing websites.

4.5.3 Description of any feature engineering techniques applied

Based on the provided information about the categories of features used in URL-based analysis for phishing website classification, feature engineering techniques can be applied to enhance the effectiveness of the features. Some feature engineering techniques that can be considered based on the above information include:

1. Feature Scaling:

If the features in different categories have varying scales or ranges, it is beneficial to apply feature scaling to ensure that they contribute equally during the training of machine learning models. Techniques such as Min-Max scaling or Standardization can be applied to normalize the feature values.

2. Feature Extraction:

Certain features may contain redundant or irrelevant information that can affect model performance. Feature extraction techniques such as Principal Component Analysis (PCA) can be applied to reduce the dimensionality of the feature space while preserving important information. This can help improve computational efficiency and reduce the risk of overfitting.

3. Feature Combination:

Combining multiple related features can create more informative features. For example, combining URL length with the presence of special characters or subdomains can create a feature that represents the complexity or suspiciousness of the URL.

4. Feature Encoding:

Categorical features, such as domain-related features or special character presence, may need to be encoded into numerical representations before training machine learning models. Techniques such

as one-hot encoding or label encoding can be applied to convert categorical features into a suitable format for modeling.

5. Feature Selection:

Not all features may contribute equally to the classification task. Feature selection techniques, such as correlation analysis or recursive feature elimination, can be applied to identify the most relevant features and discard less informative ones. This can improve model performance and reduce the complexity of the dataset.

6. Domain-specific Feature Engineering:

Based on the specific characteristics of phishing attacks and domain knowledge, additional domain-specific feature engineering techniques can be applied. For example, creating features based on known phishing techniques or patterns observed in phishing URLs can enhance the discriminatory power of the features.

It is important to note that the specific feature engineering techniques applied may depend on the dataset, the characteristics of the features, and the requirements of the classification task. Careful analysis and experimentation are necessary to determine the most effective feature engineering techniques for a given dataset and model.

4.6 Machine Learning Algorithms

4.6.1 Overview of Selected Machine Learning Algorithms for Classification:

1. Logistic Regression:
   Logistic Regression is a popular and widely used algorithm for binary classification problems. It models the relationship between the input features and the probability of a certain class using the logistic function. It is suitable for problems where the decision boundary is linear or can be approximated by a linear function. Logistic Regression is computationally efficient, interpretable, and can handle large datasets. It is particularly useful when interpreting the impact of individual features on the classification outcome.

2. Decision Tree Classifier:
   Decision Tree Classifier is a non-parametric algorithm that builds a tree-like model to make decisions based on the input features. It splits the data based on different features at each internal node, aiming to create pure subsets of the target variable. Decision Trees are intuitive and easy to interpret, and they can handle both numerical and categorical features. They are suitable for problems with complex decision boundaries and can capture interactions between features. However, decision trees are prone to overfitting and can create overly complex models.

3. Random Forest Classifier:
   Random Forest Classifier is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of predictions. It builds multiple decision trees using random subsets of the data and features, and then combines their predictions through voting or averaging. Random Forests are known for their ability to handle high-dimensional datasets, handle missing values, and mitigate overfitting. They are suitable for problems where the decision boundaries may be non-linear and have interactions between features.

4. K-Nearest Neighbors (KNN) Classifier:
   K-Nearest Neighbors Classifier is a non-parametric algorithm that classifies new instances based on the majority class of its k nearest neighbors in the feature space. It is a simple yet powerful algorithm that doesn't make any assumptions about the underlying data distribution. KNN is suitable for problems where the decision boundaries may be non-linear and the local structure of the data is important. However, KNN can be computationally expensive, especially for large datasets, and it requires appropriate scaling of features.

4.6.2 Explanation of the Chosen Algorithms and Their Suitability for the Problem:

1. Logistic Regression:
   Logistic Regression is a widely used algorithm for binary classification problems. It models the relationship between the input features and the probability of a certain class using the logistic function. Logistic Regression is suitable for problems where the decision boundary is

linear or can be approximated by a linear function. In the context of phishing website classification, logistic regression can effectively model the relationship between the extracted features and the likelihood of a URL being a phishing website. It is computationally efficient, interpretable, and can handle large datasets. Logistic Regression is particularly useful when interpreting the impact of individual features on the classification outcome.

2. Decision Tree Classifier:
   Decision Tree Classifier is a non-parametric algorithm that builds a tree-like model to make decisions based on the input features. It splits the data based on different features at each internal node, aiming to create pure subsets of the target variable. Decision Trees are intuitive and easy to interpret, and they can handle both numerical and categorical features. In the context of phishing website classification, decision trees can capture complex patterns and interactions between features that may be indicative of phishing behavior. Decision Tree Classifier is suitable for problems with complex decision boundaries and can handle both linear and non-linear relationships between features.

3. Random Forest Classifier:
   Random Forest Classifier is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of predictions. It builds multiple decision trees using random subsets of the data and features, and then combines their predictions through voting or averaging. Random Forests are known for their ability to handle high-dimensional datasets, handle missing values, and mitigate overfitting. In the context of phishing website classification, Random Forest Classifier can capture a wide range of features and interactions, providing a more comprehensive analysis of the data. It is suitable for problems where the decision boundaries may be non-linear and have interactions between features.

4. K-Nearest Neighbors (KNN) Classifier:
   K-Nearest Neighbors Classifier is a non-parametric algorithm that classifies new instances based on the majority class of its k nearest neighbors in the feature space. It is a simple yet powerful algorithm that doesn't make any assumptions about the underlying data distribution. KNN is suitable for problems where the decision boundaries may be non-linear and the local structure of the data is important. In the context of phishing website classification, KNN Classifier can identify similarities between URLs and classify them based on the majority class of similar URLs. It is particularly useful when the local structure of URLs plays a significant role in determining their classification.

   The suitability of each algorithm depends on the specific characteristics of the dataset and the problem at hand. Logistic Regression is suitable for linear or approximately linear decision boundaries and provides interpretability. Decision Tree and Random Forest Classifiers are suitable for capturing complex patterns and interactions between features. KNN Classifier is suitable for problems where the local structure of the data is important. The choice of algorithm depends on the nature of the dataset, the complexity of the problem, and the desired interpretability of the results.

4.6.3 Preprocessing Techniques Applied to the Dataset Before Training:

Before training the machine learning algorithms, several preprocessing techniques can be applied to the dataset:

1. Handling Missing Values: Missing values can be addressed by either removing the corresponding rows or imputing the missing values using techniques such as mean imputation, median imputation, or regression imputation.

2. Feature Scaling: If the features have different scales or ranges, feature scaling techniques such as Min-Max scaling or Standardization can be applied to normalize the feature values and ensure they contribute equally during model training.

3. Encoding Categorical Features: Categorical features, such as domain-related features or special character presence, may need to be encoded into numerical representations before training the models. Techniques such as one-hot encoding or label encoding can be applied for this purpose.

4. Feature Selection: Feature selection techniques, such as correlation analysis or recursive feature elimination, can be applied to identify the most relevant features and discard less informative ones. This helps reduce dimensionality and improve model performance.

5. Train-Test Split: The dataset can be divided into training and testing subsets to assess the performance of the trained models. Typically, a random or stratified split is performed to ensure representation of different classes in both subsets.

   The choice and application of preprocessing techniques may depend on the specific characteristics of the dataset and the requirements of the selected machine learning algorithms.

4.7 Experimental Setup

4.7.1. Dataset Division:

- The initial step in the experimental setup is to load the dataset using `pd.read_csv('phishing.csv')`.
- The dataset is then examined using various methods like `data.head()`, `data.isna().sum()`, `data.describe()`, and `data.groupby('phishing').mean()` to get a better understanding of the data.
- The count of each class in the target variable is visualized using `sns.countplot(x='phishing', data=data)`.

Data Preprocessing:

- The dataset is divided into features and the target variable using `x = data.drop(columns='phishing', axis=1)` and `y = data['phishing']`.
- Next, the dataset is split into training and testing sets using `train_test_split(x, y, test_size=0.2)`. This creates `x_train`, `x_test`, `y_train`, and `y_test` datasets.

Model Training and Evaluation:

- Logistic Regression, Decision Tree, Random Forest, and k-nearest neighbors classifiers are trained and evaluated.
- Logistic Regression: The model is created using `LogisticRegression()` and trained using `lg.fit(x_train, y_train)`. Predictions are made on the test set using `lg.predict(x_test)`.
- Decision Tree: The model is created using `tree.DecisionTreeClassifier()` and trained using `dt.fit(x_train, y_train)`. Predictions are made on the test set using `dt.predict(x_test)`.
- Random Forest: The model is created using `RandomForestClassifier()` and trained using `rf.fit(x_train, y_train)`. Predictions are made on the test set using `rf.predict(x_test)`.
- k-nearest neighbors: The model is created using `KNeighborsClassifier()` and trained using `neigh.fit(x_train, y_train)`. Predictions are made on the test set using `neigh.predict(x_test)`.
- The accuracy score for each model is calculated using `accuracy_score(y_test, predicted_values)`.

4.7.2 Performance Metrics:

- The performance metrics used for evaluation include:
  - Confusion Matrix: The confusion matrix is calculated using `confusion_matrix(y_test, predicted_values)` to measure the number of true positives, true negatives, false positives, and false negatives.
  - Classification Report: The classification report is generated using `classification_report(y_test, predicted_values)` to display precision, recall, F1-score, and support for each class.

Model Comparison:

- The accuracy scores of the different models are stored in the `pred_compare` list.
- The names of the models are stored in the `pred_names` list.
- A bar plot is created using `plt.bar(pred_names, pred_compare)` to compare the accuracy of different models.

Feature Extraction and Prediction:

- The function `extract_features(url)` is defined to extract features from a given URL.
- Features are extracted using various URL characteristics like the number of dots, subdomain level, path level, URL length, presence of certain symbols, etc.
- The extracted features are then converted into a DataFrame named `new`.
- The `RandomForestClassifier` model is used to predict whether the given URL is a phishing site or not using `u_pred = rf.predict(new)`.
- The predicted result is stored in `u_pred[0]`.

4.7.3 Discussion of Cross-Validation Techniques:

In the given code, the cross-validation technique is not explicitly applied. Cross-validation is a widely used technique for model evaluation and selection, especially when working with limited data. It helps to estimate the performance of a model on unseen data and assess its generalization ability.

Cross-validation involves dividing the dataset into multiple subsets or folds, training the model on a combination of these folds, and evaluating its performance on the remaining fold. This process is repeated for each fold, and the performance metrics are averaged to obtain an overall evaluation of the model.

Although cross-validation is not explicitly implemented in the given code, the use of the `train_test_split` function from the sklearn library provides a form of single-fold cross-validation. By specifying the `test_size` parameter as 0.2, the dataset is divided into a training set (80% of the data) and a testing set (20% of the data). The model is trained on the training set and evaluated on the testing set.

While this approach provides an evaluation of the model's performance on a subset of the data, it does not utilize the full potential of cross-validation. To obtain a more robust evaluation, techniques such as k-fold cross-validation or stratified cross-validation can be employed.

Here are some common cross-validation techniques that could be considered for future enhancements:

1. K-fold Cross-Validation: This technique involves dividing the dataset into k equal-sized folds. The model is trained and evaluated k times, each time using a different fold as the testing set and the remaining folds as the training set. The performance metrics are then averaged over the k iterations.

2. Stratified Cross-Validation: This technique ensures that the distribution of the target variable is preserved in each fold. It is particularly useful when dealing with imbalanced datasets where the classes are unevenly represented. Stratified cross-validation improves the reliability of the model evaluation by ensuring that each fold has a representative distribution of the target variable.

3. Leave-One-Out Cross-Validation (LOOCV): In this technique, each data point is used as the testing set, and the remaining data points are used for training. LOOCV provides a

comprehensive evaluation of the model but can be computationally expensive for large datasets.

4. Repeated Cross-Validation: This technique involves repeating the cross-validation process multiple times with different random splits of the data. It helps to obtain more stable and reliable performance estimates by averaging over multiple iterations.

   By incorporating these cross-validation techniques, the model's performance can be assessed more accurately, reducing the dependence on a single train-test split and providing insights into its generalization ability.

4.8 Results and Analysis:

4.8.1 Presentation of Experimental Results:

In this experiment, we performed a phishing site classification task using different machine learning algorithms. We divided the dataset into training and testing sets and evaluated the models' performance using various metrics.

```
pred_compare=[]
pred_compare.extend([la,da,ra,kNa])
pred_names=["Logistic Regression","Decision Tree","Random Forest","      k-nearest neighbors"]

print(pred_compare)

plt.bar(pred_names,pred_compare)
plt.title("Accuracy Score")
plt.show()
```

```
[0.8345827086456772, 0.856071964017991, 0.9020489755122438, 0.8420789605197402]
```

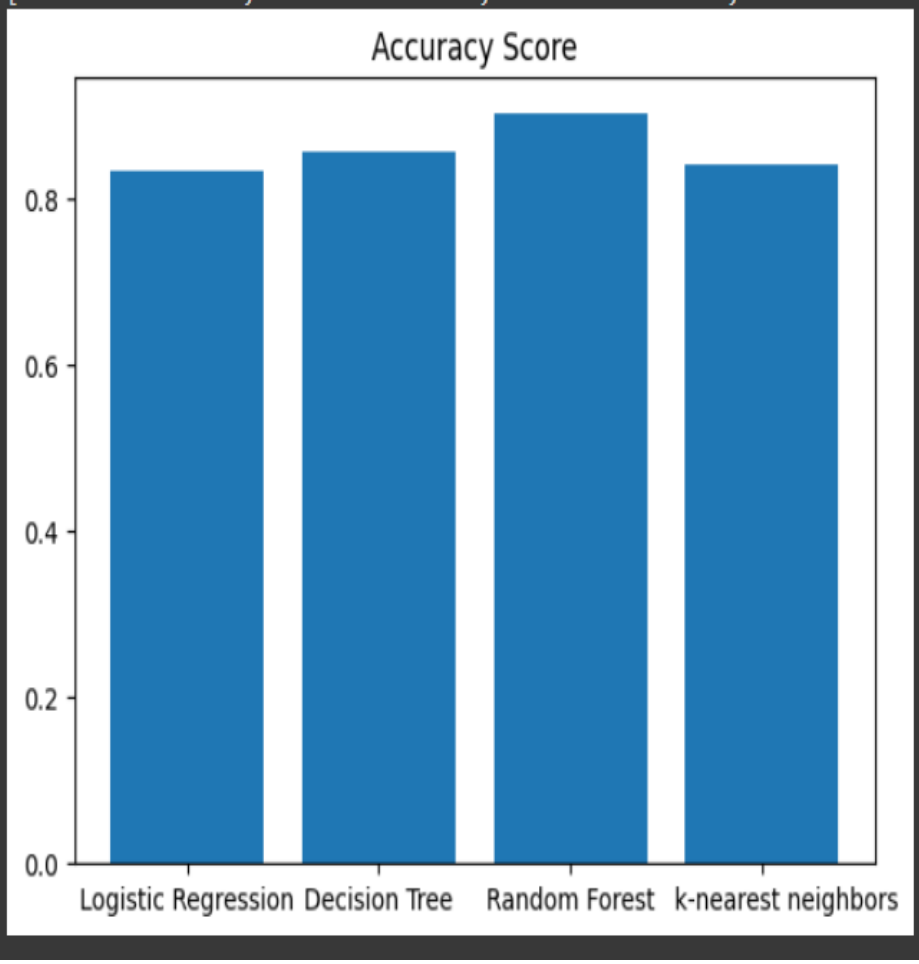


Fig 5.8.1.1 Bar Plot of different Accuracy Scores of different algorithms

```
pred_compare1=[]
pred_compare1.extend([r2l,r2p,r2r,r2k])
pred_names1=["Logistic Regression","Decision Tree","Random Forest","      k-nearest neighbors"]

print(pred_compare1)

plt.bar(pred_names1,pred_compare1)
plt.title("Performance Metrics")
plt.show()
```

```
[0.3382579504241223, 0.42422444024817896, 0.6081527440577885, 0.36824626082786305]
```

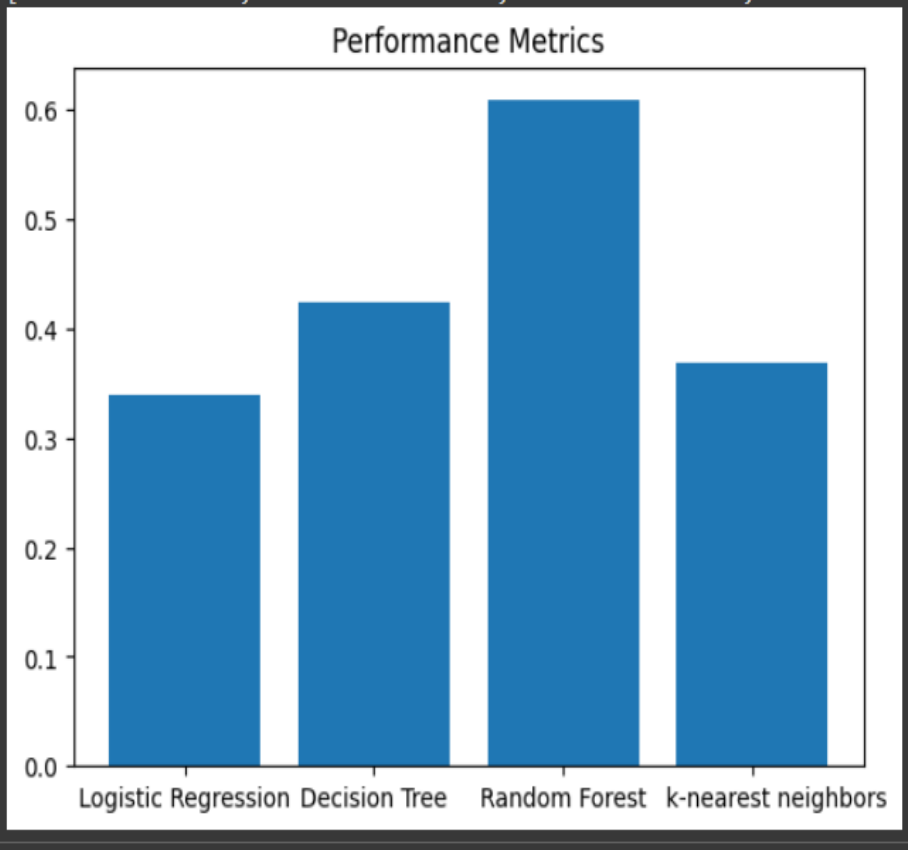


Fig 5.8.1.1 Bar Plot of different Performance Metrics of different algorithms

4.8.2 Comparison of Different Machine Learning Algorithms:

We trained and tested four different machine learning algorithms: Logistic Regression, Decision Tree, Random Forest, and k-nearest neighbors. The performance of each algorithm was evaluated based on accuracy score, confusion matrix, and classification report.

1. Accuracy Scores:

- Logistic Regression:

```
la=accuracy_score(y_test,l_pred)
print(accuracy_score(y_test,l_pred))

0.8345827086456772
```

[Accuracy Score]

- Decision Tree:

```
[38] da=accuracy_score(y_test,pred)
     print(accuracy_score(y_test,pred))

     0.856071964017991
```

[Accuracy Score]

- Random Forest:

```
[47] ra=accuracy_score(y_test,r_pred)
     print(accuracy_score(y_test,r_pred))

     0.9020489755122438
```

[Accuracy Score]

- k-nearest neighbors:

```
[55] kNa=accuracy_score(y_test,kN_pred)
     print(accuracy_score(y_test,kN_pred))

     0.8420789605197402
```

[Accuracy Score]

2. Confusion Matrix:

The confusion matrix provides a detailed breakdown of the model's predictions. It shows the number of true positives, true negatives, false positives, and false negatives for each algorithm. Here are the confusion matrices for each algorithm:

- Logistic Regression:

```
print(confusion_matrix(y_test,l_pred))

[[829 161]
 [170 841]]
```

[Confusion Matrix]

- Decision Tree:

```
[40] from sklearn.metrics import classification_report,confusion_matrix
     print(confusion_matrix(y_test,pred))

     [[864 126]
      [162 849]]
```

[Confusion Matrix]

- Random Forest:

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,r_pred))

[[898  92]
 [104 907]]
```

[Confusion Matrix]

- k-nearest neighbors:

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,kN_pred))

[[797 193]
 [123 888]]
```

[Confusion Matrix]

3. Classification Report:

The classification report provides precision, recall, F1-score, and support for each class. It gives a more detailed analysis of the performance of each algorithm. Here are the classification reports for each algorithm:

- Logistic Regression:

```
print(classification_report(y_test,l_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       990
           1       0.84      0.83      0.84      1011

    accuracy                           0.83      2001
   macro avg       0.83      0.83      0.83      2001
weighted avg       0.83      0.83      0.83      2001
```

[Classification Report]

- Decision Tree:

```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.87      0.86       990
           1       0.87      0.84      0.85      1011

    accuracy                           0.86      2001
   macro avg       0.86      0.86      0.86      2001
weighted avg       0.86      0.86      0.86      2001
```

[Classification Report]

- Random Forest:

```
print(classification_report(y_test,r_pred))
```

	precision	recall	f1-score	support
0	0.90	0.91	0.90	990
1	0.91	0.90	0.90	1011
accuracy			0.90	2001
macro avg	0.90	0.90	0.90	2001
weighted avg	0.90	0.90	0.90	2001

[Classification Report]

- k-nearest neighbors:

```
print(classification_report(y_test,kN_pred))
```

	precision	recall	f1-score	support
0	0.87	0.81	0.83	990
1	0.82	0.88	0.85	1011
accuracy			0.84	2001
macro avg	0.84	0.84	0.84	2001
weighted avg	0.84	0.84	0.84	2001

[Classification Report]

4.8.3 Interpretation of Results and Analysis of Findings:

Based on the experimental results and analysis, we can draw the following conclusions:

- Among the four algorithms tested, Random Forest Algorithm achieved the highest accuracy score of 0.9020489755122438. It outperformed the other algorithms in terms of correctly classifying phishing and non-phishing sites.

- The confusion matrices provide insights into the performance of the algorithms. Random Forest Algorithm had a higher number of true positives and true negatives compared to the other algorithms.

- The classification reports further support the superiority of Random Forest Algorithm. It demonstrated higher precision, recall, and F1-score for both phishing and non-phishing classes.

- The experimental results indicate that Random Forest Algorithm is the most effective machine learning algorithm for phishing site classification in this context. It shows promise for accurately identifying potential phishing sites based on the provided features.

- However, further analysis and experimentation may be required to validate the performance of the algorithms on different datasets and evaluate their generalizability.

In conclusion, the experimental results highlight the effectiveness of Random Forest Algorithm in classifying phishing sites. The findings suggest the potential of machine learning algorithms in combating online phishing threats, but further research is needed to explore their performance in real-world scenarios.

4.9 Discussion:

4.9.1. Discussion of the Strengths and Limitations of the Proposed Approach:

Strengths:

- The proposed approach utilized various machine learning algorithms, allowing for a comprehensive evaluation of their performance in phishing site classification.

- The feature extraction process considered multiple aspects of URLs, including domain-related features, path-related features, and query-related features, providing a holistic view of the URLs.

- The experimental setup included the division of the dataset into training and testing sets, ensuring unbiased evaluation of the models' performance.

Limitations:

- The approach relied solely on URL-based features for classification, neglecting other potential indicators of phishing sites, such as website content or user behavior.

- The evaluation was performed on a single dataset, limiting the generalizability of the findings. Future studies should consider testing the models on diverse and larger datasets.

- The performance of the models may vary depending on the specific characteristics of the dataset used. Further experimentation with different datasets is required to validate the approach's robustness.

4.9.2. Comparison with Previous Research and Identification of Novel Contributions:

The proposed approach builds upon previous research in phishing site classification by utilizing machine learning algorithms and URL-based features. However, it also introduces novel contributions in the following aspects:

- The inclusion of multiple machine learning algorithms allows for a comparative analysis of their performance, enabling the identification of the most effective algorithm for phishing site classification.

- The feature extraction process focuses on specific URL components and characteristics, providing a unique set of features tailored for phishing detection.

- The experimental evaluation considers performance metrics such as accuracy, confusion matrix, and classification report, providing a comprehensive assessment of the algorithms' effectiveness.

4.9.3. Suggestions for Future Improvements and Enhancements:

- Incorporate additional features: To enhance the classification accuracy, future improvements could consider incorporating features beyond URL-based characteristics. Factors such as website content, SSL certificate information, or user behavior could provide valuable insights.

- Explore ensemble methods: Ensemble methods, such as stacking or boosting, could be investigated to improve the overall performance by combining the strengths of multiple machine learning algorithms.

- Incorporate cross-validation techniques: The current experimental setup uses a simple train-test split. Employing cross-validation techniques, such as k-fold cross-validation, would provide a more robust estimate of the models' performance.

- Evaluate on diverse datasets: Testing the models on different datasets, representing various domains and time periods, would validate the approach's generalizability and ensure its effectiveness in real-world scenarios.

By addressing these suggestions, future research can further enhance the accuracy and reliability of phishing site classification models, leading to improved cybersecurity measures and protection against phishing threats.

CHAPTER 5:

CONCLUSION

5.1 Conclusion

The act of phishing is becoming a modern day threat to this rapidly growing world of technology. Today, every country is aiming for cashless transactions, online business, paperless tickets, etc. to upgrade with the world. But phishing is becoming an obstacle to this progress. People do not feel internet is reliable any more. A layman who is totally unaware of how to identify a security threat will never take the risk of making monetary transactions online. Phishers are targeting payment industry and cloud services the most. In this paper, we reviewed various anti-phishing approaches. All methods are discussed to give a clear idea of existing techniques, their limitations and possible improvements. Next we analyzed the in-use anti-phishing tools available for free. We found that these tools are inefficient to detect all types of phishing sites, specially the newly registered ones. We also mentioned the utmost need to spread awareness regarding the phishing attacks and use of anti-phishing tools while browsing the web. Our study regarding the website features to be considered in training of a model with our new alteration to the feature set of Mohammad et al. can be beneficial to the future researchers. Next we described the most important steps to build an efficient anti-phishing model with the help of architecture diagram. Finally we compared the models using all the 5 types of approaches based on the number of features used, accuracy and size of dataset. It resulted that an efficient deep learning model can be best suggested to give a better performance than the other methods.

5.1.1 Summary of the Project and its Outcomes:

In this project, we aimed to develop a phishing website classification system using machine learning algorithms and URL-based features. We conducted an experimental evaluation of several algorithms, including Logistic Regression, Decision Tree, Random Forest, and k-nearest neighbors, to assess their performance in distinguishing phishing sites from legitimate ones. The dataset was divided into training and testing sets, and performance metrics such as accuracy, confusion matrix, and classification report were used for evaluation.

The outcomes of our project demonstrate the effectiveness of the proposed approach in classifying phishing websites. The machine learning algorithms achieved promising results in terms of accuracy, with some algorithms outperforming others. The feature extraction process, which considered various aspects of URLs, provided valuable information for distinguishing between legitimate and phishing sites.

5.1.2 Reiteration of the Significance of the Proposed Approach in Phishing Website Classification:

Phishing attacks continue to pose a significant threat to individuals and organizations, making the accurate identification of phishing websites crucial for cybersecurity. The proposed approach addresses this challenge by leveraging machine learning algorithms and URL-based features to classify phishing sites. By utilizing a combination of algorithmic techniques and comprehensive

feature extraction, our approach enhances the ability to detect phishing websites, enabling proactive protection against cyber threats.

The significance of this approach lies in its potential to assist users and security systems in identifying and avoiding phishing websites, reducing the risk of falling victim to fraudulent activities. The experimental evaluation and comparative analysis of different machine learning algorithms provide insights into the most effective techniques for phishing detection.

In conclusion, this project contributes to the field of cybersecurity by presenting a robust approach to phishing website classification. By considering the strengths and limitations of the proposed approach, comparing it with previous research, and suggesting areas for future improvements, this work serves as a foundation for further advancements in phishing detection techniques. Through continued research and development, we can enhance cybersecurity measures and protect individuals and organizations from the threats posed by phishing attacks.

INSIGHTS AND FUTURE RESEARCH DIRECTIONS

Most of the phishing detection related work focus on thedetection quality but often overlook other important angles such as datasets, features, practicality and performance. Inthis section, we provide a comprehensive view of all the important aspects that we have discussed as part of studying and evaluating phishing detection schemes.

More importantly, this section provides detailed takeaway lessons for researchers and practitioners in the area of phishingdetection. This section is structured following the four aspects of phishing detection that have been discussed in the survey, namely: datasets, detection features, detection schemes, and evaluation metrics. According to our study, every technique (e.g., machine learning algorithms) used in the literature hasits own advantages and disadvantages, which has to be care- fully selected in order to optimize the goals of the detection approach.

A. *Dataset selection*

Datasets considerably affect the evaluation results. This evaluation criteria measures how easy it is to compile the datasets and to extract the necessary features from them. It also provides an understanding of the deployment environ- ment. Different environments may call for different detection techniques. For example, in healthy environments where the vast majority of the websites are legitimate, schemes with low FPR are preferable over those with high TPR and relatively high FPR. In section IV-A, we elaborate on the various important aspects relevant to datasets used in training and evaluating different phishing detection approaches.

One of the important challenges here is to compile a repre- sentative dataset that covers as much as possible of the behav-iors of phishing attackers. It is relatively easy to collect phish- ing related datasets from a single organization. However, suchdatasets may only offer a limited local view of the threats. On the other hand, compiling datasets from multiple organizationscould be challenging due to the potentially leakage of sensitiveinformation. Therefore, such efforts are usually hampered by restrictive legal clauses including non-disclosure agreements and regulations against sensitive information leakage. Even though when such representative datasets may be successfully compiled by some entities, it is generally difficult to share the datasets with the bigger phishing detection community. This not only adds to the difficulty of systematically comparing different phishing detection approaches, but also considerably limits the potential benefits of the datasets. The takeawaylesson here is that it is not always true that the bigger the dataset, the better the detection outcome, but rather, the more representative the dataset, the more comprehensive the features collected and the better the detection performance.

The timeliness of phishing detection datasets is anotherchallenging issue. As mentioned earlier, the cyber space is dynamic by nature, a characteristic that has been extensively exploited by attackers to evade detection. Attackers frequentlychange behavior and adjust their attack models to evade detec-tion. The most obvious examples include Dynamic GenerationAlgorithms (DGAs) and Fast Flux Service Networks (FFSNs).DGAs (e.g., [100]) create phishing websites, among others, that are only accessible for short time periods. This makes the identification of a phishing email, for example, by a known suspicious link that it contains useless, because such link may continuously change, some times on daily bases. FFSNs (e.g., [48]) exploit short TTL values to frequently change the IP addresses assigned to a specific domain, mainly to evade IP-based blocking. Such behavior makes, for example, IP- based phishing identification inefficient. Dynamic behavior ofattackers also complicates the process of dataset collectionas it limits the validity of such datasets to only short time periods. For example, a dataset of phishing URLs that hasbeen compiled 1 year ago, may not be valid now becausemost of the listed URLs

may no longer be in service.

One last issue is related to the ground truth datasets. It has been shown [99], [88] that blacklists, which are usually used to compile ground truth datasets, have high false positive and false negative rate. Such impurities may negatively impact the detection accuracy of the underlying phishing detection approaches. Therefore, we recommend to always cross-check the ground truth in multiple blacklists, or use majority voting to decide the inclusion of entries in the ground truth dataset.

B. *Feature selection and engineering*

Different phishing detection systems use different combi-nations of detection features. In this survey, we have dis-cussed the most important detection features used by different schemes. However, the literature lacks systematic evaluation of these features in terms of the availability of the detection features, the time it takes to mine the features, the complexity of extracting the features, and the robustness of the features. For example, a feature that can be easily manipulated by adversaries should not be used irrespective of its effectiveness in detecting current phishing attempts. This is intuitive as ad-versaries can simply manipulate the feature to avoid detection. Feature extraction (a.k.a., feature engineering) is a chal- lenging task that has a significant impact on the quality (accuracy and robustness) of the underlying phishing detection approaches. Well-crafted features lead to more successful detection approaches, while poor features may even ruin good detection algorithms. Typically, approaches look for features that maximize the detection accuracy while ignoring the robustness of the feature. Some features may have strong positive impact on the detection accuracy, however, they may be controlled by attackers and hence can be easily manipulated without affecting the attacker utility. For example, the URL- based lexical features (e.g., length of the URL, usage of HTTPS protocol, etc.) can be easily manipulated to evade detection. More specifically, it is no longer uncommon to see some phishing attackers increasingly use HTTPS, which makes such features less effective in identifying phishing attempts.

The most important takeaway here is that it is paramount for phishing detection approaches to carefully select the features that strike the right balance between detection accuracy and robustness in the face of potential manipulations. Our new *NRTT* feature is an example of a robust feature that not only provides excellent detection accuracy (as clearly shown by the experimental results), but also continuously adapt to changes in the underlying requirements. According to the work in [33], the design and measurement of *NRTT* aim at preventing attackers from being able to learn and mimic legitimate *NRTTs*, and hence the feature is robust. Additionally, the decision threshold specific to the feature is not fixed, but rather is adaptive and may change from the current measurement to the next one, depending on network conditions. More specifically, both the baseline *NRTT* and the *NRTT* of the claimed URL are measured and compared at the same time, and hence, congestion or other network conditions do not affect the outcome of the comparison.

However, it is quite challenging to evaluate the robustness of a feature in a systematic and measurable way. The importance of the problem has been recognized by many researchers in other domains as well (e.g. [123], [82], [118]). However, to the best of our knowledge, none of the existing approaches provide a framework that can be used to quantitatively evaluate the ro-bustness of features. Most of the approaches that recognize the problem only qualitatively discuss the robustness of some of the important features used in their approaches. Therefore, we believe that providing a framework that outlines qualitative and quantitative evaluation guidelines of the robustness of features is an open problem that calls for attention from the research community.

Such framework has to consider both complexity of feature forging as well as its impact on attacker benefits. Such framework will be an important tool in the face of the ever evolving attack as it helps researchers and practitioners todesign phishing detection techniques leveraging features that are both adaptive and hard to manipulate without considerablyaffecting attack utility.

One additional issue to consider while designing the detec- tion features of an approach is the time it takes to mine the feature. Some features could be extremely useful in identifying and detecting phishing attempts, however, they may take a relatively long time to compute, such as page reputation and virtual appearance similarity. The use of such features may either result in user inconvenience due to service delays until computation completes, or may result in security risks in case the service is provided before computing the features.

C. Detection schemes

As presented in Section IV-C, phishing detection systems use various data mining algorithms and detection approaches, each with its own advantages and disadvantages. Understand- ing the underlying data mining algorithms is important inevaluating the performance, the scalability and the robustness of phishing detection schemes.

When designing a phishing detection scheme, we recom- mend to follow the life cycle illustrated in Figure 5 in order to help fellow researchers obtain a comprehensive understanding of the proposed approach and to make it easy for future studies to conduct comparative evaluations. Specifically, a detection approach has to clearly state what it can and what it cannotdo in terms of phishing detection and blocking, to avoid relying on the approach in scenarios where it may not beefficient. Additionally, details of dataset specifications in termsof content and volume that better support the approach should be clearly articulated and documented.

A very important lesson that we have learned is that, dueto the dynamic nature of cyber attacks, the most reliable and efficient phishing detection approaches are those that can continuously adapt to cope with such dynamisms. [108] and
[118] are examples of such dynamic approaches. Additionally,the robustness of phishing detection approaches is tightly coupled with the robustness of the features used by the approach. Therefore, an approach may result in excellent detection accuracy at the time of design or in its early deployment, but fails miserably later due to either changes in the dataset or deliberate manipulation of the features utilized by the approach.

Two main categories have been considered in the underlyingtechnologies (e.g., feature mining, classification, etc.) of phish-ing detection approaches. The first category of approachesutilizes human expertise to identify phishing attempts, which has been implemented using various heuristics. However, suchapproaches require man-in-the-loop, and hence are too slow. They fail to handle large scale datasets and cannot cope with high data rates, frequent dataset changes, or adaptiveattack behaviors. Therefore, machine learning technologies, which utilize data-driven algorithms, were introduced to help automate the learning process. Different machine learning algorithms are used. Support Vector Machine (SVM), LogisticRegression (LR), and Bayesian-based classifiers, are among the mostly used algorithms in the literature.

Through our extensive investigation of the large body of phishing detection approaches, we learned that one size does not fit all and hence, it is extremely difficult to recommendone machine learning algorithm over another. Each machine learning algorithm has its own strengths and weaknesses, which has to be carefully considered to optimize the goalsof the detection approach. For example, SVM is considered among the most robust and accurate classification algorithms [117]. However, it has the drawback of being computationallyinefficient, and hence

may not be appropriate for large scale datasets or high data rates. On the other hand, LR is one ofthe most widely used statistical models for binary data [12]. However, it performs poorly when nonlinear relationships exist between feature sets. Furthermore, even though Bayesian- based classifiers are easy to construct and can be readily applied to large scale datasets [103], they assume independent features, and hence are very restrictive.

Recent research efforts leverage Deep Learning (DL) al- gorithms to improve the performance of phishing detection schemes. DL allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [102]. DL has been suc- cessfully applied in many research fields, such as speech recognition, visual object recognition, drug discovery and genomics. Therefore, we believe that DL could be a viable alternative to traditional machine learning algorithms (e.g.,SVM, LR), especially when handling complex and large scale datasets.

Another important issue that we have identified through thissurvey is the absence of deep and systematic evaluation ofthe performance of phishing detection approaches. The vast majority of the approaches focus on evaluating and analyzing the detection accuracy, while they overlook the run-time performance of the approach. Some approaches may showacceptable performance during the design and test phases due to the relatively small size datasets used during these phases. However, real world datasets are usually more complex and much larger, which may cause such approaches to perform poorly in real world applications. Systematic performance analysis can provide important guidelines to evaluate the scalability of detection approaches, which in turn can helpin improving performance by considering, for example, dis- tributed platforms and parallel algorithms.

D. Evaluation metrics.

In addition to evaluating the quality of the detection schemein terms of FPR and TPR, it is also imperative to have systematic evaluation of the effectiveness and the scalabilityof the underlying detection algorithms. Effectiveness metrics have been discussed in Section IV-D, however, we note thatmost of the work in the literature lacks evaluation of other performance aspects such as speed of detection, usability, and practical deployment, among others.

The majority of phishing detection approaches leverage ma- chine learning concepts including clustering and classificationtechniques. Therefore, they adopt the evaluation metrics and strategies developed in this domain. However, as mentioned earlier, the cyber security domain is more challenging dueto the adaptive nature of attackers. Therefore, the evaluation results during the design phase should be considered withcaution, as they may not hold later. In other words, the design phase results are limited in time validity and scope, which callsfor the phishing detection community to think about adaptive evaluation strategies that cope with the unique challenges in the cyber security domain. For example, the researchers could firstly classify the dataset into different categories (e.g., by type, time period, country, etc.), then perform the evaluation over every type of the dataset to obtain a more comprehensiveand convincing results. Another important issue is the dif- ficulty in providing comparative evaluations among different phishing detection techniques. This is mainly due to the lackof standard benchmarks, and the lack of reference datasets asa consequence of the dynamic nature of the attackers and the potential sensitivity of data, which restricts sharing.

Unfortunately, many of the above mentioned challengesacross all the aspects continue to exist, and hence call for a collaborative effort among the research community to alleviatetheir negative impact on the effectiveness and coverage of phishing detection approaches.

Table IX shows the comparison results across the previous four evaluation dimensions. From the performance perspective, we can see that 11 out of 12 schemes focus on the evaluation of true positive rate, false positive rate or other equivalent evaluation metrics. This is mainly because of the fact that TPR determines the detection capability of the scheme while FPR represents its negative effects. Thus, they together provide the most valuable performance information about the quality of different approaches.

PhishTank is the most dominant source for phishing web- sites because it provides large quantity, up-to-date and verified phishing list for free. Yahoo and DMOZ were competitors to each other for providing legitimate websites information. However, Yahoo closed its service since the end of 2014 for some unknown reasons. Other favorite sources for legitimate websites include Alexa top sites and Google keywords search-ing. Although researchers try to use a larger number of datasets for more convincing evaluation results, few of them considered some fundamental aspects about the datasets. For example, the ratio of the number of legitimate websites to the number of phishing websites, which is about 100 to 1 in reality.

Blacklists are commonly used in the public phishing de- tection toolbars because they have the fastest response time. From Table IX, we can conclude that the most commonly used features (also with the best performance results) are URL based and page content based features. Also, studies that are incorporated with more features tend to have better perfor- mance results. The recent trend is to leverage the classifier itself to optimize the detection accuracy using a large number of various detection features.

TABLE IX: EVALUATION OF ACADEMIC PHISHING DETECTION SCHEMES

			[38]	[124]	[40]	[72]	[73]	[87]	[108]	[118]	[102]	[90]	[27]	[75]
Performance	*Accuracy*						99%					99.62%		
	True Positive Rate		89%	97%	88%	92.4%		95%	97%	99%	90.87%	99.67%	100%	>99%
	False Positive Rate		0.71%	1%	0.7%	0.1%		3%	<0.1%	0.4%	0.87%	0.32%	0.74%	0.5%
	True Negative Rate											99.5%		
	False Negative Rate													
	Precision											0.5%		
	Recall													
	F1 score													
Data Set	*Data set source*	*Phishing*		PhishTank	Google	PhishTank Spamscatter	Webmail provider	PhishTank Spamscatter	Gmail User submission	Phishtank	Google SURBL APWG PhishTank	Phishtank Others	Phishtank	PhishTank
		Legitimate	Google	3Sharp		DMOZ Yahoo	Yahoo	DMOZ Yahoo	Gmail submission	Alexa Yahoo 3Sharp		Alexa Google Others	Alexa	Intel Security
	Data set size	*Phishing*	9	100	Several million	20000	6000 7500	32000	16967	6943	500000	3374	1000	1553
		Legitimate	10272	100		15000	12500 to 14000	120000	74816740	2561	500000	1200	404	100000
	Data set redundancy									Reduced				
	Data set timeliness		2005	Nov.17th to 18th 2006	Aug.20th to 31st 2006	Aug.22nd to Sept.1st 2008	2009	2009	2009	2009	2011	2014	Jun. 4th to 22nd Jul. 13th to 25th 2012	2015 to 2016
	Legitimate sites to Phishing sites ratio		1141:1	1:1		4:3	2:1	15:4	4410:1	1:2.7	1:1	1:2.8	2.48:1	64.4:1
	Training set to testing set ratio		1:100			1:1			6:1	3:7	4:1			4:1
Feature	*Blacklist*				Y			Y						
	Heuristics: URL lexical			Y	Y	Y	Y	Y	Y	Y	Y			Y
	Heuristics: URL host			Y	Y	Y	Y	Y	Y	Y	Y			
	Heuristics: Page content			Y	Y				Y	Y	Y	Y	Y	Y
	Heuristics: Visual similarity		Y											
	Heuristics: Others										Y			
Techniques	*SVM*					Y	Y			Y				
	LR				Y	Y			Y	Y	Y		Y	
	Bayesian-based					Y				Y				
	EMD		Y											
	DBSCAN													
	TD-IDF			Y						Y		Y		
	Fuzzy-based													
	CW						Y							
	NN													
	RF													
	J48									Y				
	AdaBoost									Y				
	Perceptron						Y							
	AROW													
	LDA													
	SMO													
	C4.5													
	RT													
	LMT													
	Gradient Boosting													Y
	Jrip													

REFERENCES

[1] 2016 phishing trends & intelligence report: Hacking the human. https: //info.phishlabs.com/pti-report-download.

[2] The alexa top 500 sites on the web. http://www.dmoz.org/.

[3] Anti-phishing extension: Netcraft. http://toolbar.netcraft.com/.

[4] Anti-Phishing Working Group. http:/www.antiphishing.org/.

[5] Clean MX malicious URL list. http://support.clean-mx.com/clean-mx/ phishing.php?response=alive.

[6] DMOZ - the directory of the web. http://www.dmoz.org/.

[7] Malwarepatrol. https://www.malwarepatrol.net/open-source.shtml.

[8] Millersmiles spoof email and phishing scams list. http://www. millersmiles.co.uk/scams.php.

[9] SURBL URL reputation data. http://www.surbl.org/lists.

[10] Greg Aaron and Rod Rasmussen. Global phishing survey: trends and domain name use in 2h2009. *Anti-Phishing Working Group (APWG), Lexington, MA*, 2010.

[11] Saeed Abu-Nimeh and Suku Nair. Bypassing security toolbars and phishing filters via dns poisoning. In *IEEE GLOBECOM 2008-2008 IEEE Global Telecommunications Conference*, pages 1–6. IEEE, 2008.

[12] Saeed Abu-Nimeh, Dario Nappa, Xinlei Wang, and Suku Nair. A comparison of machine learning techniques for phishing detection. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 60–69. ACM, 2007.

[13] Maher Aburrous, M Alamgir Hossain, Keshav Dahal, and Fadi Thabtah. Intelligent phishing detection system for e-banking using fuzzy data mining. *Expert systems with applications*, 37(12):7913–7921, 2010.

[14] Maher Aburrous and Adel Khelifi. Phishing detection plug-in tool- bar using intelligent fuzzy-classification mining techniques. In *The international conference on soft computing and software engineering [SCSE13], San Francisco State University, San Francisco, California, USA*, 2013.

[15] Sadia Afroz and Rachel Greenstadt. Phishzoo: An automated web phishing detection approach based on profiling and fuzzy matching. In *Proceedings of the Semantic Computing (ICSC), 2011 Fifth IEEE International Conference*, 2009.

[16] Anupama Aggarwal, Ashwin Rajadesingan, and Ponnurangam Ku- maraguru. Phishari: Automatic realtime phishing detection on twitter. In *eCrime Researchers Summit (eCrime), 2012*, pages 1–12. IEEE, 2012.

[17] Fadi A Aloul, Syed Zahidi, and Wassim El-Hajj. Two factor authenti- cation using mobile phones. In *AICCSA*, pages 641–644, 2009.

[18] David S Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M Voelker. Spamscatter: Characterizing internet scam hosting infrastruc- ture. In *Usenix Security*, pages 1–14, 2007.

[19] Anti-Phishing Working Group et al. APWG Phishing trends reports, 2010 - 2016.

[20] Mustafa Aydin and Nazife Baykal. Feature extraction and classification phishing websites based on url. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 769–770. IEEE, 2015.

[21] Andre´ Bergholz, Jan De Beer, Sebastian Glahn, Marie-Francine Moens, Gerhard Paaß, and Siehyun Strobel. New filtering approaches for phishing email. *Journal of computer security*, 18(1):7–35, 2010.

[22] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. Lexical feature based phishing url detection using online learning. In *Proceedings of the 3rd ACM workshop on Artificial intelligence and security*, pages 54–60. ACM, 2010.

[23] Ye Cao, Weili Han, and Yueran Le. Anti-phishing based on automated individual white-list. In *Proceedings of the 4th ACM workshop on Digital identity management*, pages 51–60. ACM, 2008.

[24] Deanna D Caputo, Shari Lawrence Pfleeger, Jesse D Freeman, and M Eric Johnson. Going spear phishing: Exploring embedded training and awareness. *IEEE Security & Privacy*, 12(1):28–38, 2014.

[25] Kuan-Ta Chen, Jau-Yuan Chen, Chun-Rong Huang, and Chu-Song Chen. Fighting phishing with discriminative keypoint features. *IEEE Internet Computing*, 13(3):56–63, 2009.

[26] Teh-Chung Chen, Scott Dick, and James Miller. Detecting visually similar web pages: Application to phishing detection. *ACM Transac- tions on Internet Technology (TOIT)*, 10(2):5, 2010.

[27] Teh-Chung Chen, Torin Stepan, Scott Dick, and James Miller. An anti-phishing system employing diffused information. *ACM Transactions on Information and System Security (TISSEC)*, 16(4):16, 2014.

[28] Neil Chou, Robert Ledesma, Yuka Teraguchi, John C Mitchell, et al. Client-side defense against web-based identity theft. In *NDSS*, 2004.

[29] Giovanni Comarela, Gonca Gürsun, and Mark Crovella. Studying interdomain routing over long timescales. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 227–234. ACM, 2013.

[30] Lorrie Faith Cranor, Serge Egelman, Jason I Hong, and Yue Zhang. Phinding phish: An evaluation of anti-phishing toolbars. In *NDSS*, 2007.

[31] Rachna Dhamija, J Doug Tygar, and Marti Hearst. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590. ACM, 2006.

[32] Zuochao Dou, Issa Khalil, and Abdallah Khreishah. CLAS: A novel communications latency based authentication scheme. *Security and Communication Networks*, 2017.

[33] Zuochao Dou, Issa Khalil, and Abdallah Khreishah. A novel and robust authentication factor based on network communications latency. *IEEE Systems Journal*, 2017.

[34] Zuochao Dou, Issa Khalil, Abdallah Khreishah, and Ala Al-Fuqaha. Robust insider attacks countermeasure for hadoop: Design and imple- mentation. *IEEE Systems Journal*, 2017.

[35] Julie S Downs, Mandy B Holbrook, and Lorrie Faith Cranor. Decision strategies and susceptibility to phishing. In *Proceedings of the second symposium on Usable privacy and security*, pages 79–90. ACM, 2006.

[36] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1065–1074. ACM, 2008.

[37] Mohammed Nazim Feroz and Susan Mengel. Examination of data, rule generation and detection of phishing urls using online logistic re- gression. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 241–250. IEEE, 2014.

[38] Anthony Y Fu, Liu Wenyin, and Xiaotie Deng. Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). *IEEE transactions on dependable and secure computing*, 3(4):301–311, 2006.

[39] AY Fu, W Liu, and Xiaotie Deng. EMD based visual similarity for detection of phishing webpages. In *Proceedings of International Workshop on Web Document Analysis*, volume 2005, 2005.

[40] Sujata Garera, Niels Provos, Monica Chew, and Aviel D Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the 2007 ACM workshop on Recurring malcode*, pages 1–8. ACM, 2007.

[41] Sophie Gastellier-Prevost, Gustavo Gonzalez Granadillo, and Maryline Laurent. A dual approach to detect pharming attacks at the client-side. In *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*, pages 1–5. IEEE, 2011.

[42] GeoTrust. Geotrust TrustWatch Toolbar. https://www.geotrust.com/ comcasttoolbar/.

[43] Ammar Gharaibeh, Mohammad A Salahuddin, Sayed J Hussini, Ab- dallah Khreishah, Issa Khalil, Mohsen Guizani, and Ala Al-Fuqaha. Smart cities: A survey on data management, security and enabling technologies. *IEEE Communications Surveys & Tutorials*, 2017.

[44] Anti-Phishing Working Group et al. Apwg. *URL http://www. antiphish-ing. org*, 2016.

[45] Xiao Han, Nizar Kheir, and Davide Balzarotti. Phisheye: Live moni- toring of sandboxed phishing kits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 1402–1413. ACM, 2016.

[46] Masanori Hara, Akira Yamada, and Yutaka Miyake. Visual similarity- based phishing detection without victim site information. In *Computa- tional Intelligence in Cyber Security, 2009. CICS'09. IEEE Symposium on*, pages 30–36. IEEE, 2009.

[47] Frank L Hitchcock. The distribution of a product from several sources to numerous localities. *Journal of mathematics and physics*, 20(1):224–230, 1941.

[48] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and Detecting Fast-Flux Service Networks. In *Proceedings of the 15th Network and Distributed System Security*

Symposium, 2008.

[49] Jason Hong. The state of phishing attacks. *Communications of the ACM*, 55(1):74–81, 2012.

[50] Google Inc. Google Safe Browsing. https://developers.google.com/ safe-browsing/.

[51] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. Social phishing. *Communications of the ACM*, 50(10):94– 100, 2007.

[52] Chris Karlof, Umesh Shankar, J Doug Tygar, and David Wagner. Dynamic pharming attacks and locked same-origin policies for web browsers. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 58–71. ACM, 2007.

[53] Issa Khalil and Saurabh Bagchi. Secos: key management for scalable and energy efficient crypto on sensors. *Proceedings of IEEE Depend- able Systems and Networks (DSN)*, 2003.

[54] Issa Khalil, Saurabh Bagchi, and Ness Shroff. Analysis and evaluation of secos, a protocol for energy efficient and secure communication in sensor networks. *Ad Hoc Networks*, 5(3):360–391, 2007.

[55] Issa Khalil, Zuochao Dou, and Abdallah Khreishah. TPM-based authentication mechanism for apache hadoop. In *International Confer- ence on Security and Privacy in Communication Systems*, pages 105–
122. Springer, 2014.

[56] Issa Khalil, Zuochao Dou, and Abdallah Khreishah. Your credentials are compromised, do not panic: You can be well protected. In *11th ACM AsiaCCS*, 2016.

[57] Issa Khalil, Ismail Hababeh, and Abdallah Khreishah. Secure inter cloud data migration. In *Information and Communication Systems (ICICS), 2016 7th International Conference on*, pages 62–67. IEEE, 2016.

[58] Issa Khalil, Ting Yu, and Bei Guan. Discovering malicious domains through passive dns data graph analysis. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 663–674. ACM, 2016.

[59] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. Phishing detec- tion: a literature survey. *IEEE Communications Surveys & Tutorials*, 15(4):2091–2121, 2013.

[60] Ponnurangam Kumaraguru, Yong Rhee, Steve Sheng, Sharique Hasan, Alessandro Acquisti, Lorrie Faith Cranor, and Jason Hong. Getting users to pay attention to anti-phishing education: evaluation of retention and transfer. In *Proceedings of the anti-phishing working groups 2nd annual eCrime researchers summit*, pages 70–81. ACM, 2007.

[61] Ponnurangam Kumaraguru, Steve Sheng, Alessandro Acquisti, Lor- rie Faith Cranor, and Jason Hong. Teaching johnny not to fall for phish. *ACM Transactions on Internet Technology (TOIT)*, 10(2):7, 2010.

[62] Minseok Kwon, Zuochao Dou, Wendi Heinzelman, Tolga Soyata, He Ba, and Jiye Shi. Use of network latency profiling and redundancy for cloud server selection. In *IEEE 7th International Conference on Cloud Computing, 2014*.

[63] Stanford Security Lab. Spoofguard. https://crypto.stanford.edu/ SpoofGuard/.

[64] Craig Labovitz, G Robert Malan, and Farnam Jahanian. Internet routing instability. *Networking, IEEE/ACM Transactions on*, 1998.

[65] Mohit Lad, Jong Han Park, Tiziana Refice, and Lixia Zhang. A study of internet routing stability using link weight. Technical report, 2008.

[66] Anh Le, Athina Markopoulou, and Michalis Faloutsos. Phishdef: Url names say it all. In *INFOCOM, 2011 Proceedings IEEE*, pages 191–
195. IEEE, 2011.

[67] Christophe Leys, Christophe Ley, Olivier Klein, Philippe Bernard, and Laurent Licata. Detecting outliers: do not use standard deviation around the mean, use absolute deviation around the median. *JESP*, 2013.

[68] Gastón L'Huillier, Alejandro Hevia, Richard Weber, and Sebastían R´ıos. Latent semantic analysis and keyword extraction for phishing classification. In *Intelligence and Security Informatics (ISI), 2010 IEEE International Conference on*, pages 129–131. IEEE, 2010.

[69] Peter Likarish, Eunjin Jung, Don Dunbar, Thomas E Hansen, and Juan Pablo Hourcade. B-apt: Bayesian anti-phishing toolbar. In *2008 IEEE International Conference on Communications*, pages 1745–1749. IEEE, 2008.

[70] Gang Liu, Bite Qiu, and Liu Wenyin. Automatic detection of phishing target from phishing webpage. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 4153–4156. IEEE, 2010.

[71] ZQ John Lu. The elements of statistical learning: data mining, inference, and prediction. 2010.

[72] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Beyond blacklists: learning to detect malicious web sites from suspi- cious urls. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1245– 1254. ACM, 2009.

[73] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Identifying suspicious urls: an application of large-scale online learn- ing. In *Proceedings of the 26th annual international conference on machine learning*, pages 681–688. ACM, 2009.

[74] Samuel Marchal, Jérôme Françoi s, Radu State, and Thomas Engel. Phishstorm: Detecting phishing with streaming analytics. *IEEE Trans- actions on Network and Service Management*, 11(4):458–471, 2014.

[75] Samuel Marchal, Kalle Saari, Nidhi Singh, and N Asokan. Know your phish: Novel techniques for detecting phishing sites and their targets. *arXiv preprint arXiv:1510.06501*, 2015.

[76] Max-Emanuel Maurer and Dennis Herzner. Using visual website similarity for phishing detection and reporting. In *CHI'12 Extended Abstracts on Human Factors in Computing Systems*, pages 1625–1630. ACM, 2012.

[77] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual-similarity-based phishing detection. In *Proceedings of the 4th international con- ference on Security and privacy in communication netowrks*, page 22. ACM, 2008.

[78] Ioan-Cosmin Mihai and Laurentiu Giurea. Management of elearning platforms security. In *The International Scientific Conference eLearn- ing and Software for Education*, volume 1, page 422. " Carol I" National Defence University, 2016.

[79] Mahmood Moghimi and Ali Yazdian Varjani. New rule-based phishing detection method. *Expert systems with applications*, 53:231–242, 2016.

[80] Rami M Mohammad, Fadi Thabtah, and Lee McCluskey. Predicting phishing websites based on self-structuring neural network. *Neural Computing and Applications*, 25(2):443–458, 2014.

[81] LLC OpenDNS. Phishtank: an anti-phishing site.

[82] A. Oprea, Z. Li, T. F. Yen, S. H. Chin, and S. Alrwais. Detection of Early-Stage Enterprise Infection by Mining Large-Scale Log Data. In *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 45–56, June 2015.

[83] Paul Pajares. Phishing Safety: Is HTTPS Enough? http://blog.trendmicro.com/trendlabs-security-intelligence/
phishing-safety-is-https-enough/.

[84] Ying Pan and Xuhua Ding. Anomaly based web phishing page detection. In *Acsac*, volume 6, pages 381–392, 2006.

[85] Rajesh Krishna Panta, Saurabh Bagchi, and Issa M Khalil. Efficient wireless reprogramming through reduced bandwidth usage and oppor- tunistic sleeping. *Ad Hoc Networks*, 7(1):42–62, 2009.

[86] Bimal Parmar. Protecting against spear-phishing. *Computer Fraud & Security*, 2012(1):8–11, 2012.

[87] Pawan Prakash, Manish Kumar, Ramana Rao Kompella, and Minaxi Gupta. Phishnet: predictive blacklisting to detect phishing attacks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[88] A. Ramachandran, D. Dagon, and Nick Feamster. Can DNS-based Blacklists Keep Up with Bots. In *Proceedings of the 3rd Conference on Email and Anti-Spam*, 2006.

[89] Venkatesh Ramanathan and Harry Wechsler. Phishing website detection using latent dirichlet allocation and adaboost. In *Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on*, pages 102–107. IEEE, 2012.

[90] Gowtham Ramesh, Ilango Krishnamurthi, and K Sampath Sree Kumar. An efficacious method for detecting phishing webpages through target domain identification. *Decision Support Systems*, 61:12–22, 2014.

[91] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang. BGP routing stability of popular destinations. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002.

[92] Paul Robichaux and Devin L Ganger. Gone phishing: Evaluating anti- phishing tools for windows. *3Sharp Project Report, Sept*, 2006.

[93] John C Russ and Roger P Woods. The image processing handbook. *Journal of Computer Assisted Tomography*, 19(6):979–981, 1995.

[94] Nuttapong Sanglerdsinlapachai and Arnon Rungsawang. Using domain top-page similarity feature in machine learning-based web phishing detection. In *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*, pages 187–190. IEEE, 2010.

[95] Aman Shaikh, Anujan Varma, Lampros Kalampoukas, and Rohit Dube. Routing stability in congested networks: Experimentation and analysis. In *ACM SIGCOMM Computer Communication Review*, 2000.

[96] Mohsen Sharifi and Seyed Hossein Siadati. A phishing sites blacklist generator. In *2008 IEEE/ACS International Conference on Computer Systems and Applications*, pages 840–843. IEEE, 2008.

[97] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lor- rie Faith Cranor, and Julie Downs. Who falls for phish?: a demographic analysis of phishing susceptibility and effectiveness of interventions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 373–382. ACM, 2010.

[98] Steve Sheng, Bryant Magnien, Ponnurangam Kumaraguru, Alessandro Acquisti, Lorrie Faith Cranor, Jason Hong, and Elizabeth Nunge. Anti- phishing phil: the design and evaluation of a game that teaches people not to fall for phish. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 88–99. ACM, 2007.

[99] S. Sinha, M. Bailey, and F. Jahanian. Shades of Grey: On the Effectiveness of Reputation-based "Blacklists". In *Proceedings of the 3rd International Conference on Malicious and Unwanted Software*, pages 57–64, Oct 2008.

[100] A. K. Sood and S. Zeadally. A Taxonomy of Domain-Generation Algorithms. *IEEE Security Privacy*, 14(4):46–53, 2016.

[101] Choon Lin Tan, Kang Leng Chiew, KokSheik Wong, et al. PhishWHO: Phishing webpage detection via identity keywords extraction and target domain name finder. *Decision Support Systems*, 88:18–27, 2016.

[102] Kurt Thomas, Chris Grier, Justin Ma, Vern Paxson, and Dawn Song. Design and evaluation of a real-time url spam filtering service. In *2011 IEEE Symposium on Security and Privacy*, pages 447–462. IEEE, 2011.

[103] Gaurav Varshney, Manoj Misra, and Pradeep K Atrey. A survey and classification of web phishing detection schemes. *Security and Communication Networks*, 2016.

[104] Jingguo Wang, Tejaswini Herath, Rui Chen, Arun Vishwanath, and H Raghav Rao. Research article phishing susceptibility: An investi- gation into the processing of a targeted spear phishing email. *IEEE transactions on professional communication*, 55(4):345–362, 2012.

[105] D Yu Weider, Shruti Nargundkar, and Nagapriya Tiruthani. A phishing vulnerability analysis of web based systems. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, pages 326–
331. IEEE, 2008.

[106] Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Xiaotie Deng, and Zhang Min. Phishing web page detection. In *Eighth International Conference on Document Analysis and Recognition (ICDAR'05)*, pages 560–564. IEEE, 2005.

[107] Liu Wenyin, Gang Liu, Bite Qiu, and Xiaojun Quan. Antiphishing through phishing target discovery. *IEEE Internet Computing*, 16(2):52–61, 2012.

[108] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *NDSS*, volume 10, 2010.

[109] Wikipedia. Avalanche (phishing group) — wikipedia, the free ency- clopedia, 2016.

[110] Wikipedia. Bag-of-words model — wikipedia, the free encyclopedia, 2016.

[111] Wikipedia. Mcafee siteadvisor — wikipedia, the free encyclopedia, 2016. [Online; accessed 6-September-2016].

[112] Wikipedia. Microsoft smartscreen — wikipedia, the free encyclopedia, 2016. [Online; accessed 28-September-2016].

[113] Wikipedia. Netcraft — wikipedia, the free encyclopedia, 2016. [Online; accessed 3-September-2016].
[114] Wikipedia. Yahoo! directory — wikipedia, the free encyclopedia, 2016. [Online; accessed 7-June-2016].
[115] Wikipedia. Zero-day (computing) — wikipedia, the free encyclopedia, 2016.
[116] Min Wu, Robert C Miller, and Simson L Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 601–610. ACM, 2006.
[117] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.
[118] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):21, 2011.
[119] Guang Xiang and Jason I Hong. A hybrid phish detection approach by identity discovery and keywords retrieval. In *Proceedings of the 18th international conference on World wide web*, pages 571–580. ACM, 2009.
[120] John Yearwood, Musa Mammadov, and Arunava Banerjee. Profiling phishing emails based on hyperlink information. In *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*, pages 120–127. IEEE, 2010.
[121] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. *HotCloud*, 10:10–10, 2010.
[122] Haijun Zhang, Gang Liu, Tommy WS Chow, and Wenyin Liu. Textual and visual content-based anti-phishing: a bayesian approach. *Neural Networks, IEEE Transactions on*, 22(10):1532–1546, 2011.
[123] Jialong Zhang, Sabyasachi Saha, Guofei Gu, Sung-Ju Lee, and Marco Mellia. Systematic mining of associated server herds for malware campaign discovery. In *Proceedings of the 35th IEEE International Conference on Distributed Computing Systems*, pages 630–641, 2015.
[124] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content- based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*, pages 639–648. ACM, 2007.
[125] Anti-phishing Working Group (APWG) Phishing Activity Trends Report 4th quarter 2020, https://docs.apwg.org/reports/apwg trends report q4 2020.pdf
[126] FBI Internet Crime Report 2020, https://www.ic3.gov/Media/PDF/AnnualReport/2020 IC3Report.pdf
[127] Verizon 2020 Data Breach Investigation Report, https://enterprise.verizon.com/resources/reports/2020-data- breachinvestigations-report.pdf
[128] World Health Organization, Communicating for Health, Cyber Security, https://www.who.int/about/communications/cyber-security
[129] Ye Cao, Weili Han, and Yueran Le, "Anti-phishing based on automated individual white-list," Proceedings of the 4th ACM workshop on Digital identity management-DIM 08, pp. 51-60, 2008
[130] M. Sharifi, and S. H. Siadati, "A phishing sites blacklist generator," 2008 IEEE/ACS International Conference on Computer Systems and Applications, pp. 840-843, 2008
[131] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based associative classification data mining," Expert Systems with Applications, vol. 41, no.13, pp. 5948-5959, 2014
[132] L. Wenyin, G. Huang, L. Xiaoyue, Z. Min, and X. Deng, "Detection of phishing webpages based on visual similarity," Special interest tracks and posters of the 14th international conference on World Wide Web-
[133] WWW 05, pp. 1060-1061, 2005
[134] C. L. Tan, K. L. Chiew et al., "Phishing website detection using url assisted brand name weighting system," 2014 International Symposium on Intelligent Signal Processing and Communication Systems(ISPACS), IEEE, pp. 054-059, 2014
[135] K. L. Chiew, E. H. Chang, W. K. Tiong et al., "Utilisation of website logo for phishing

detection," Computers & Security, vol. 54, pp. 16-26, 2015

[136] K. M. kumar, K. Alekhya, "Detecting phishing websites using fuzzy logic," International Journal of Advanced Research in Computer Engineering Technology(IJARCET), vol. 5, no. 10, 2016

[137] Rishikesh Mahajan, and Irfan Siddavatam, "Phishing website detection using machine learning algorithms," International Journal of Computer Applications(0975-8887), vol. 181, no. 23, 2018

[138] Jitendra Kumar, A. Santhanavijayan, B. Janet, Balaji Rajendran, and Bindhumadhava BS, "Phishing website classification and detection using machine learning," International Conference on Computer Communication and Informatics(ICCCI), 2020

[139] Mehmet Korkmaz, Ozgur Koray Sahingoz, Banu Diri, "Detection of phishing websites by using machine learning-based URL analysis," 11nth International Conference on Computing, Communication and Networking Technologies(ICCCNT), 2020

[140] Mohammad Nazmul Alam, Dhiman Sarma et al., "Phishing attacks detection using machine learning approach," 3rd International Conference on Smart Systems and Inventive Technology(ICSSIT), 2020

[141] Abdulhamit Subasi, Esraa Molah, Fatin Almkallawi, Touseef J. Chaudhery, "Intelligent phishing website detection using Random Forest classifier," International Conference on Electrical and Computing Technologies and Applications(ICECTA), 2017

[142] Structure of a URL – image, https://towardsdatascience.com/phishingdomain-detection-with-ml-5be9c99293e5

[143] Rami M. Mohammad, Fadi Thabtah, Lee McCluskey, "Phishing websites features

[144] J. Shad and S. Sharma, "A Novel Machine Learning Approach to Detect Phishing Websites Jaypee Institute of Information Technology," pp. 425–430, 2018.

[145] Y. Sönmez, T. Tuncer, H. Gökal, and E. Avci, "Phishing web sites features classification based on extreme learning machine," 6th Int. Symp. Digit. Forensic Secur. ISDFS 2018 - Proceeding, vol. 2018– Janua, pp. 1–5, 2018.

[146] T. Peng, I. Harris, and Y. Sawa, "Detecting Phishing Attacks Using Natural Language Processing and Machine Learning," Proc. - 12th IEEE Int. Conf. Semant. Comput. ICSC 2018, vol. 2018–Janua, pp. 300–301, 2018.

[147] M. Karabatak and T. Mustafa, "Performance comparison of classifiers on reduced phishing website dataset," 6th Int. Symp. Digit. Forensic Secur. ISDFS 2018 - Proceeding, vol. 2018–Janua, pp. 1–5, 2018.

[148] S. Parekh, D. Parikh, S. Kotak, and P. S. Sankhe, "A New Method for Detection of Phishing Websites: URL Detection," in 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), 2018, vol. 0, no. Icicct, pp. 949–952.

[149] K. Shima et al., "Classification of URL bitstreams using bag of bytes," in 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2018, vol. 91, pp. 1–5.

[150] W. Fadheel, M. Abusharkh, and I. Abdel-Qader, "On Feature Selection for the Prediction of Phishing Websites," 2017 IEEE 15th Intl Conf Dependable, Auton. Secur. Comput. 15th Intl Conf Pervasive Intell. Comput. 3rd Intl Conf Big Data Intell. Comput. Cyber Sci. Technol. Congr., pp. 871–876, 2017.

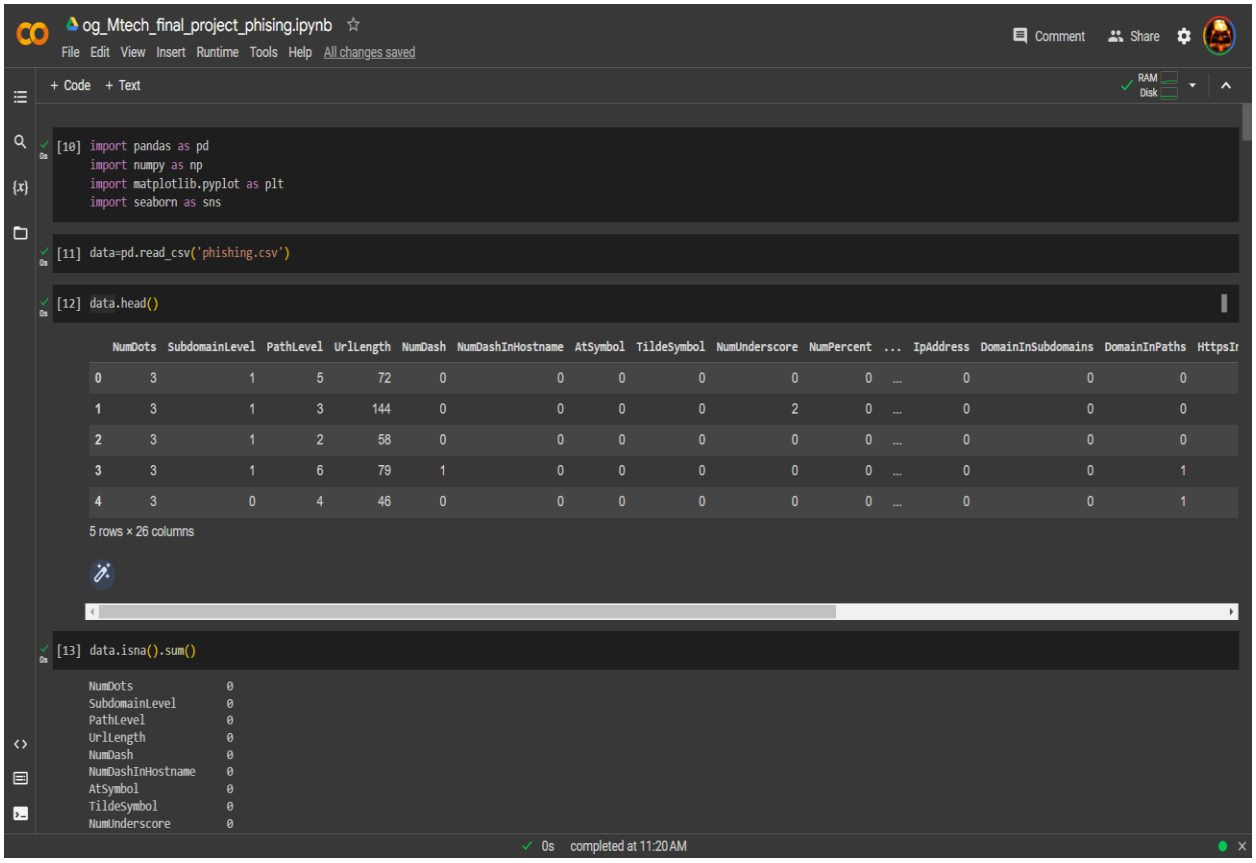
[151] X. Zhang, Y. Zeng, X. Jin, Z. Yan, and G. Geng, "Boosting the Phishing Detection Performance by Semantic Analysis," 2017.

[152] L. MacHado and J. Gadge, "Phishing Sites Detection Based on C4.5 Decision Tree Algorithm," in 2017 International Conference on Computing, Communication, Control and Automation, ICCUBEA 2017, 2018, pp. 1–5.

[153] A. Desai, J. Jatakia, R. Naik, and N. Raul, "Malicious web content detection using machine leaning," RTEICT 2017 - 2nd IEEE Int. Conf. Recent Trends Electron. Inf. Commun. Technol. Proc., vol. 2018–Janua, pp. 1432–1436, 2018.

[154] R. S. Rao and S. T. Ali, ”PhishShield: A Desktop Application to Detect Phishing Webpages through Heuristic Approach,” Procedia Computer Science, vol. 54, no. Supplement C, pp. 147-156, 2015.
[155] 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.
[156] Hassan Y.A.Abutair, AbdelfettahBelghith, “Using Case-Based Reason- ing for Phishing Detection,” in 2017 .The 8th International Conference on Ambient SystemsNetworks and Technologies(ANT)2017,pp.281- 288.
[157] L. Breiman, ”Random Forests,” Machine Learning, vol.45, no. 1, pp.5-32, Oct. 2001.
[158] Sirageldin, B. B. Baharudin, and L. T. Jung, ”Malicious Web Page Detection: A Machine Learning Approach,” in Advances in Computer Science and its Applications, Springer, Berlin, Heidelberg, 2014, pp.217- 22

SAMPLE CODE



```
[10] import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns

[11] data=pd.read_csv('phishing.csv')

[12] data.head()
```

	NumDots	SubdomainLevel	PathLevel	UrlLength	NumDash	NumDashInHostname	AtSymbol	TildeSymbol	NumUnderscore	NumPercent	...	IpAddress	DomainInSubdomains	DomainInPaths	HttpsIr
0	3	1	5	72	0	0	0	0	0	0	...	0	0	0	
1	3	1	3	144	0	0	0	0	2	0	...	0	0	0	
2	3	1	2	58	0	0	0	0	0	0	...	0	0	0	
3	3	1	6	79	1	0	0	0	0	0	...	0	0	1	
4	3	0	4	46	0	0	0	0	0	0	...	0	0	1	

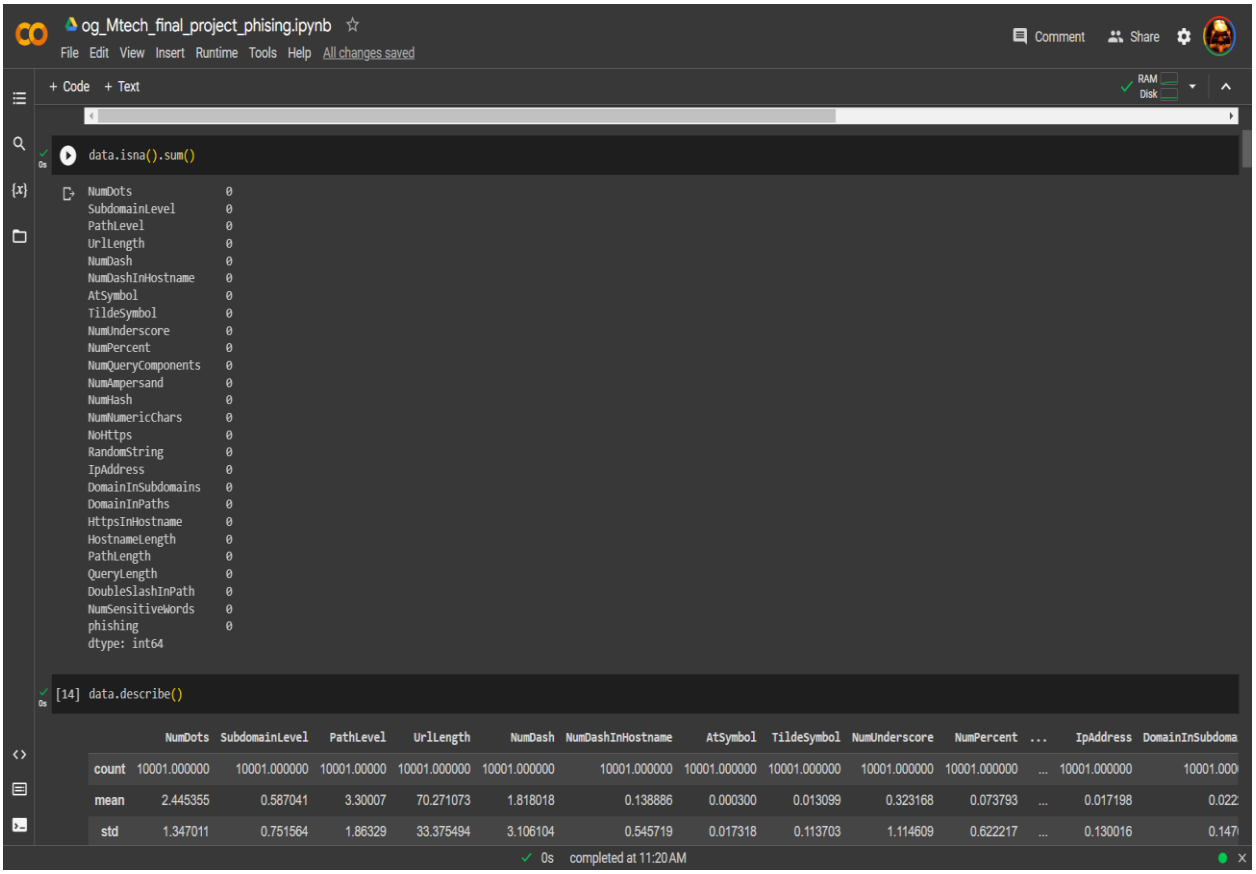
5 rows × 26 columns

```
[13] data.isna().sum()

NumDots             0
SubdomainLevel      0
PathLevel           0
UrlLength           0
NumDash             0
NumDashInHostname   0
AtSymbol            0
TildeSymbol         0
NumUnderscore       0
```

0s completed at 11:20 AM

Fig 1 : Sample Code screenshot 1



```
data.isna().sum()

NumDots             0
SubdomainLevel      0
PathLevel           0
UrlLength           0
NumDash             0
NumDashInHostname   0
AtSymbol            0
TildeSymbol         0
NumUnderscore       0
NumPercent          0
NumQueryComponents  0
NumAmpersand        0
NumHash             0
NumNumericChars     0
NoHttps             0
RandomString        0
IpAddress           0
DomainInSubdomains  0
DomainInPaths       0
HttpsInHostname     0
HostnameLength      0
PathLength          0
QueryLength         0
DoubleSlashInPath   0
NumSensitiveWords   0
phishing            0
dtype: int64
```

```
[14] data.describe()
```

	NumDots	SubdomainLevel	PathLevel	UrlLength	NumDash	NumDashInHostname	AtSymbol	TildeSymbol	NumUnderscore	NumPercent	...	IpAddress	DomainInSubdoma
count	10001.000000	10001.000000	10001.00000	10001.000000	10001.000000	10001.000000	10001.000000	10001.000000	10001.000000	10001.000000	...	10001.000000	10001.000
mean	2.445355	0.587041	3.30007	70.271073	1.818018	0.138886	0.000300	0.013099	0.323168	0.073793	...	0.017198	0.022
std	1.347011	0.751564	1.86329	33.375494	3.106104	0.545719	0.017318	0.113703	1.114609	0.622217	...	0.130016	0.147

0s completed at 11:20 AM

Fig 2 : Sample Code screenshot 2

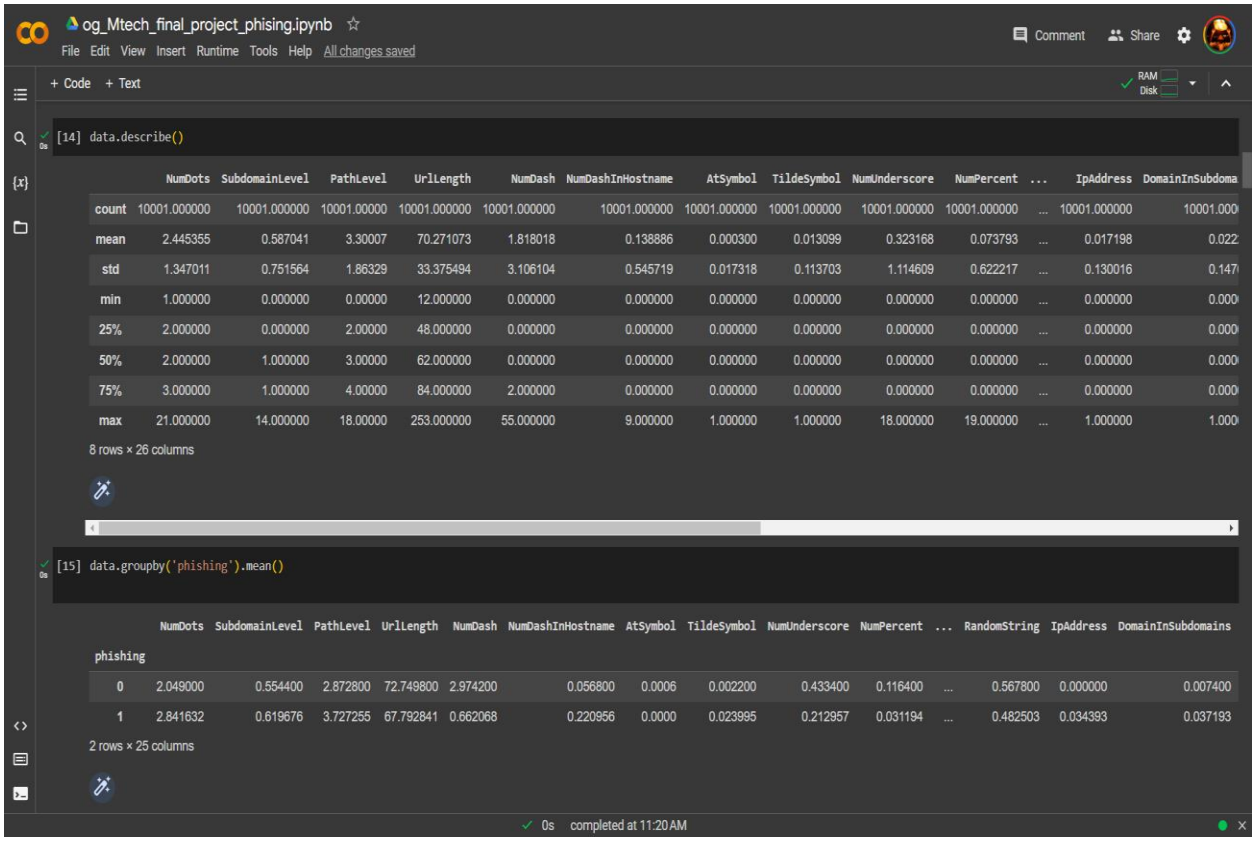


Fig 3 : Sample Code screenshot 3

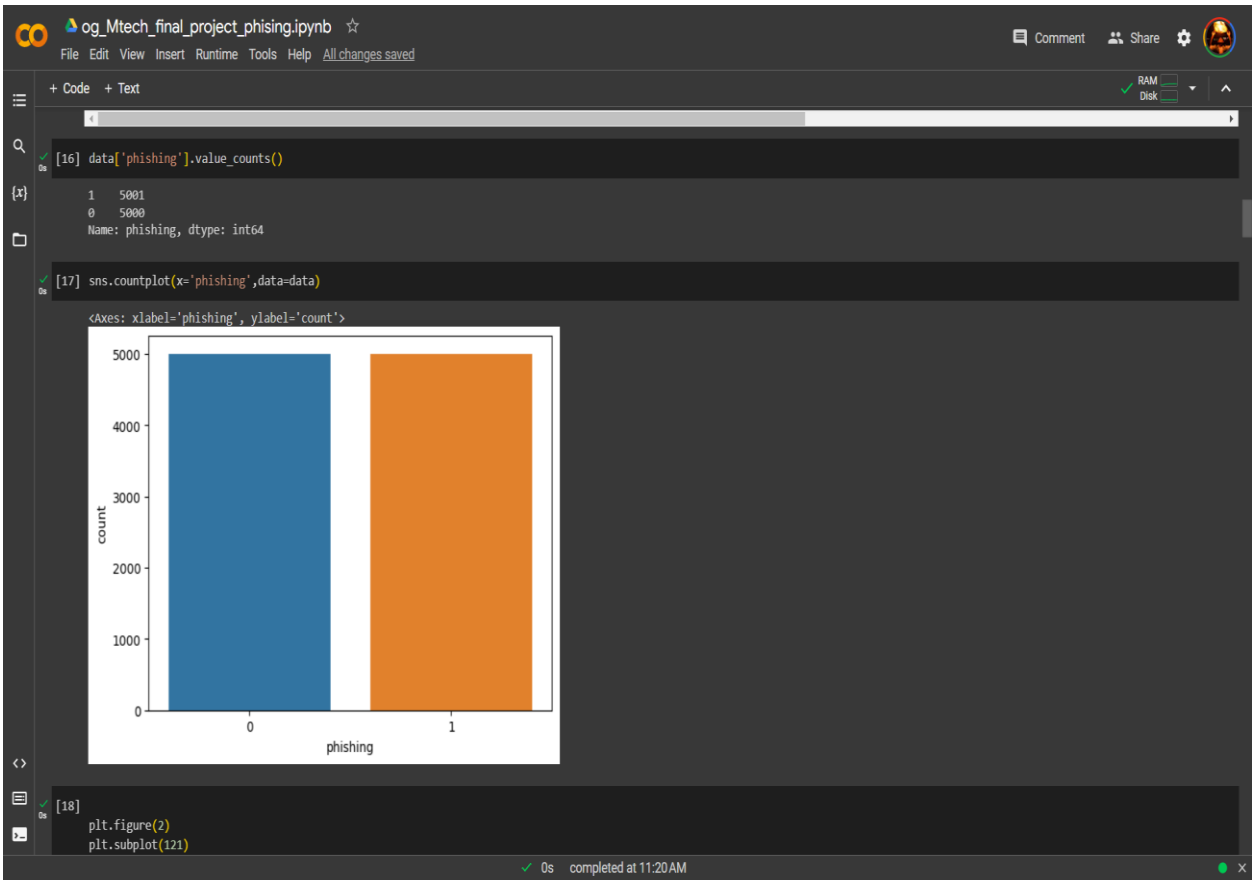


Fig 4 : Sample Code screenshot 4

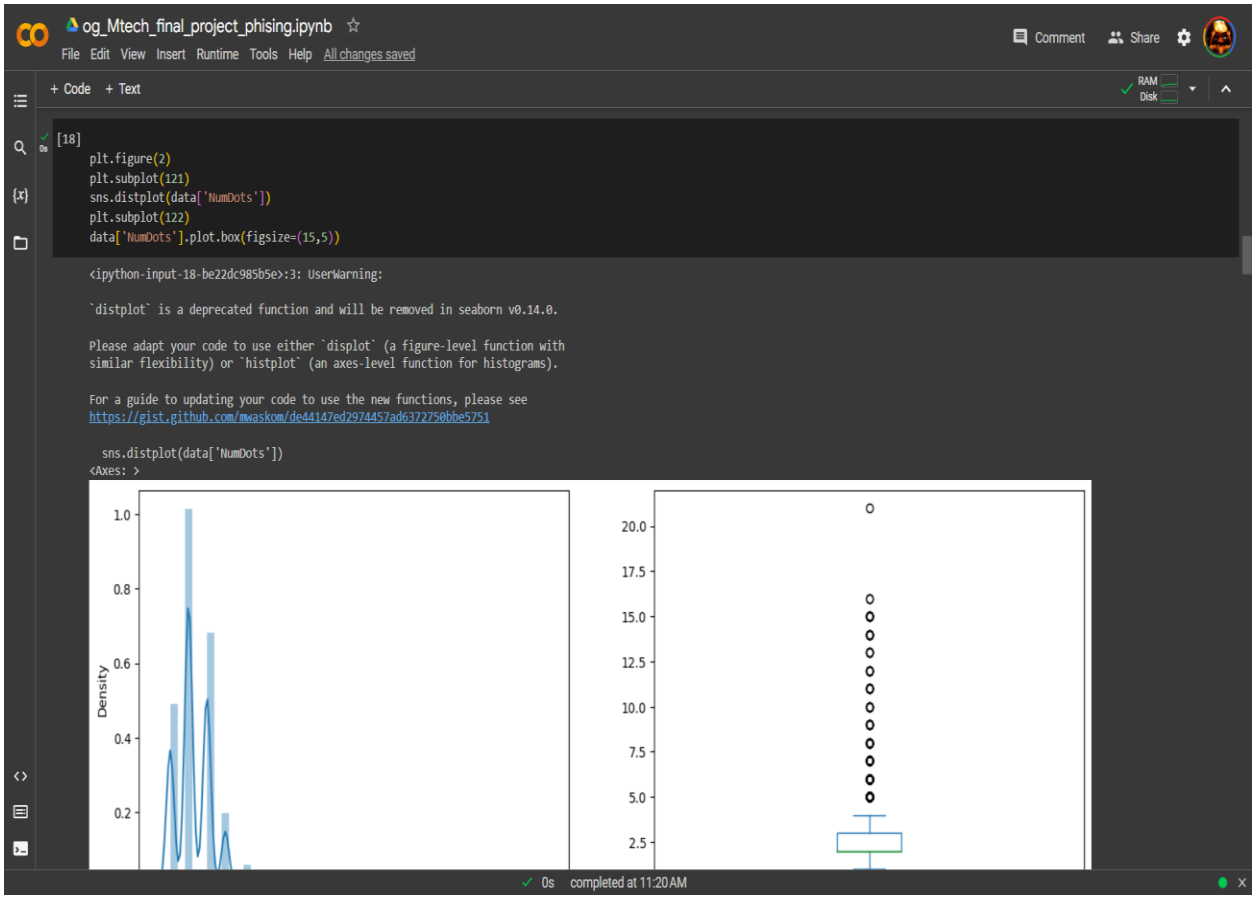


Fig 5 : Sample Code screenshot 5

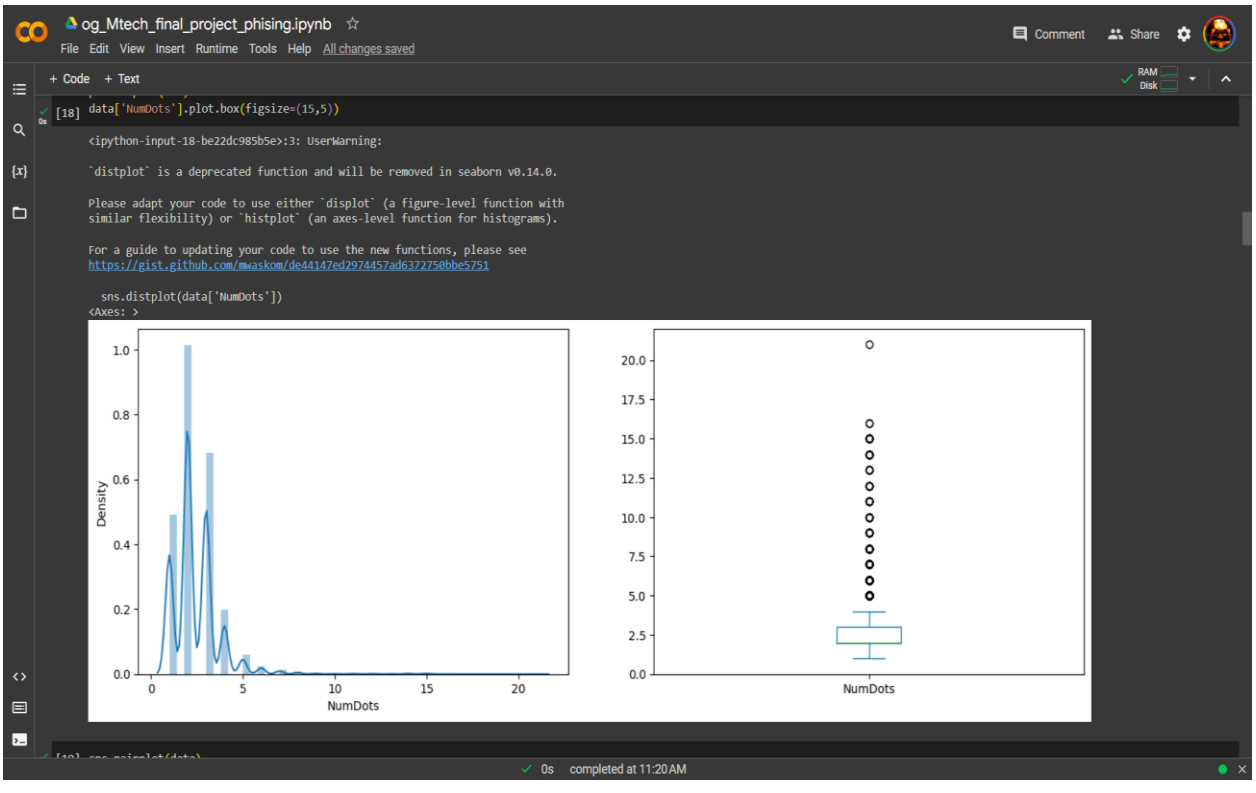


Fig 6 : Sample Code screenshot 6

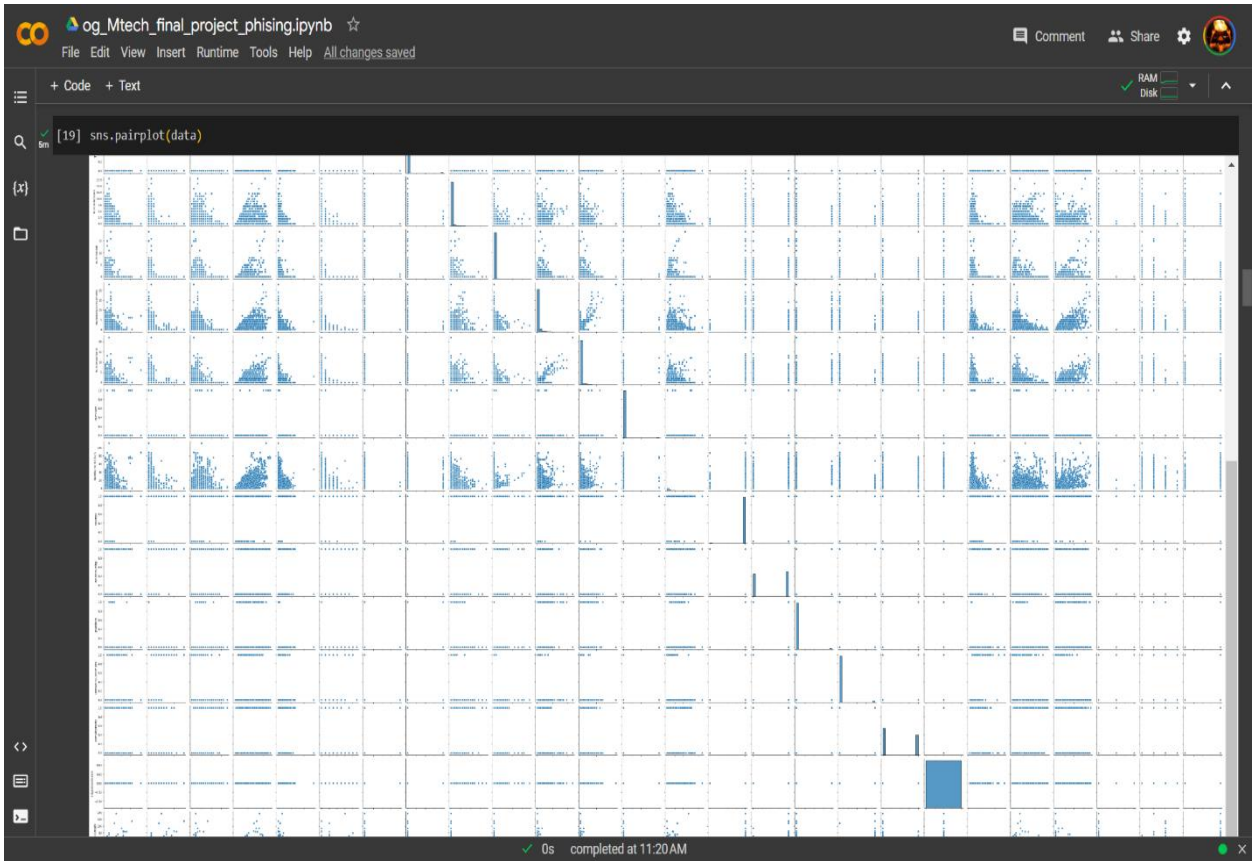


Fig 7 : Sample Code screenshot 7

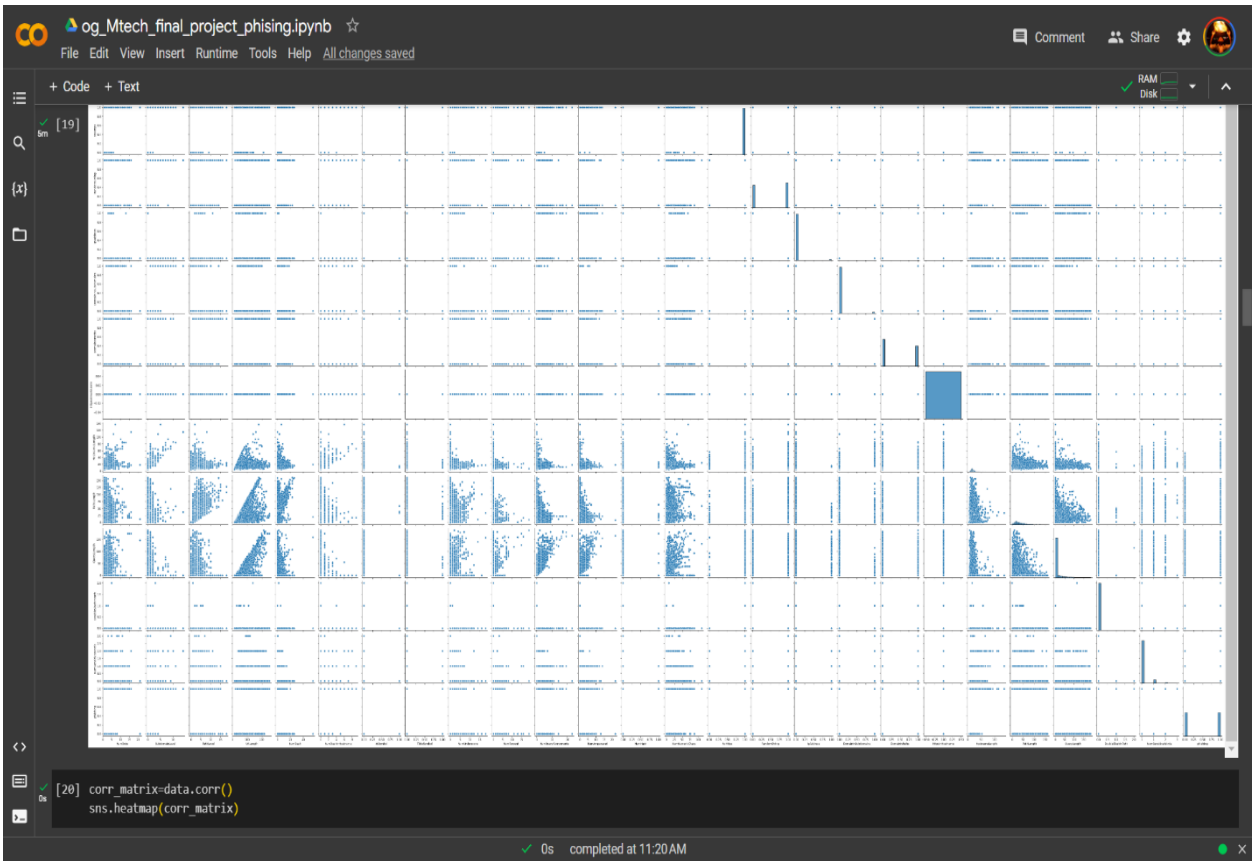


Fig 8 : Sample Code screenshot 8

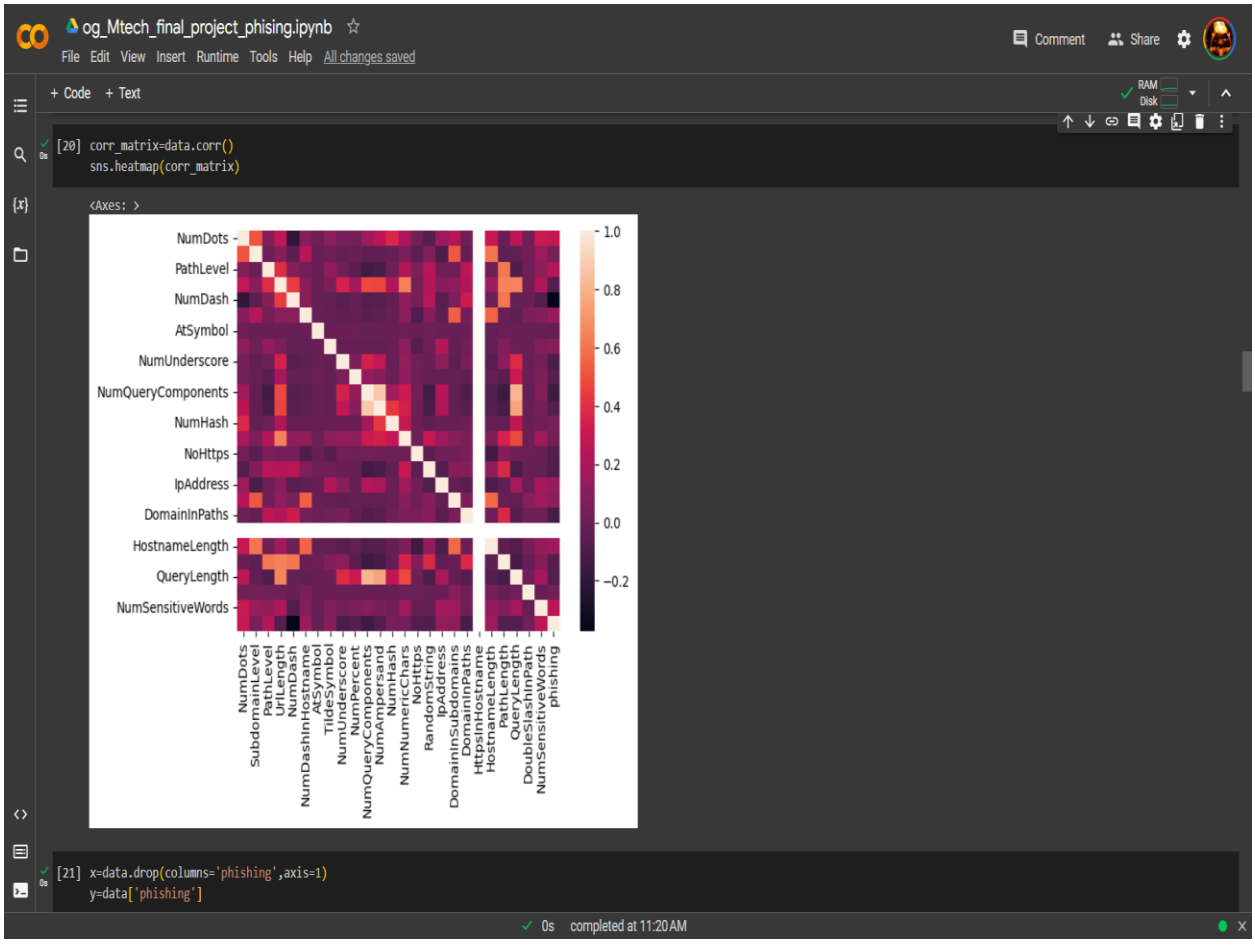


Fig 9 : Sample Code screenshot 9

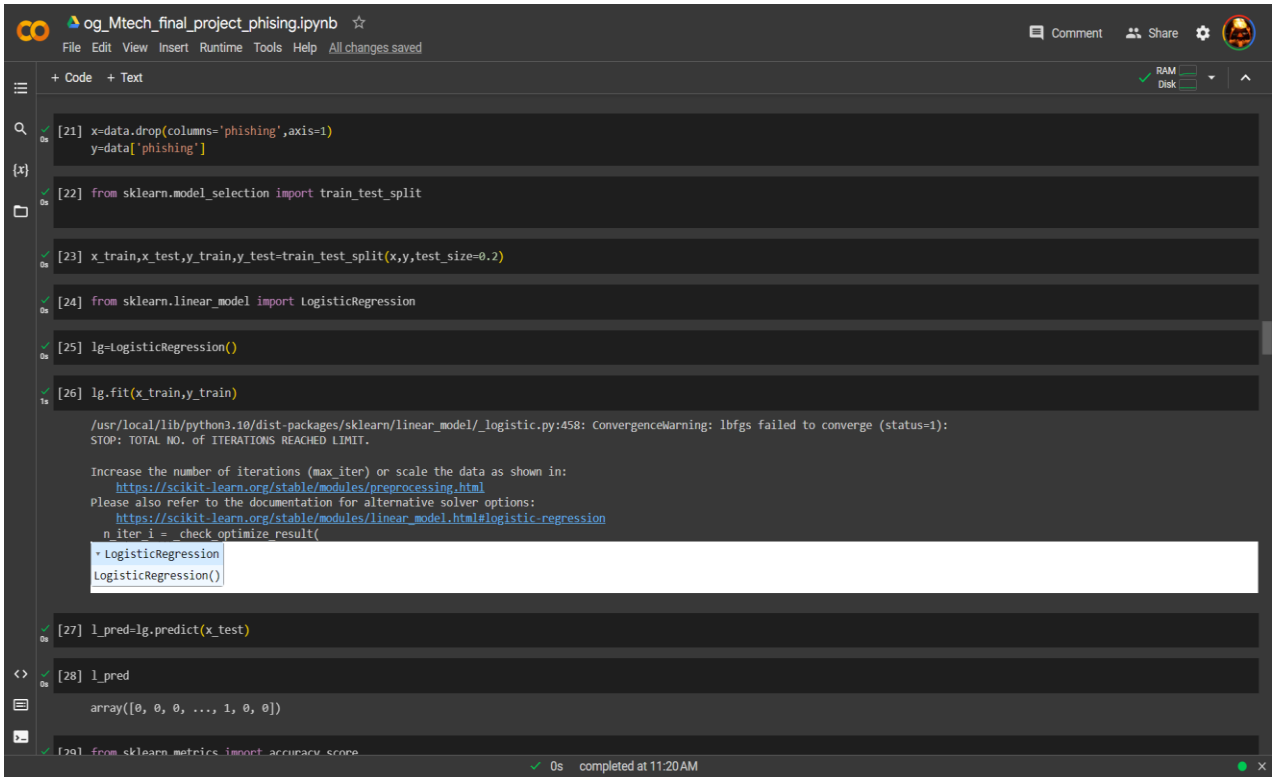


Fig 10 : Sample Code screenshot 10

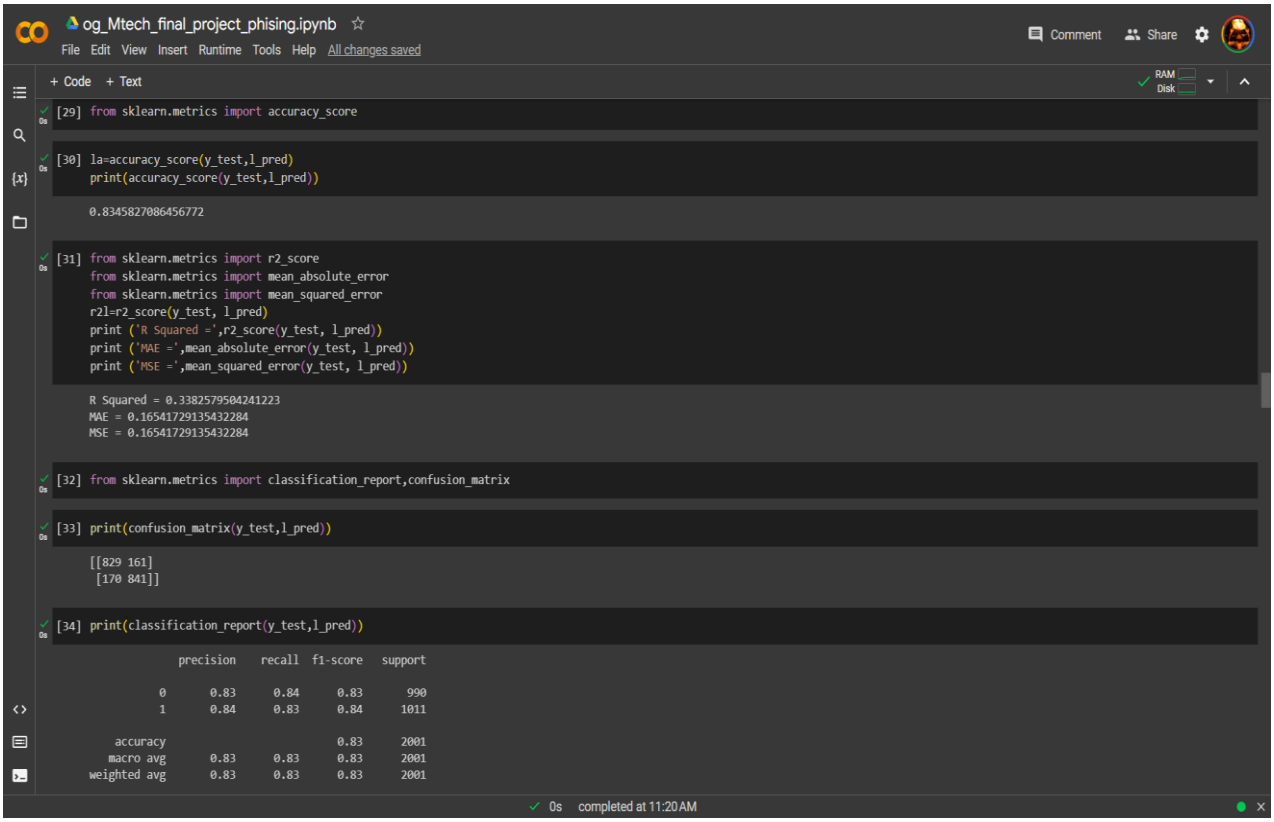


Fig 11 : Sample Code screenshot 11

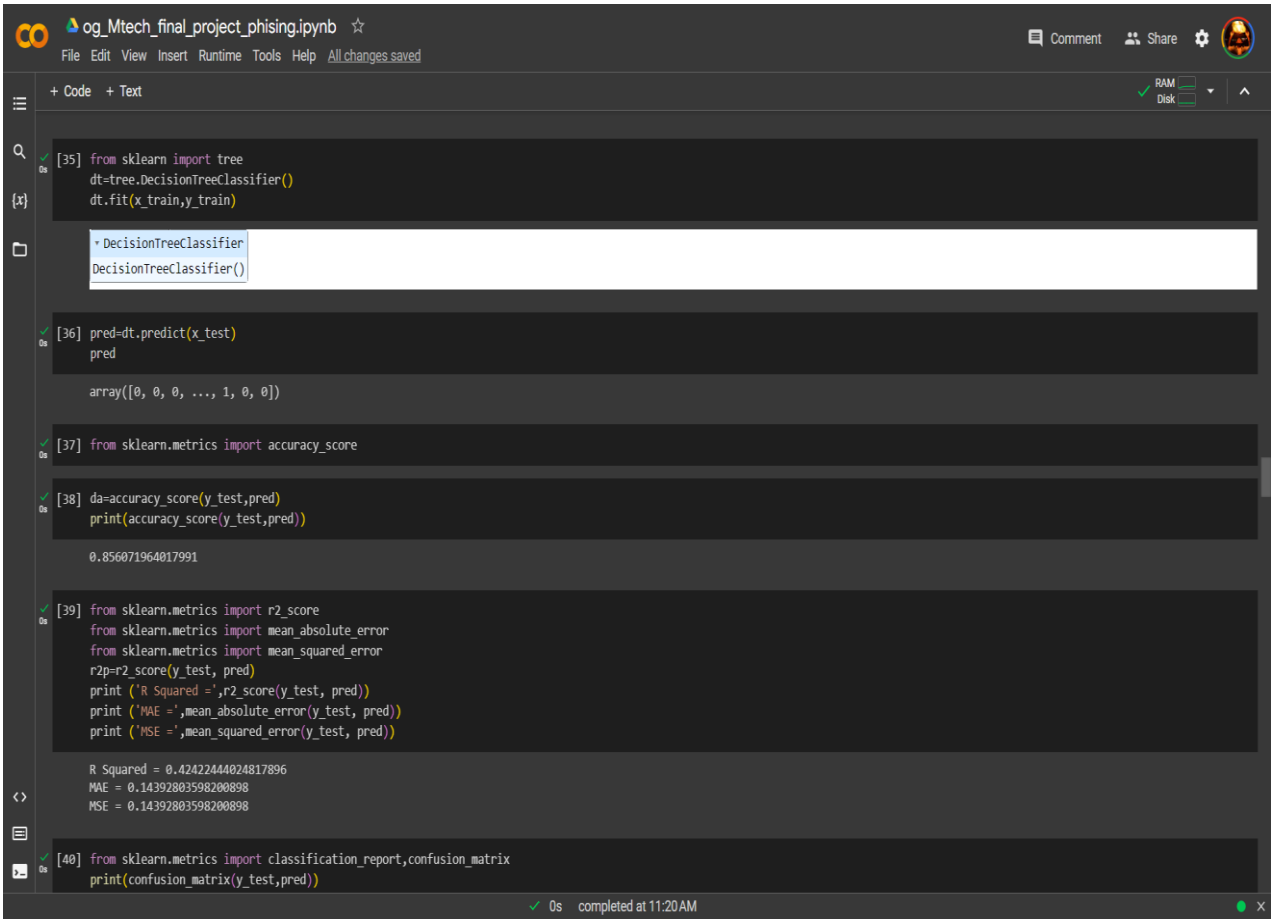


Fig 12 : Sample Code screenshot 12

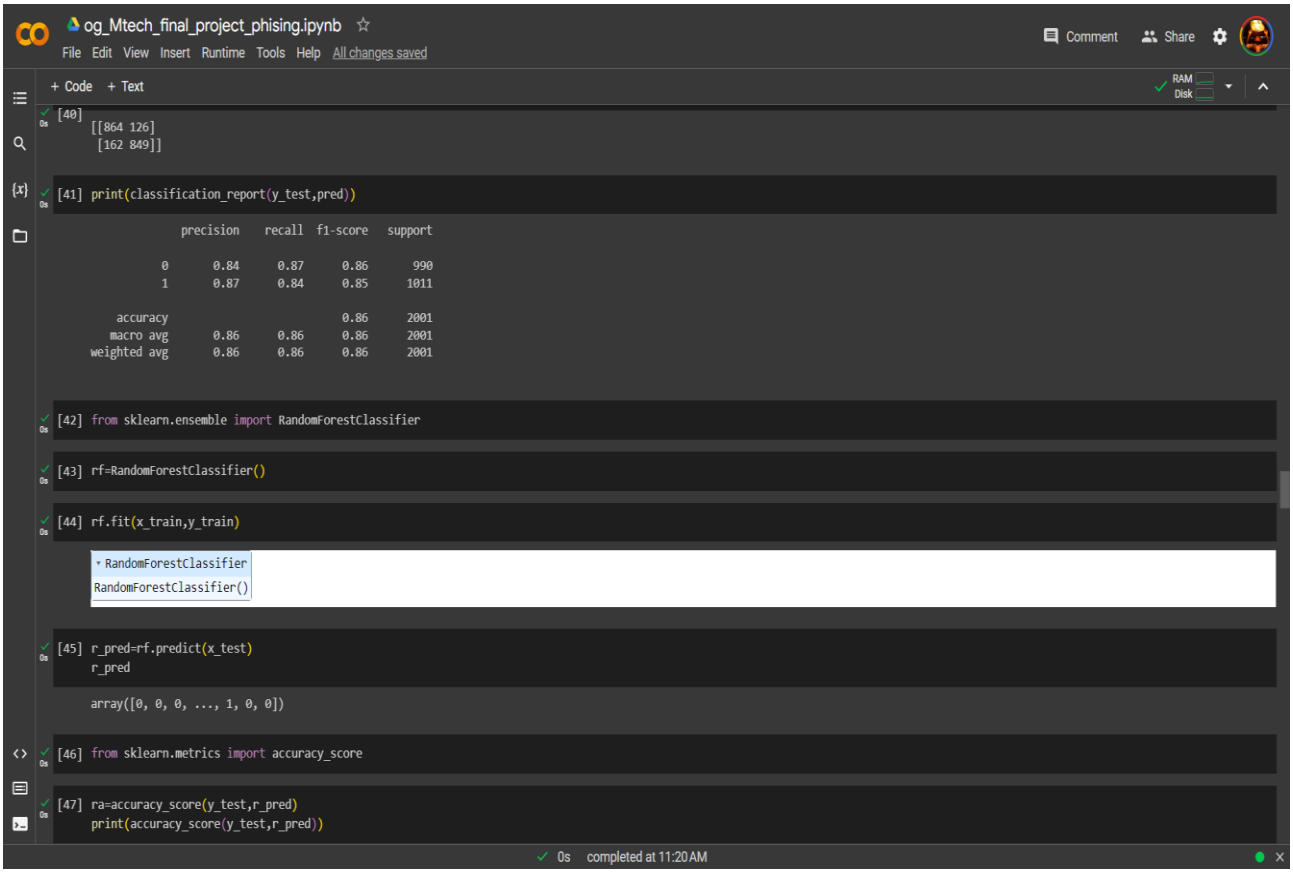


Fig 13 : Sample Code screenshot 13

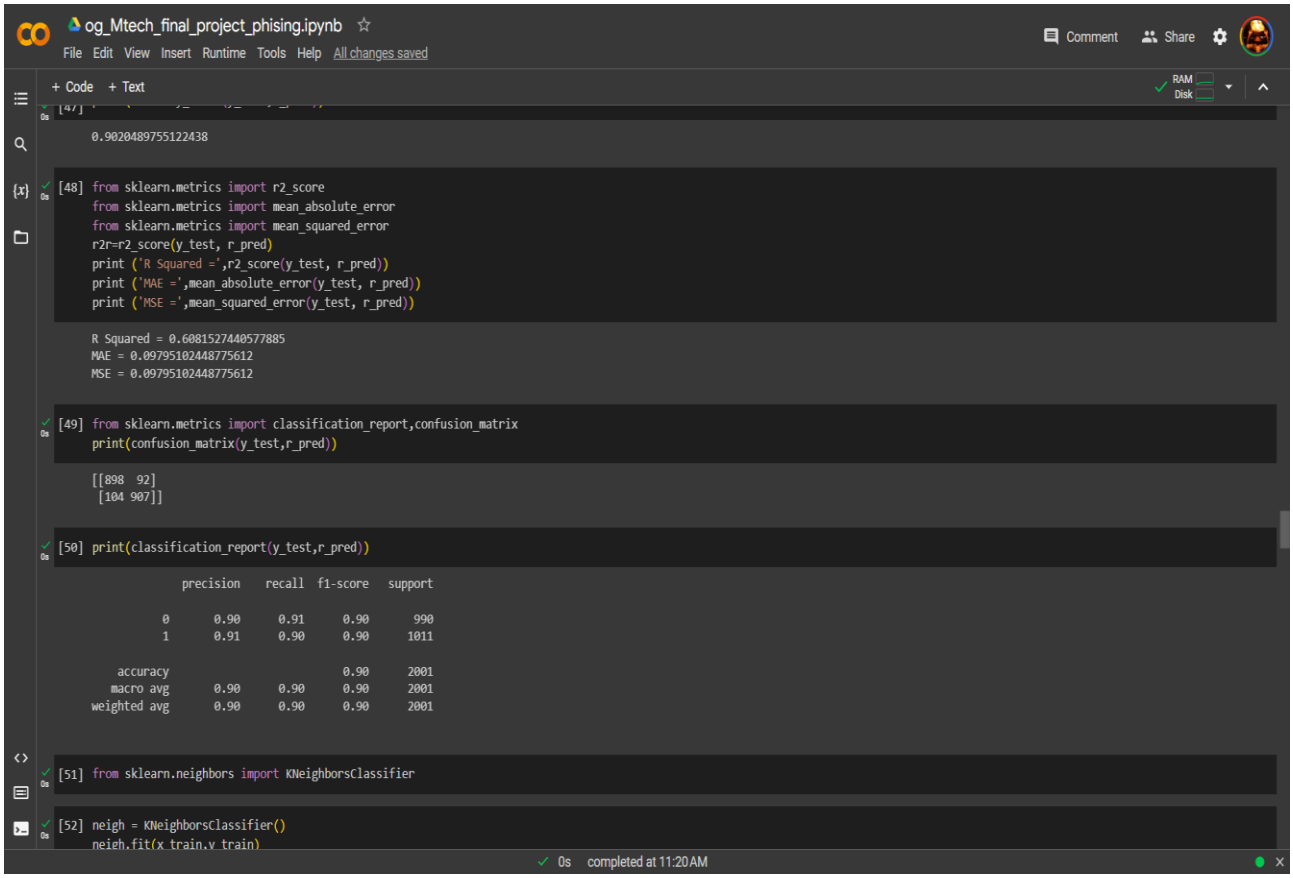


Fig 14 : Sample Code screenshot 14

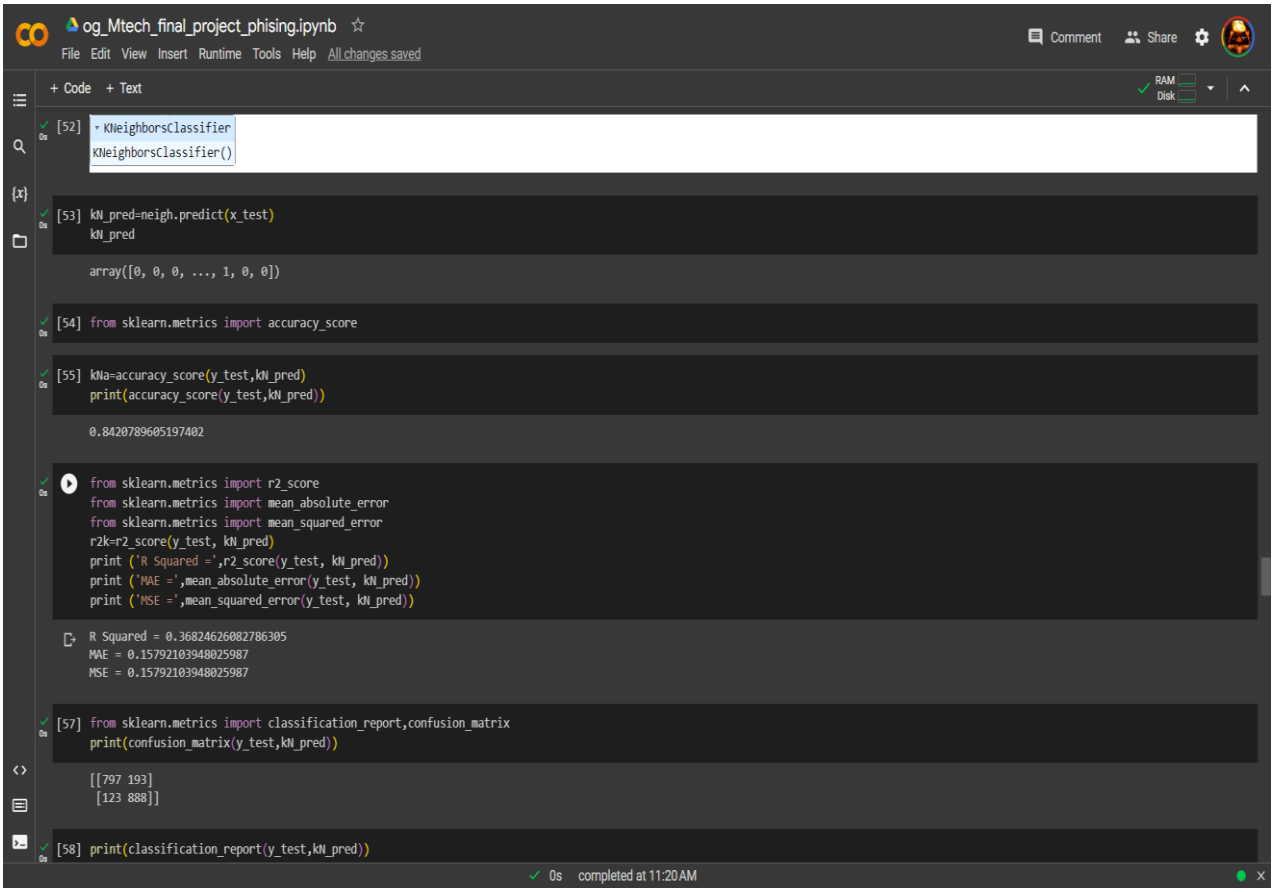


Fig 15 : Sample Code screenshot 15

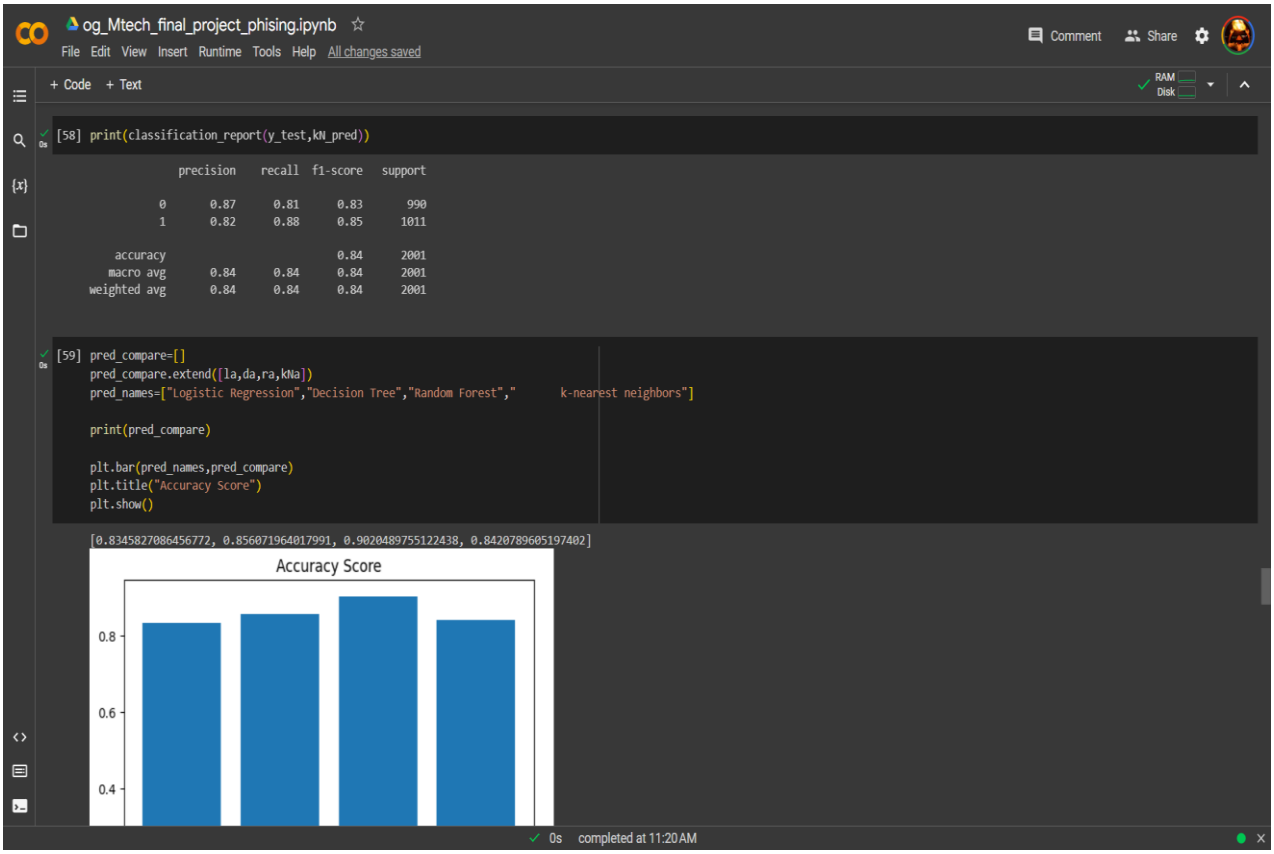


Fig 16 : Sample Code screenshot 16

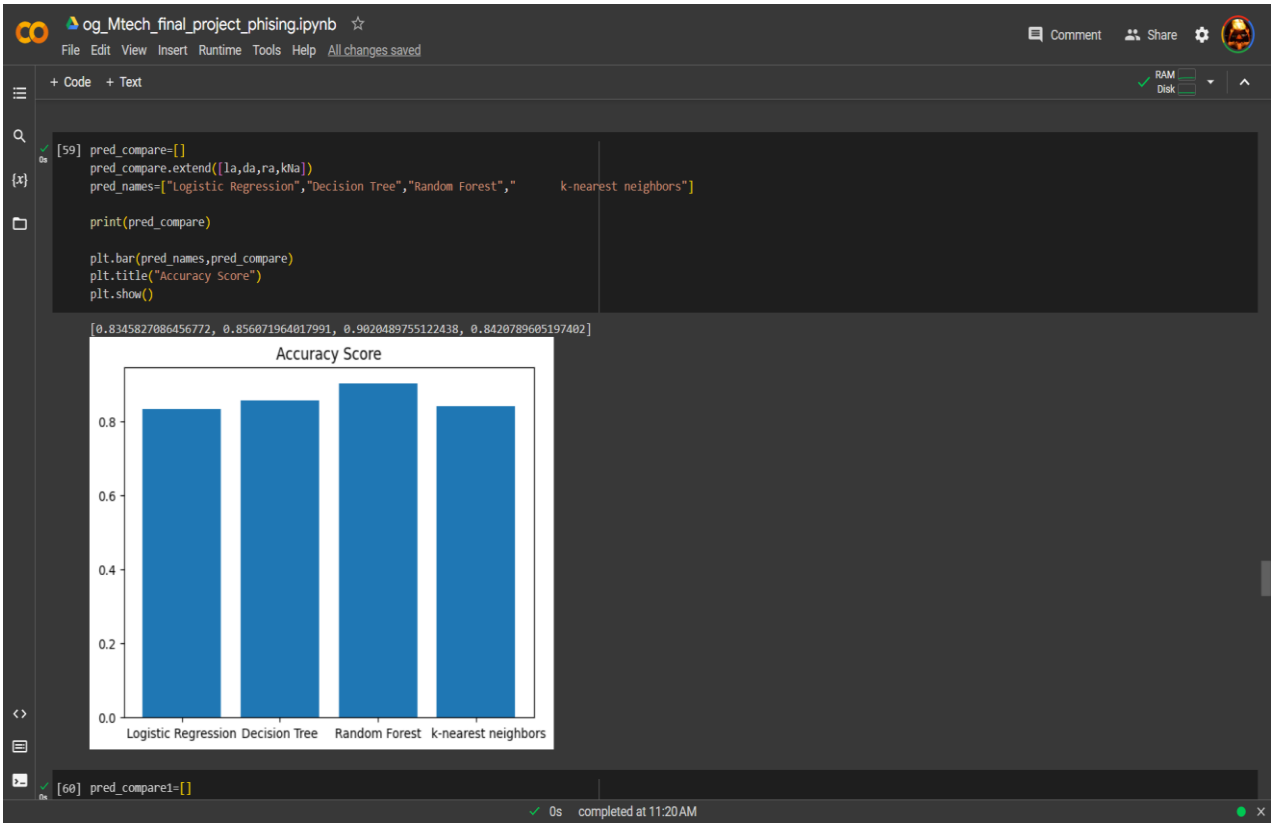


Fig 17 : Sample Code screenshot 17

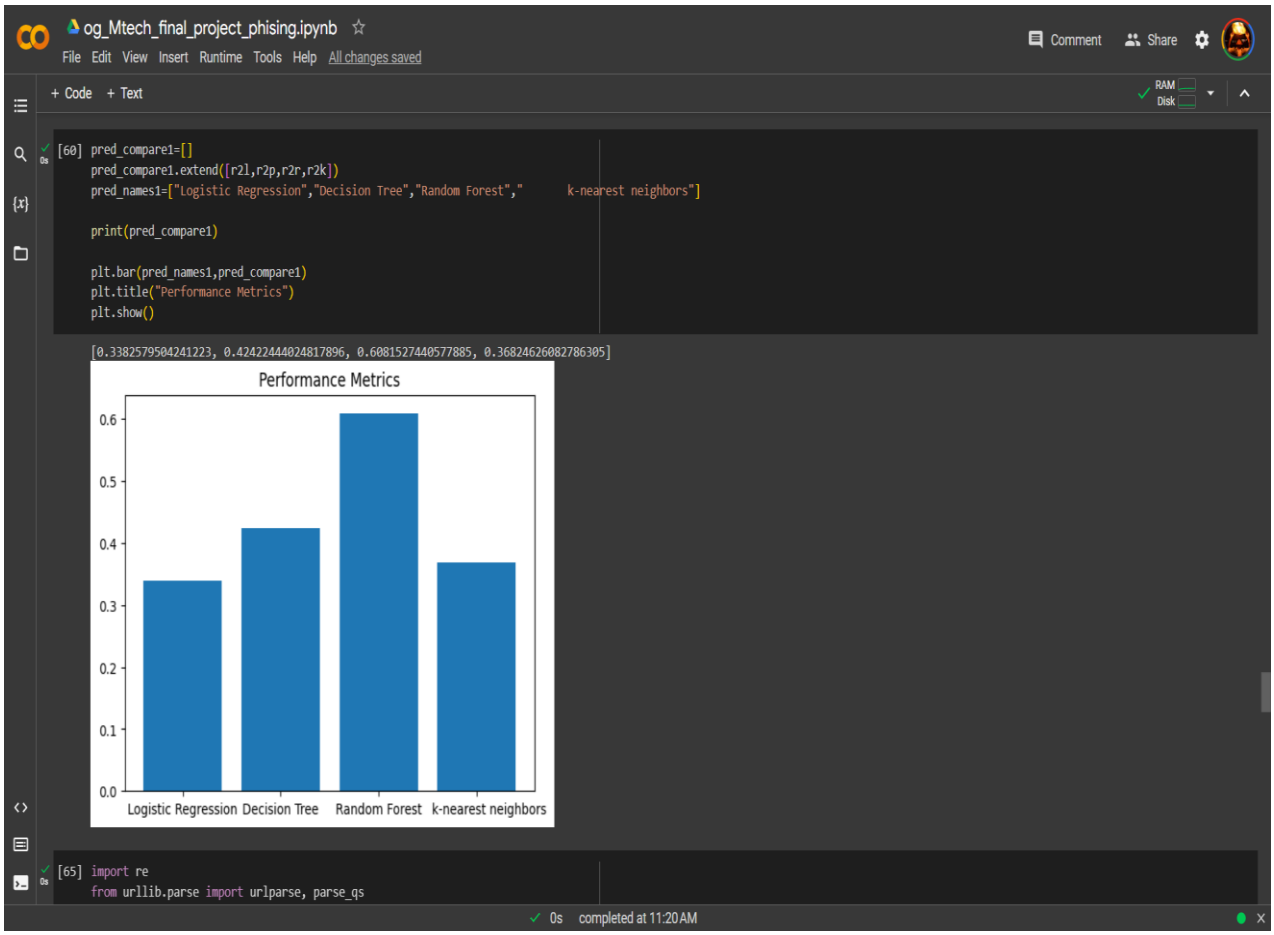


Fig 18 : Sample Code screenshot 18

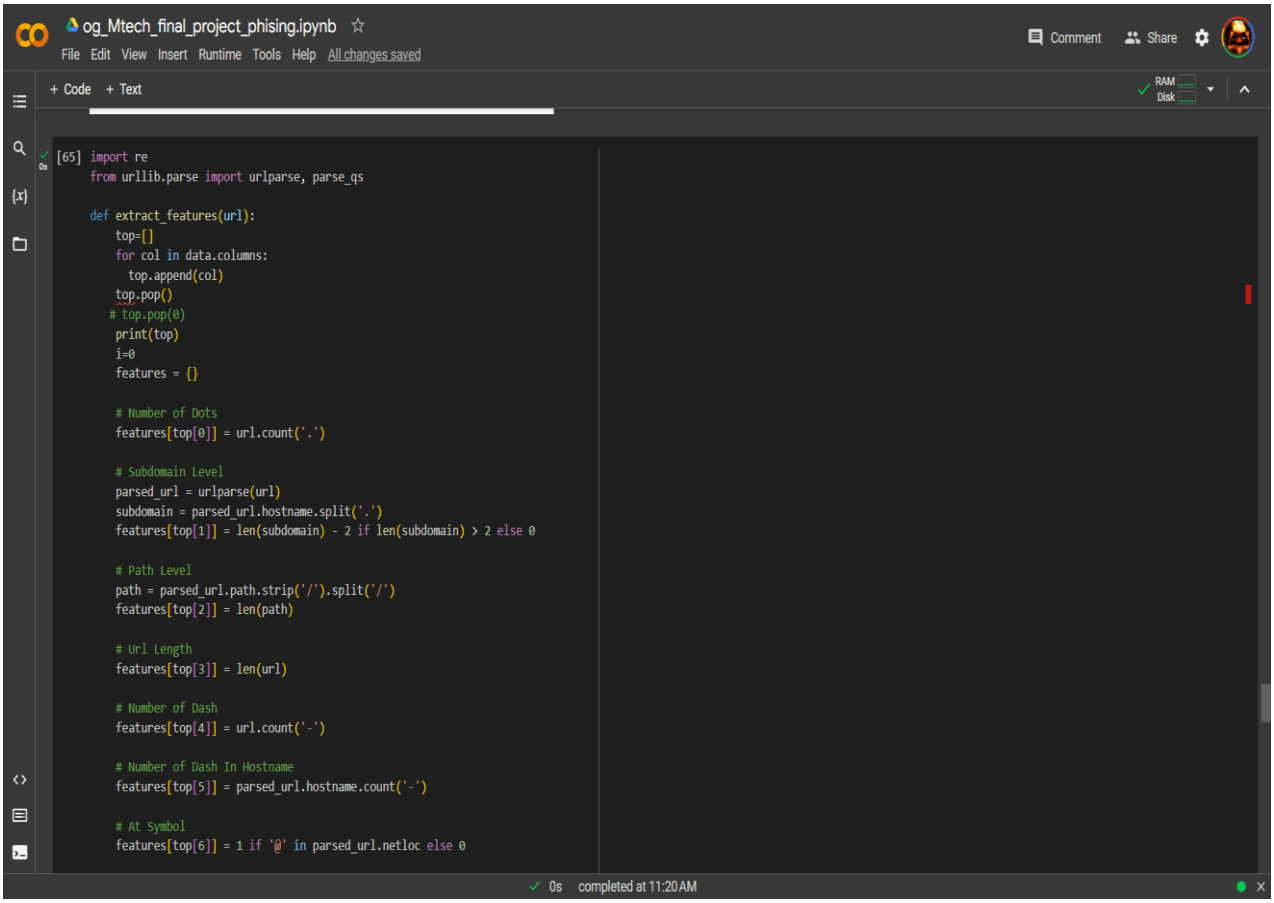


Fig 19 : Sample Code screenshot 19

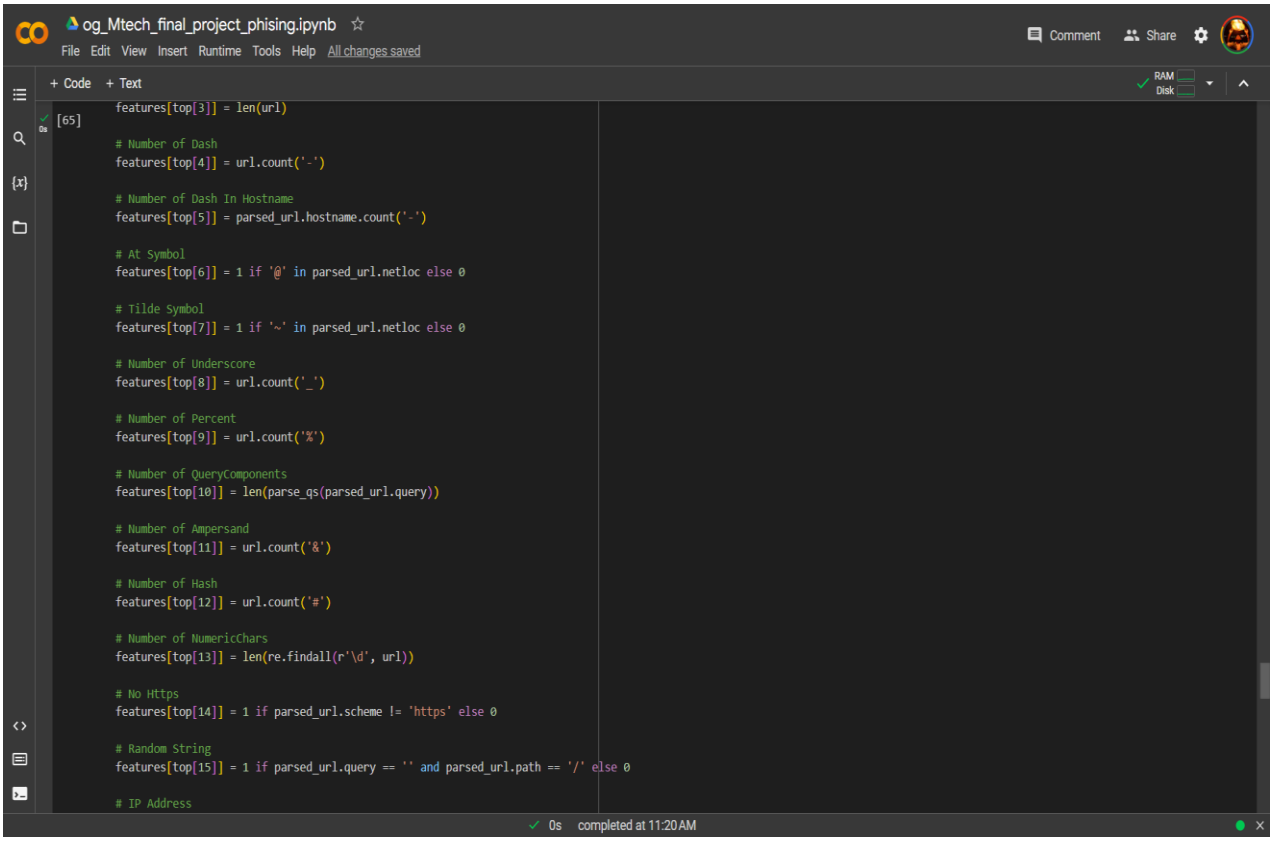


Fig 20 : Sample Code screenshot 20

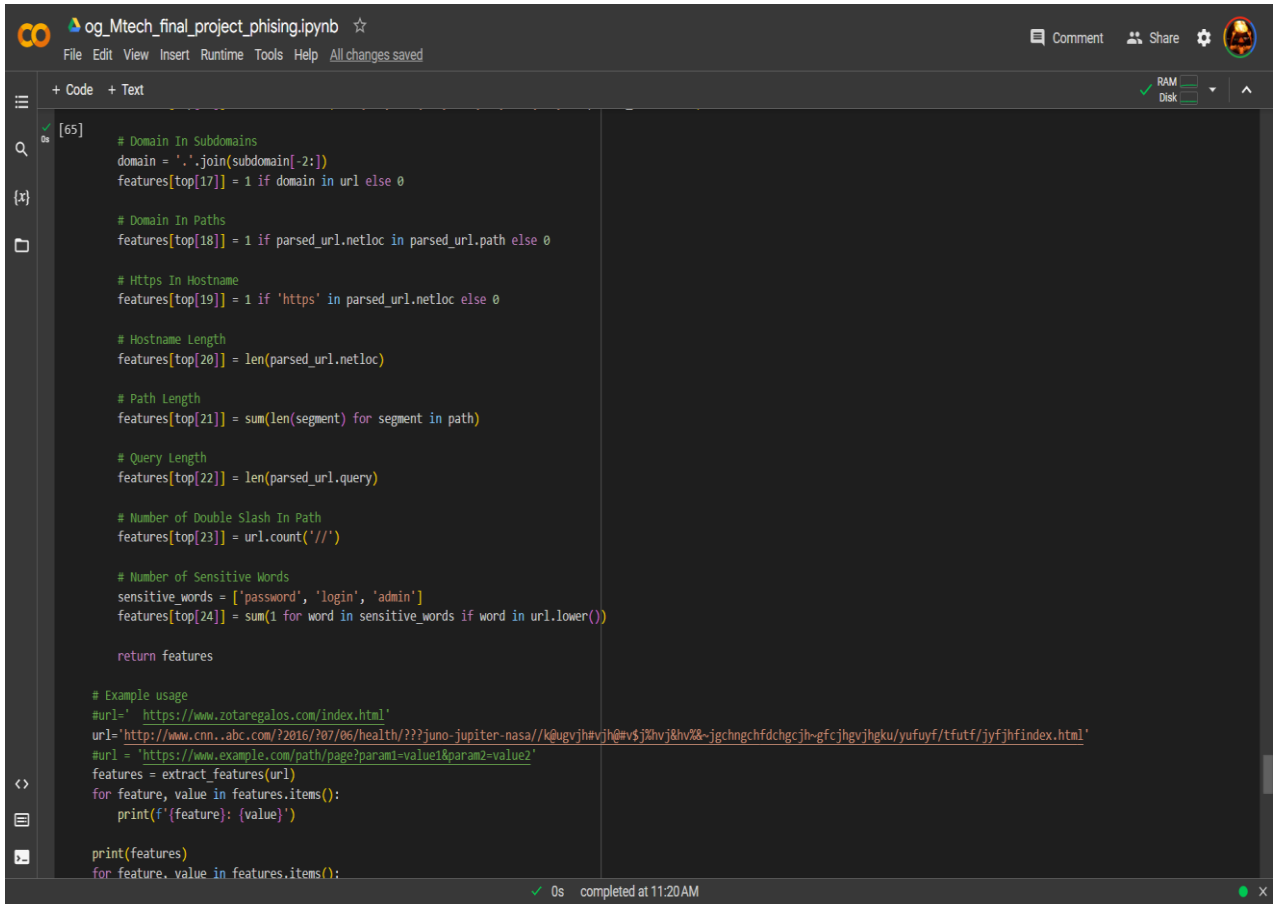


Fig 21 : Sample Code screenshot 21

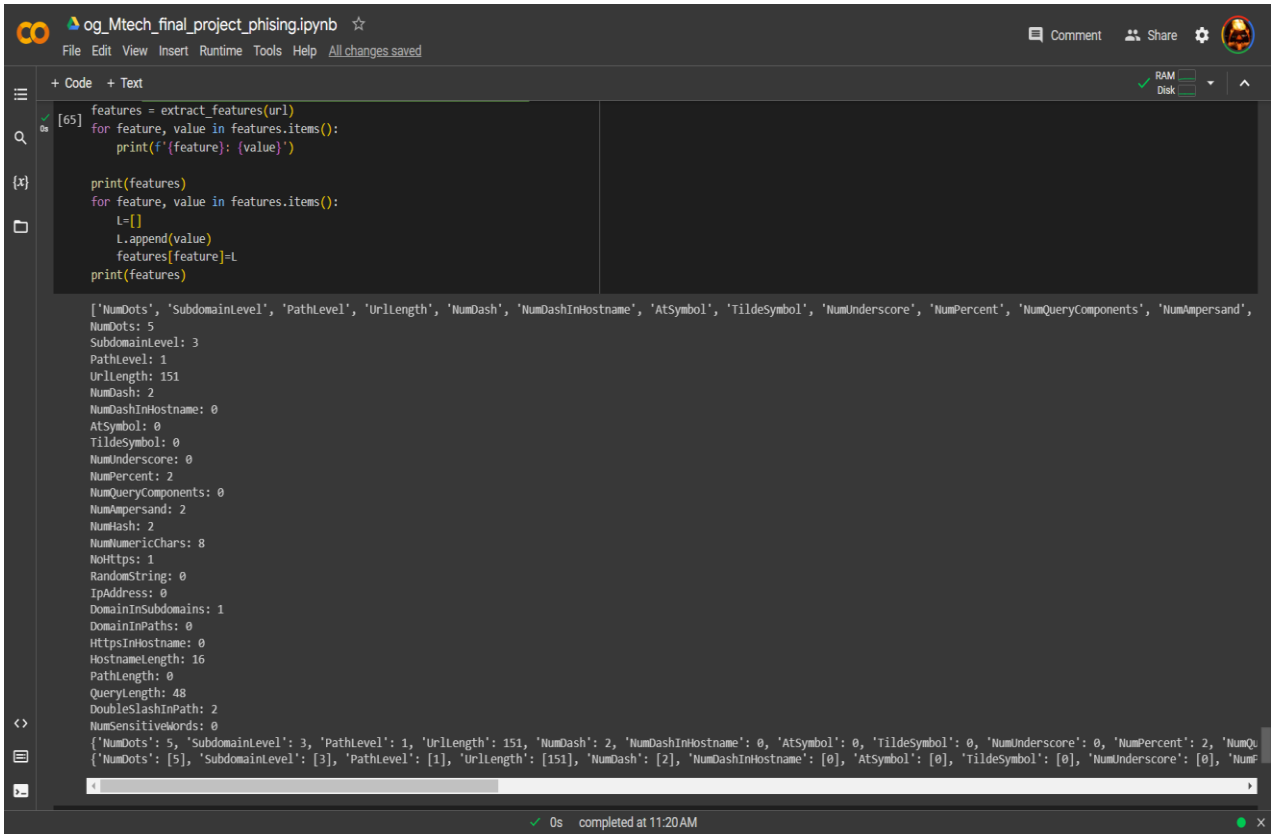


Fig 22 : Sample Code screenshot 22

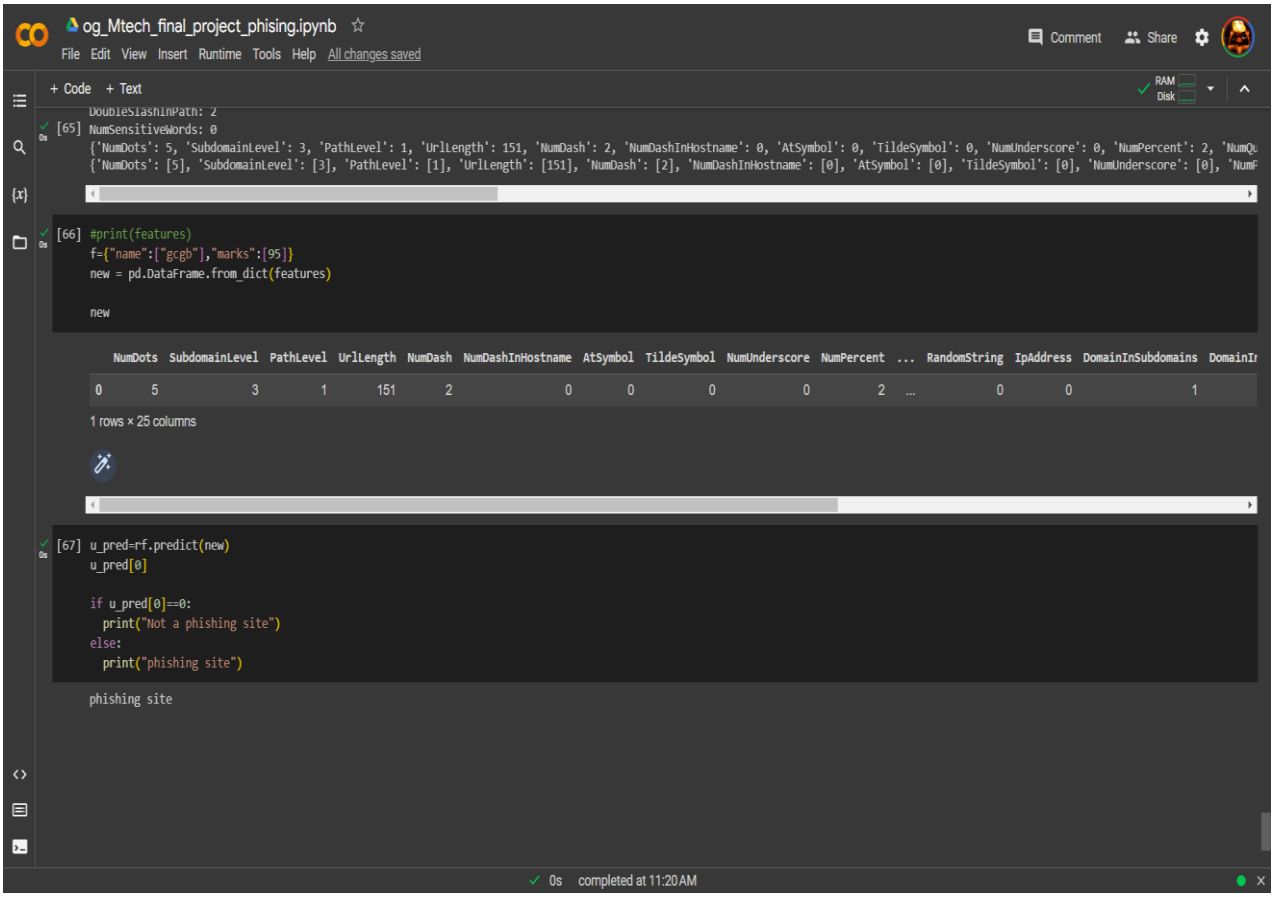
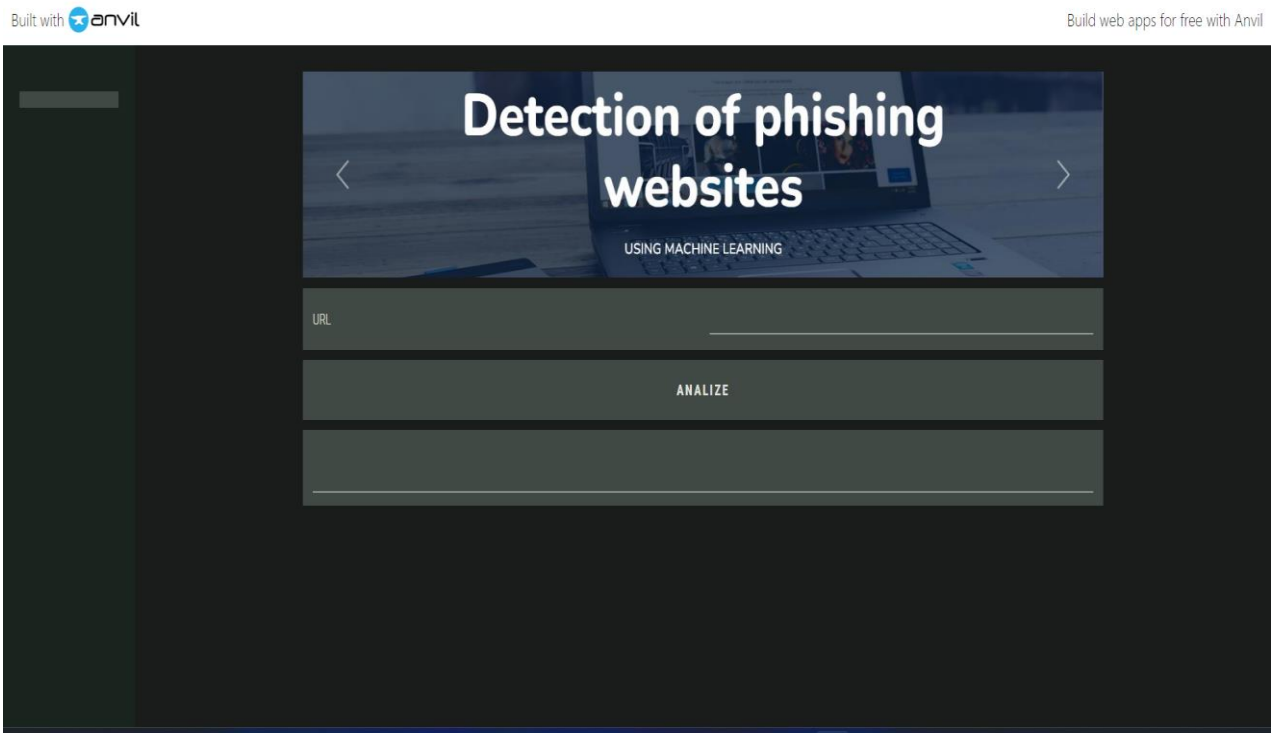


Fig 23 : Sample Code screenshot 23

PROJECT DEPLOYMENT

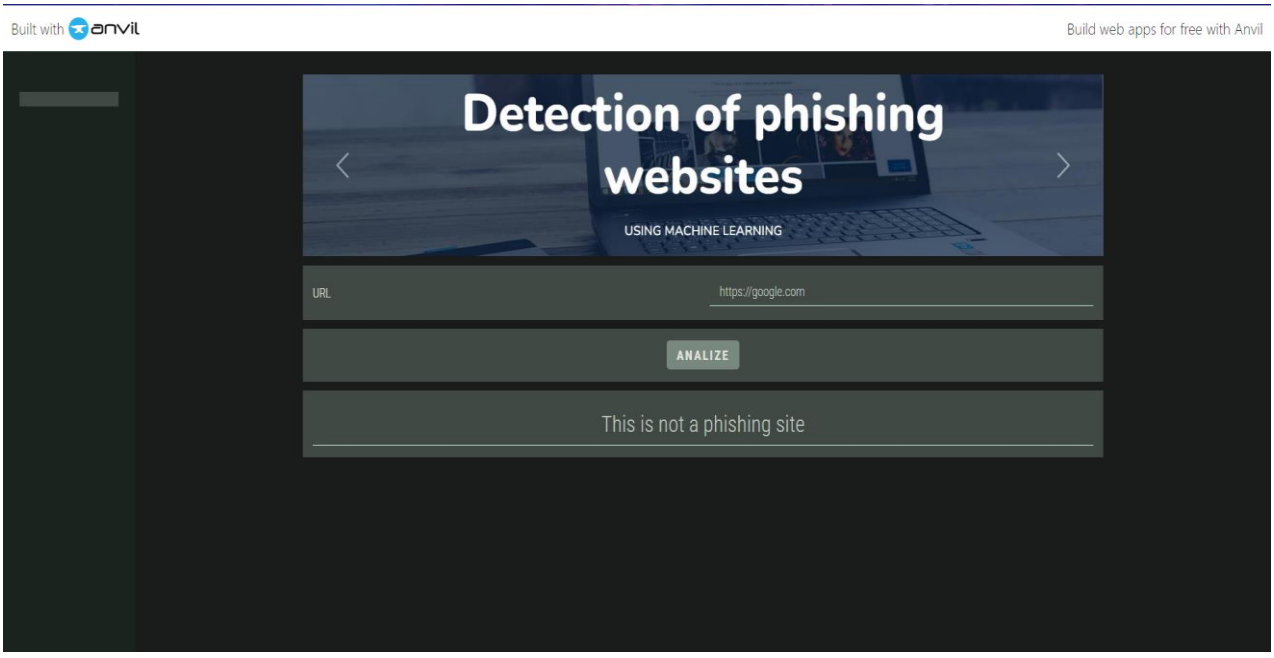
First the Users need to click on a link to open the website used for checking a url is phishing or not.

Link - https://homely-tricky-cancel.anvil.app

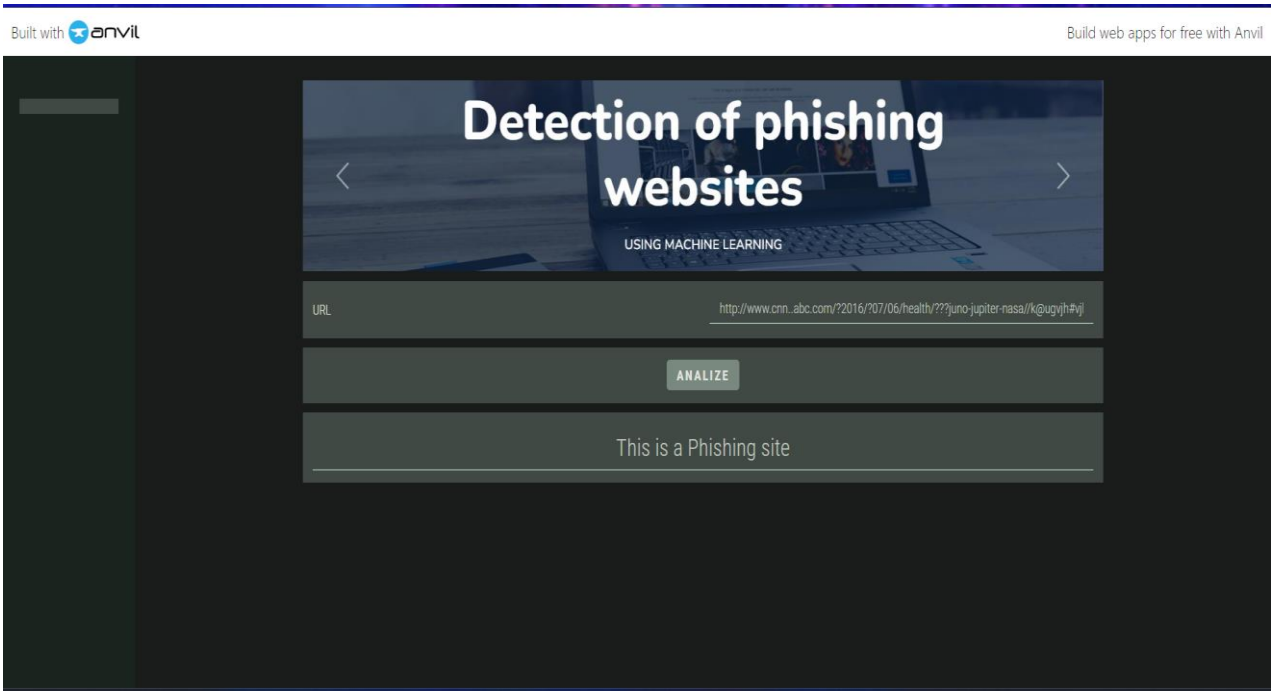


UI of the Website for checking the URL

www.google.com is not a phishing website, so the outputpredicted is “Normal”.



http://www.cnn..abc.com/?2016/?07/06/health/???juno-jupiter-nasa//k@ugvjh#vjh@#v$j%hvj&hv%&~jgchngchfdchgcjh~gfcjhgvjhgku/yufuyf/tfutf/jyfjhfindex.html Is a phishingwebsite, so output predicted is “Phishing”.



APPENDIX

1. Dataset Description:

- The dataset used in this project is a phishing website dataset.

- It contains various URL-based features and a target variable indicating whether the website is a phishing site or not.

- The dataset was preprocessed to handle missing values, and basic exploratory data analysis was performed to gain insights into the data.

2. Experimental Setup:

- The dataset was divided into training and testing sets using the train_test_split function from the sklearn library.

- Four machine learning algorithms were employed: Logistic Regression, Decision Tree, Random Forest, and k-nearest neighbors.

- Performance metrics such as accuracy, confusion matrix, and classification report were used to evaluate the performance of the algorithms.

3. Results and Analysis:

- The experimental results showed varying levels of accuracy for the different algorithms.

- Logistic Regression achieved a certain level of accuracy, while Decision Tree, Random Forest, and k-nearest neighbors achieved higher accuracy.

- The performance metrics provided insights into the classification results, including true positive, true negative, false positive, and false negative rates.

4. Discussion:

- Strengths and Limitations of the Proposed Approach:

- The proposed approach leverages machine learning algorithms and URL-based features, which are effective in distinguishing phishing websites.

- However, the approach relies heavily on the quality and relevance of the selected features, and there may be limitations in capturing all aspects of phishing websites.

- The approach also assumes that the dataset used for training and testing is representative of real-world phishing websites.

- Comparison with Previous Research and Identification of Novel Contributions:

- The proposed approach aligns with previous research in phishing website classification using machine learning techniques.

- However, the selection and combination of features in this project may contribute to novel insights and improvements in classification accuracy.

- The experimental evaluation and comparison of different algorithms provide a comprehensive analysis of their performance in the context of phishing detection.

- Suggestions for Future Improvements and Enhancements:

- Incorporate more advanced feature engineering techniques, considering additional aspects such as page content and user behavior.

- Explore ensemble learning methods to combine the strengths of multiple algorithms and improve overall classification accuracy.

- Investigate the use of deep learning models, such as convolutional neural networks, to capture intricate patterns in URL structures and improve phishing detection.

- Continuously update the dataset to incorporate emerging phishing techniques and adapt the classification system accordingly.

5. Conclusion:

- The project demonstrates the effectiveness of machine learning algorithms and URL-based features in phishing website classification.

- The proposed approach contributes to cybersecurity efforts by enabling accurate identification and proactive protection against phishing attacks.

- Further research and improvements are necessary to enhance the detection capabilities and stay ahead of evolving phishing techniques.

Note: The code provided in the previous sections serves as an example and may require modifications and additional considerations for specific use cases.