**Machine Learning and Artificial Intelligence helps an IOT (Internet of Things) from Phishing links**

**Akash Dhar**

Bachelor’s of Computer Application of Dooars Academy of Technology and Management (of Affiliation), Maulana Abul Kalam Azad University of Technology formerly known as West Bengal University of Technology, Alipurduar,West Bengal, India, 736121; [bcadept.199.datm@gmail.com](mailto:bcadept.199.datm@gmail.com)

Corresponding Author: akash1711dhar@gmail.com

Date of Submission: 23-05-2023

***Abstract –* The future of Machine Learning and Artificial Intelligence will have a deep economical, commercial, and social impact on our lives, the participating nodes in IoT (Internet of Things) networks are usually limited resources, which makes them alluring targets for cyber attacks like phishing links.** **Advancements in Internet and cloud technologies have led to a significant increase in electronic trading in which consumers make online purchases and transactions. This growth leads to unauthorized access to users’ sensitive information and damages the resources of an enterprise. Phishing is one of the familiar attacks that trick users to access malicious content and gain their information. In terms of website interface and uniform resource locator (URL), most phishing WebPages look identical to the actual WebPages. Various strategies for detecting phishing websites, such as blacklist, heuristic, Etc., have been suggested.  In this regard, extensive efforts have been made to address the security and privacy issues in IoT networks primarily through traditional cryptographic approaches. However, the unique characteristics of IoT nodes render the existing solutions insufficient to encompass the entire security spectrum of the IoT networks. Machine Learning (ML) techniques.** **Which are able to provide embedded intelligence in the IoT devices and networks can be leveraged to cope with different security problems. In this paper, we systematically review the security requirements, attack vectors, and the current security solutions for the IoT networks. We then shed light on the gaps in these security solutions that call for ML and DL approaches. We also discuss in detail the existing ML and DL solutions for addressing different security problems in IoT networks. We also discuss several future research directions for ML- and DL-based IoT security.**

***Keywords— Machine Learning, Deep Learning, IoT, Internet of Things, Phishing links, cyber Attacks, Security, URL, Privacy, WebPages, Cryptographic.***

**1. Introduction**

As part of planning and preparation for this paper, a study of security using IoT by the help of ML and DL which is considered as an interconnected and distributed network of embedded systems communicating through wired or wireless communication technologies. IoT empowered with limited computation, storage, and communication capabilities as well as embedded with electronics (such as sensors and actuators), software, and network connectivity that enable these objects to collect, sometimes process, and exchange data. The IoT things are refer to the objects from our daily life ranging from smart house-hold devices such as smart bulb, smart adapter, smart meter, smart refrigerator, smart oven, AC, temperature sensor, smoke detector, IP camera, to more sophisticated devices such as Radio Frequency Identification (RFID) devices, heartbeat detectors, accelerometers, sensors in parking lot, and a range of other sensors in automobiles etc . IoT devices generate a sheer amount of data and therefore, traditional data collection, storage, and processing techniques may not work at this scale. Furthermore, the sheer amount of data can also be used for patterns, behaviours, predictions, and assessment. Additionally, the heterogeneity of the data generated by IoT creates another front for the current data processing mechanisms. Therefore, to harness the value of the IoT-generated data, new mechanisms are needed. In this context, Machine Learning (ML) is considered to be one of the most suitable computational paradigms to provide embedded intelligence in the IoT devices. ML can help machines and smart devices to infer useful knowledge from the device- or human-generated data. It can also be defined as the ability of a smart device to vary or automate the situation or behaviour based on knowledge which is considered as an essential part for an IoT solution. ML techniques have been used in tasks such as classification, regression and density estimation. Variety of applications such as computer vision, fraud detection, bio-informatics etc. In this paper, however, we focus on the applications of ML in providing security and privacy services to the IoT networks. In the following, we discuss the existing surveys that are already published in the literature covering different aspects of security in IoT networks through ML.

**2. LITERATURE REVIEW**

Majority of occurrences that we encounter on a daily basis involve a certain level of ambiguity, that which link is not a phishing links even our system firewall some of the time is not working whenever we are using the internet and surfing various links so that we are not known that those links are phishing links are not. In this regard IoT helps to achieve the safe link by some filtering algorithm using ML.

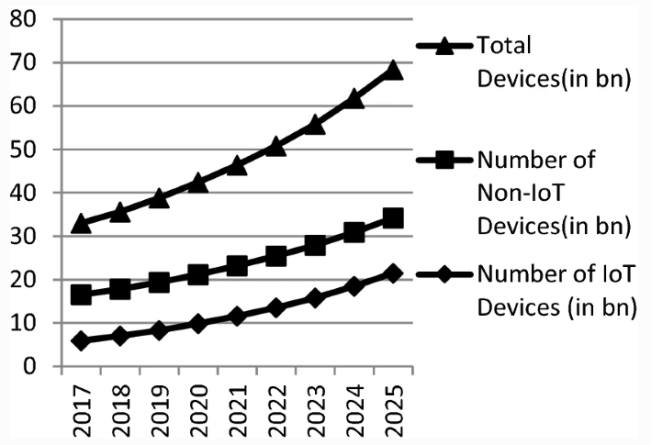


Fig1: IoT devices are used by the user

## **2.1 Problem Statement**

There are many IoT security solutions out in the market for enterprise-level protection but not enough of resources are available for average end-users to learn how to securely manage and protect their information and privacy. Despite the comfort from IoT devices, it would be extremely hard for people with non-technical background to understand complex technology paper to build a sound and secure network themselves. Therefore, many of the tech-giants are focusing on providing the comprehensive platform service that people need. This project focuses on the services of three world-famous tech-companies: Microsoft, Google, and Amazon.

There are numerous IoT devices and applications which support comfortable usage of customers in every area of the life. For example, electric vehicle charger that support Android application and Bluetooth connection: (“Kaspersky Lab Security Services”), smart meter for home electricity usage, Fitbit area tracking personal health information, Google Nest thermostat: (G. Hernandez et al. 1-8), Tesla electric vehicle, chamberlain myQ for home garage door access, drones for work and fun, IP camera system for home surveillance, and millions of other devices are out in the market to attract customers with their features that will let people have more comfort. However, these devices and systems listed have been susceptible to cyber-attacks. Information theft on any of the devices connected to home or personal network can lead to a life destroying results.

**3. RESEARCH QUESTION/HYPOTHESIS/PHILOSOPHY**

## **3.1. Question**

The important question is that IoT can protect from phishing link as well as the Man-in-Middle attack. Using ML algorithm and its testing models . There are also extra sub-questions which are stated as follows:

1. What are the parameters need to be focused to filter the links ?

2. Does the the data is over fitted for predicting which may arise any problems to the IoT to protect the cyber attacks?

## **3.2. Hypothesis**

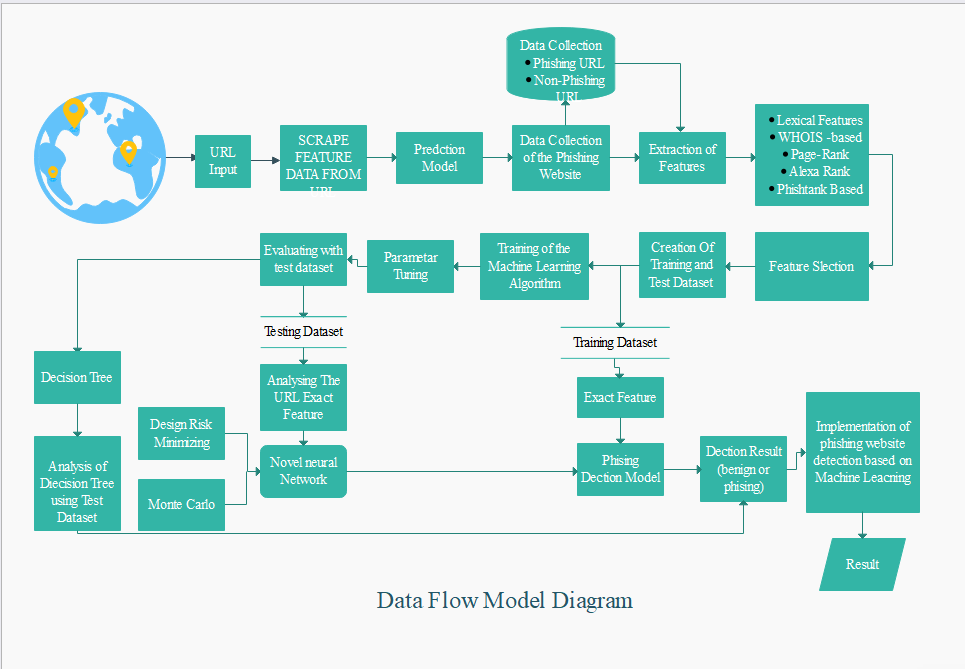
## By applying the Ensemble algorithm to filter the data and trained the ML model which is help to IoT device to protect from phishing links and DL helps to perceive the data and train the network

## **3.3. Philosophy**

## The philosophy behind the proposed model is to find a mechanism to improve the IoT device that can user feel protect during surfing on the internet and the IoT can train from the user ; It perceives the data and update its dataset.

## **4. Methodology**

## **4.1 Structure of the system**



**Fig2 :** Design of system or DFD of system

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

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

**Rule: IF{ 𝑈𝑅𝐿 𝑙𝑒𝑛𝑔𝑡ℎ < 54 → 𝑓𝑒𝑎𝑡𝑢𝑟𝑒 = Legitimate 𝑒𝑙𝑠𝑒 𝑖𝑓 𝑈𝑅𝐿 𝑙𝑒𝑛𝑔𝑡ℎ ≥ 54 𝑎𝑛𝑑 ≤ 75 → 𝑓𝑒𝑎𝑡𝑢𝑟𝑒 =** 𝑆𝑢𝑠𝑝𝑖𝑐𝑖𝑜𝑢𝑠 𝑜𝑡ℎ𝑒𝑟𝑤𝑖𝑠𝑒 → 𝑓𝑒𝑎𝑡𝑢𝑟𝑒 = Phishing We have been able to update this feature rule by using a method based on frequency and thus improving upon its accuracy. 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. This is accomplished by means of an “HTTP Redirect” on a domain name that is short, which links to the webpage that has a long URL.

**Rule: IF{ TinyURL → Phishing Otherwise → Legitimate**

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

**Rule: IF { Url Having @ Symbol → Phishing Otherwise → Legitimate**

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

**Rule: IF { ThePosition of the Last Occurrence of "//" in the URL > 7 → Phishing Otherwise → Legitimate**

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.

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

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), which in our example is “uk”. The “ac” part is shorthand for “academic”, the combined “ac.uk” is called a second-level domain (SLD) and “hud” is the actual name of the domain. To produce a rule for extracting this feature, we firstly have to omit the (www.) from the URL which is in fact a sub domain in itself. Then, we have to remove the (ccTLD) if it exists. Finally, we count the remaining dots. If the number of dots is greater than one, then the URL is classified as “Suspicious” since it has one sub domain.

**Rule: IF { Dots In Domain Part = 1 → Legitimate Dots In Domain Part = 2 → Suspicious Otherwise → Phishing**

PageRank is a value ranging from “0” to “1”. PageRank aims to measure how important a webpage is on the Internet. The greater the PageRank value the more important the webpage. In our datasets, we find that about 95% of phishing webpages have no PageRank. Moreover, we find that the remaining 5% of phishing webpages may reach a PageRank value up to “0.2”.

**Rule: IF{ PageRank < 0.2 → Phishing Otherwise → Legitimate**

## **4.2. The dataset of phishing links**

id having\_IP\_Address URL\_Length Shortining\_Service having\_At\_Symbol \

0 1 -1 1 1 1

1 2 1 1 1 1

2 3 1 0 1 1

3 4 1 0 1 1

4 5 1 0 -1 1

5 6 -1 0 -1 1

6 7 1 0 -1 1

7 8 1 0 1 1

double\_slash\_redirecting Prefix\_Suffix having\_Sub\_Domain SSLfinal\_State \

0 -1 -1 -1 -1

1 1 -1 0 1

2 1 -1 -1 -1

3 1 -1 -1 -1

4 1 -1 1 1

5 -1 -1 1 1

6 1 -1 -1 -1

7 1 -1 -1 -1

Domain\_registeration\_length ... popUpWidnow Iframe age\_of\_domain \

0 -1 ... 1.0 1.0 -1.0

1 -1 ... 1.0 1.0 -1.0

2 -1 ... 1.0 1.0 1.0

3 1 ... 1.0 1.0 -1.0

4 -1 ... -1.0 1.0 -1.0

5 -1 ... 1.0 1.0 1.0

6 1 ... 1.0 1.0 1.0

7 1 ... 1.0 1.0 -1.0

DNSRecord web\_traffic Page\_Rank Google\_Index Links\_pointing\_to\_page \

0 -1.0 -1.0 -1.0 1.0 1.0

1 -1.0 0.0 -1.0 1.0 1.0

2 -1.0 1.0 -1.0 1.0 0.0

3 -1.0 1.0 -1.0 1.0 -1.0

4 -1.0 0.0 -1.0 1.0 1.0

5 1.0 1.0 -1.0 1.0 -1.0

6 -1.0 -1.0 -1.0 1.0 0.0

7 -1.0 0.0 -1.0 1.0 0.0

Statistical\_report Result

0 -1.0 -1.0

1 1.0 -1.0

2 -1.0 -1.0

3 1.0 -1.0

4 1.0 1.0

5 -1.0 1.0

6 -1.0 -1.0

7 1.0 -1.0

[8 rows x 32 columns]

**Figure 3.** DATASET OF PHISHING LINKS

[[-1. 1. 1. ... 1. 1. -1.]

[ 1. 1. 1. ... 1. 1. 1.]

[ 1. 0. 1. ... 1. 0. -1.]

...

[-1. -1. 1. ... -1. 0. 1.]

[ 1. -1. 1. ... 1. 0. 1.]

[ 1. -1. 1. ... nan nan nan]] [[-1.]

[-1.]

[-1.]

...

[ 1.]

[ 1.]

[nan]]

Removing the unwanted data

After using grid search technique by the help of ML algorithm the RandomForestClassifier and Linear regression helps to filter the links of website URLs or we can say the Phishing links.

import regex

from tldextract import extract

import ssl

import socket

from bs4 import BeautifulSoup

import urllib.request

import whos

import datetime

def url\_having\_ip(url):

#using regular function

# symbol = regex.findall(r'(http((s)?)://)((((\d)+).)\*)((\w)+)(/((\w)+))?',url)

# if(len(symbol)!=0):

# having\_ip = 1 #phishing

# else:

# having\_ip = -1 #legitimate

#return(having\_ip)

return 0

def url\_length(url):

length=len(url)

if(length<54):

return -1

elif(54<=length<=75):

return 0

else:

return 1

def url\_short(url):

#ongoing

return 0

def having\_at\_symbol(url):

symbol=regex.findall(r'@',url)

if(len(symbol)==0):

return -1

else:

return 1

def doubleSlash(url):

#ongoing

return 0

def prefix\_suffix(url):

subDomain, domain, suffix = extract(url)

if(domain.count('-')):

return 1

else:

return -1

def sub\_domain(url):

subDomain, domain, suffix = extract(url)

if(subDomain.count('.')==0):

return -1

elif(subDomain.count('.')==1):

return 0

else:

return 1

def SSLfinal\_State(url):

try:

#check wheather contains https

if(regex.search('^https',url)):

usehttps = 1

else:

usehttps = 0

#getting the certificate issuer to later compare with trusted issuer

#getting host name

subDomain, domain, suffix = extract(url)

host\_name = domain + "." + suffix

context = ssl.create\_default\_context()

sct = context.wrap\_socket(socket.socket(), server\_hostname = host\_name)

sct.connect((host\_name, 443))

certificate = sct.getpeercert()

issuer = dict(x[0] for x in certificate['issuer'])

certificate\_Auth = str(issuer['commonName'])

certificate\_Auth = certificate\_Auth.split()

if(certificate\_Auth[0] == "Network" or certificate\_Auth == "Deutsche"):

certificate\_Auth = certificate\_Auth[0] + " " + certificate\_Auth[1]

else:

certificate\_Auth = certificate\_Auth[0]

trusted\_Auth = ['Comodo','Symantec','GoDaddy','GlobalSign','DigiCert','StartCom','Entrust','Verizon','Trustwave','Unizeto','Buypass','QuoVadis','Deutsche Telekom','Network Solutions','SwissSign','IdenTrust','Secom','TWCA','GeoTrust','Thawte','Doster','VeriSign']

#getting age of certificate

startingDate = str(certificate['notBefore'])

endingDate = str(certificate['notAfter'])

startingYear = int(startingDate.split()[3])

endingYear = int(endingDate.split()[3])

Age\_of\_certificate = endingYear-startingYear

#checking final conditions

if((usehttps==1) and (certificate\_Auth in trusted\_Auth) and (Age\_of\_certificate>=1) ):

return -1 #legitimate

elif((usehttps==1) and (certificate\_Auth not in trusted\_Auth)):

return 0 #suspicious

else:

return 1 #phishing

except Exception as e:

return 1

def domain\_registration(url):

try:

w = whois.whois(url)

updated = w.updated\_date

exp = w.expiration\_date

length = (exp[0]-updated[0]).days

if(length<=365):

return 1

else:

return -1

except:

return 0

def favicon(url):

#ongoing

return 0

def port(url):

#ongoing

return 0

def https\_token(url):

subDomain, domain, suffix = extract(url)

host =subDomain +'.' + domain + '.' + suffix

if(host.count('https')): #attacker can trick by putting https in domain part

return 1

else:

return -1

def request\_url(url):

try:

subDomain, domain, suffix = extract(url)

websiteDomain = domain

opener = urllib.request.urlopen(url).read()

soup = BeautifulSoup(opener, 'lxml')

imgs = soup.findAll('img', src=True)

total = len(imgs)

linked\_to\_same = 0

avg =0

for image in imgs:

subDomain, domain, suffix = extract(image['src'])

imageDomain = domain

if(websiteDomain==imageDomain or imageDomain==''):

linked\_to\_same = linked\_to\_same + 1

vids = soup.findAll('video', src=True)

total = total + len(vids)

for video in vids:

subDomain, domain, suffix = extract(video['src'])

vidDomain = domain

if(websiteDomain==vidDomain or vidDomain==''):

linked\_to\_same = linked\_to\_same + 1

linked\_outside = total-linked\_to\_same

if(total!=0):

avg = linked\_outside/total

if(avg<0.22):

return -1

elif(0.22<=avg<=0.61):

return 0

else:

return 1

except:

return 0

def url\_of\_anchor(url):

try:

subDomain, domain, suffix = extract(url)

websiteDomain = domain

opener = urllib.request.urlopen(url).read()

soup = BeautifulSoup(opener, 'lxml')

anchors = soup.findAll('a', href=True)

total = len(anchors)

linked\_to\_same = 0

avg = 0

for anchor in anchors:

subDomain, domain, suffix = extract(anchor['href'])

anchorDomain = domain

if(websiteDomain==anchorDomain or anchorDomain==''):

linked\_to\_same = linked\_to\_same + 1

linked\_outside = total-linked\_to\_same

if(total!=0):

avg = linked\_outside/total

if(avg<0.31):

return -1

elif(0.31<=avg<=0.67):

return 0

else:

return 1

except:

return 0

def Links\_in\_tags(url):

try:

opener = urllib.request.urlopen(url).read()

soup = BeautifulSoup(opener, 'lxml')

no\_of\_meta =0

no\_of\_link =0

no\_of\_script =0

anchors=0

avg =0

for meta in soup.find\_all('meta'):

no\_of\_meta = no\_of\_meta+1

for link in soup.find\_all('link'):

no\_of\_link = no\_of\_link +1

for script in soup.find\_all('script'):

no\_of\_script = no\_of\_script+1

for anchor in soup.find\_all('a'):

anchors = anchors+1

total = no\_of\_meta + no\_of\_link + no\_of\_script+anchors

tags = no\_of\_meta + no\_of\_link + no\_of\_script

if(total!=0):

avg = tags/total

if(avg<0.25):

return -1

elif(0.25<=avg<=0.81):

return 0

else:

return 1

except:

return 0

def sfh(url):

#ongoing

return 0

def email\_submit(url):

try:

opener = urllib.request.urlopen(url).read()

soup = BeautifulSoup(opener, 'lxml')

if(soup.find('mailto:')):

return 1

else:

return -1

except:

return 0

def abnormal\_url(url):

#ongoing

return 0

def redirect(url):

#ongoing

return 0

def on\_mouseover(url):

#ongoing

return 0

def rightClick(url):

#ongoing

return 0

def popup(url):

#ongoing

return 0

def iframe(url):

#ongoing

return 0

def age\_of\_domain(url):

try:

w = whois.whois(url)

start\_date = w.creation\_date

current\_date = datetime.datetime.now()

age =(current\_date-start\_date[0]).days

if(age>=180):

return -1

else:

return 1

except Exception as e:

print(e)

return 0

def dns(url):

#ongoing

return 0

def web\_traffic(url):

#ongoing

return 0

def page\_rank(url):

#ongoing

return 0

def google\_index(url):

#ongoing

return 0

def links\_pointing(url):

#ongoing

return 0

def statistical(url):

#ongoing

return 0

def main(url):

check = [[url\_having\_ip(url),url\_length(url),url\_short(url),having\_at\_symbol(url),

doubleSlash(url),prefix\_suffix(url),sub\_domain(url),SSLfinal\_State(url),

domain\_registration(url),favicon(url),port(url),https\_token(url),request\_url(url),

url\_of\_anchor(url),Links\_in\_tags(url),sfh(url),email\_submit(url),abnormal\_url(url),

redirect(url),on\_mouseover(url),rightClick(url),popup(url),iframe(url),

age\_of\_domain(url),dns(url),web\_traffic(url),page\_rank(url),google\_index(url),

links\_pointing(url),statistical(url)]]

print(check)

return check

This program helps to give the input of url

#importing libraries

from sklearn.externals import joblib

import inputScript

#load the pickle file

classifier = joblib.load('final\_models/rf\_final.pkl')

#input url

print("enter url")

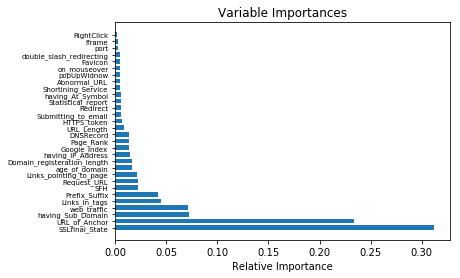
url = input()

#checking and predicting

checkprediction = inputScript.main(url)

prediction = classifier.predict(checkprediction)

print(prediction)

Fig4: PHISHING URLS.

**4.3 Build Neural network**

1. **Data preparation**: Collect and pre-process your data. This involves gathering a dataset that is relevant to the problem you want to solve and performing any necessary data cleaning, normalization, or feature engineering.
2. **Choose a neural network architecture**: Decide on the type of neural network architecture you want to use for your problem. Some common types include feed forward neural networks, convolution neural networks (CNNs) for image-related tasks, recurrent neural networks (RNNs) for sequential data, and more advanced architectures like transformers or GANs.
3. **Split the dataset**: Split your dataset into training, validation, and test sets. The training set is used to train the neural network, the validation set is used to tune hyper parameters and monitor the model's performance, and the test set is used to evaluate the final model's performance.
4. **Model Defining:** Define the structure of your neural network model. This involves specifying the number and type of layers, the activation functions, and any other architectural choices. Libraries like TensorFlow or PyTorch can be used to define and create the neural network model.
5. **Model Initialization:** Initialize the weights and biases of your neural network model. Random initialization is often used, but there are also other techniques available.
6. **Train the Model:** Train the neural network using the training set. This involves feeding the input data through the network, comparing the output with the desired output, and adjusting the weights and biases using an optimization algorithm such as gradient descent. This process is typically done in multiple iterations or epochs.
7. **Model Tuning:** Validate the model using the validation set and adjust hyper parameters as needed. Hyper parameters include learning rate, batch size, regularization techniques, and network architecture choices. This step helps optimize the model's performance and prevent over fitting.
8. **Model Evaluation:** Evaluate the final trained model using the test set, which provides an unbiased estimate of its performance. Calculate metrics such as accuracy, precision, recall, or mean squared error, depending on the nature of the problem.
9. **Predictions:** Now we have to predict the model with new data set

**This is an syntax to form a neural network**

import tensorflow as tf

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(64, activation='relu', input\_shape=(input\_dim,)),

tf.keras.layers.Dense(32, activation='relu'),

tf.keras.layers.Dense(output\_dim, activation='softmax')

])

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.summary()

The above code is used for more smooth ‘S -Curved’ other than the normal linear curved we are used geometrical problem to solved this kind of problems so we are used Gradient Decent technique or Adams technique where we can easily evaluate the model fitness curve by fit() .by this sigmoid function. where it helps to predict the best accuracy of the model.

The ‘Tensor’ means it is a matrix of 1D ,2D matrix it helps to form a list help to get the dataset. Which help to build the model. And also help to build the artificial intelligence based applications.

## **4.4. Tools used for implementation**

The Go Lab helps to implement the projects and other than we can use jupyter lab through Anaconda but it may arise some issue those who have not have sufficient memory (RAM , and GPU) so for that reason GO LAB helps to implement the whole project because we get the support of TPU which is more friendly other than Jupyter Lab

So we suggest to use implement this project in Go Lab. It help us to get more accurate prediction on minimum span of time . Keras api and Tenser flow helps to solve complex problem in less time.

## **5. Conclusion**

The project helps to filter the phishing links and the neurons are helps to detect the links and defend in the application the . Those preserving data will help to form a new dataset and according to those new data set the IoT will be performed and it will prevent the phishing links other than it will prevent the cyber threats or the ransom ware .

**REFERENCES**

[1] EMPLOYEE PERFORMANCE APPRAISAL SYSTEM USING FUZZY LOGIC Adnan Shaout and Mohamed Khalid Yousif,International Journal of Computer Science & Information Technology (IJCSIT) Vol 6, No 4, August 2014.

[2] Website https://www.activestate.com/blog/phishing-url-detection-with-python-and-ml/

[3] website https://www.tensorflow.org/