**Machine Learning and Concept Drift-Based Approach for Malicious Website Detection**

Assistant Professor, Mr.K.Steephen,

Department of CSE

¹P.Swetha, ²V.Varsha, ³T.Kaviya, ⁴S.Shruthi,

¹²³⁴Final Year Project Members, Bachelor of Engineering Department of Computer Science and Engineering

Vivekanandha College of Technology for Women, Tiruchengode, Tamil Nadu.

***ABSTRACT: Malicious websites pose a significant threat to internet users by exploiting vulnerabilities and engaging in malicious activities such as phishing, malware distribution, and fraud. Traditional approaches for detecting malicious websites often struggle to adapt to the dynamic nature of these threats, as they fail to account for concept drift— the phenomenon where the statistical properties of the data change over time. In this paper, we propose a novel machine learning framework that addresses the challenge of concept drift in malicious website detection. Our approach leverages adaptive learning algorithms capable of dynamically adjusting to evolving data distributions. We employ feature engineering techniques to extract relevant features from website content and network traffic data, enhancing the detection accuracy. Furthermore, we introduce a concept drift detection mechanism that continuously monitors model performance and triggers retraining when significant changes in data distribution are detected. We evaluate our approach on real-world datasets and demonstrate its effectiveness in detecting malicious websites under evolving conditions. Experimental results show that our proposed framework outperforms traditional static models, achieving higher accuracy and robustness against concept drift.***

***Keywords:*** *Malicious website detection, Machine learning, Concept drift, Adaptive learning, Feature engineering.*

1. **INTRODUCTION**

**DOMAIN:** Machine Learning.

Malicious websites continue to proliferate on the internet, posing a severe threat to cybersecurity. These websites are designed to deceive users, steal sensitive information, or compromise their devices for nefarious purposes. Traditional signature-based detection methods are inadequate in combating these threats, as they rely on known patterns and are easily evaded by sophisticated attackers. Moreover, the landscape of malicious websites constantly evolves, with attackers employing novel techniques to evade detection.

1. **OBJECTIVE**

Investigate the characteristics of concept drift in web traffic data and its impact on the performance of detection models. Design and implement an adaptive learning mechanism capable of detecting and responding to concept drift in real-time. Evaluate the effectiveness of the proposed approach in terms of detection accuracy, false positive rate, and adaptability to changing threat landscapes.

1. **RELATED WORK**

**[1]** [Doyen Sahoo](https://arxiv.org/search/cs?searchtype=author&query=Sahoo%2C+D), [Chenghao Liu](https://arxiv.org/search/cs?searchtype=author&query=Liu%2C+C), [Steven C.H. Hoi](https://arxiv.org/search/cs?searchtype=author&query=Hoi%2C+S+C)**. “MALICIOUS URL DETECTION USING MACHINE LEARNING”**

Malicious URL, a.k.a. Malicious website, is a not unusual place and extreme chance to cyber security. Malicious URLs host unsolicited content (spam, phishing, drive-via way of means of exploits, etc.) and trap unsuspecting customers to grow to be sufferers of scams (economic loss, robbery of personal information, and malware installation), and cause losses of billions of dollars every year. It is vital to hit upon and act on such threats in a well timed manner. Traditionally, this detection is accomplished by and large via using blacklists. However, blacklists can not be exhaustive, and absence the capacity to stumble on newly generated malicious URLs. To enhance the generality of malicious URL detectors, gadget getting to know strategies had been explored with growing interest in latest years. This article targets to offer a complete survey and a structural expertise of Malicious URL Detection strategies the use of system learning. We gift the formal components of Malicious URL Detection as a device mastering task, and categorize and assessment the contributions of literature research that addresses distinct dimensions of this problem (feature representation, algorithm design, etc.). Further, this newsletter gives a well timed and complete survey for a selection of various audiences, now no longer simplest for system gaining knowledge of researchers and engineers in academia, however additionally for experts and practitioners in cyber security industry, to help them understand the state of the art and facilitate their own research and practical applications. We additionally speak realistic problems in device design, open studies challenges, and factor out a few crucial guidelines for destiny studies

**[2]** Ripon Patgiri, Hemanth Katari, Ronit Kumar, Dheeraj Sharma **LEARNING** **EMPIRICAL STUDY ON MALICIOUS URL DETECTION USING MACHINE**

In this paper, the malicious URLs detection is handled as a binary class hassle and overall performance of numerous famous classifiers are examined with take a look at data. The algorithms Random Forests and guide Vector Machine (SVM) are studied mainly which obtain a excessive accuracy. These algorithms are used for schooling the dataset for class of properly and terrible URLs. The dataset of URLs is divided into training and test data in 60:40, 70:30 and 80:20 ratios. Accuracy of Random Forests and SVMs is calculated for numerous iterations for every cut up ratio. According to the results, the cut up ratio 80:20 is found as greater correct cut up and common accuracy of Random Forests is greater than SVMs. SVM is observed to be more fluctuating than Random Forests in accuracy.

**[3]** [Hung Le](https://arxiv.org/search/cs?searchtype=author&query=Le%2C+H), [Quang Pham](https://arxiv.org/search/cs?searchtype=author&query=Pham%2C+Q), [Doyen Sahoo](https://arxiv.org/search/cs?searchtype=author&query=Sahoo%2C+D), [Steven C.H. Hoi](https://arxiv.org/search/cs?searchtype=author&query=Hoi%2C+S+C) **LEARNING A URL REPRESENTATION WITH MACHINE LEARNING FOR MALICIOUS URL DETECTION**

Malicious URLs host unsolicited content material cand are used to perpetrate cybercrimes. It is vital to hit upon them in a well timed manner. Traditionally, that is performed via using blacklists, which can't be exhaustive, and can't locate newly generated malicious URLs. To cope with this, latest years have witnessed numerous efforts to carry out Malicious URL Detection the use of Machine Learning. The maximum famous and scalable procedures use lexical properties of the URL string through extracting Bag-of-phrases like features, observed through making use of system studying models which include SVMs. There also are different functions designed via way of means of specialists to enhance the prediction overall performance of the model. These processes be afflicted by numerous limitations: (i) Inability to correctly seize semantic which means and sequential styles in URL strings; (ii) Requiring vast guide characteristic engineering; and (iii) Inability to deal with unseen capabilities and generalize to check data. To deal with those challenges, we suggest URLNet, an give up-to-give up device mastering framework to examine a nonlinear URL embedding for Malicious URL Detection without delay from the URL. Specifically, we exercise Convolutional Neural Networks to every characters and terms of the URL String to investigate the URL embedding in a on the identical time optimized framework. This method permits the version to seize numerous varieties of semantic information, which became now no longer feasible through the prevailing models. We additionally endorse superior word-embedding’s to remedy the trouble of too many rare words discovered on this task. We conduct good sized experiments on a large-scale dataset and display a large overall performance advantage over current methods. We additionally conduct ablation research to assess the overall performance of diverse additives of URLNet.

**[4]**[Baojiang Cui](https://www.inderscienceonline.com/author/Cui%2C+Baojiang) **MALICIOUS URL DETECTION WITH FEATURE EXTRACTION BASED ON MACHINE LEARNING**

Many web applications suffer from various web attacks due to the lack of awareness concerning security. Therefore, it is necessary to improve the reliability of web applications by accurately detecting malicious URLs. In previous studies, keyword matching has always been used to detect malicious URLs, but this method is not adaptive. In this paper, statistical analyses based on gradient learning and feature extraction using a sigmoidal threshold level are combined to propose a new detection approach based on machine learning techniques. Moreover, the naïve Bayes, decision tree and SVM classifiers are used to validate the accuracy and efficiency of this method. Finally, the experimental results demonstrate that this method has a good detection performance, with an accuracy rate above 98.7%. In practical use, this system has been deployed online and is being used in large-scale detection, analyzing approximately 2 TB of data every day.

[5]Aaron Blum, Brad Wardman **LEXICAL FEATURE BASED PHISHING URL DETECTION USING ONLINE LEARNING**

Phishing is a shape of cybercrime wherein spammed emails and fraudulent web sites lure sufferers to offer sensitive data to the phishers. The received sensitive facts is sooner or later used to thieve identities or benefit get admission to to money. This paper explores the opportunity of using self assurance weighted category mixed with content material primarily based totally phishing URL detection to provide a dynamic and extensible gadget for detection of gift and emerging types of phishing domains. Our gadget is able to detecting rising threats as they seem and finally can offer elevated safety towards 0 hour threats not like conventional blacklisting strategies which feature reactively.

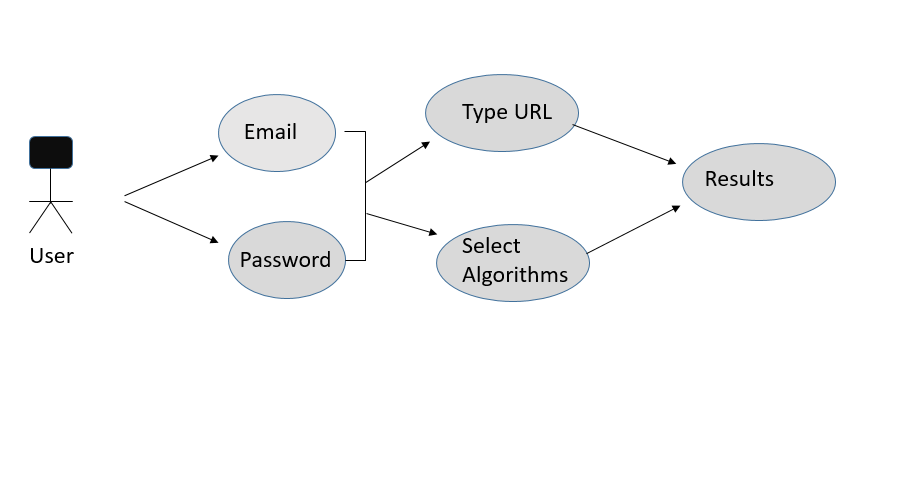
**[6]**[Rajesh Kumar](https://ieeexplore.ieee.org/author/37086345992); [Xiaosong Zhang](https://ieeexplore.ieee.org/author/37598091300); [Hussain Ahmad Tariq](https://ieeexplore.ieee.org/author/37086345969); [Riaz Ullah Khan](https://ieeexplore.ieee.org/author/37086349035) **MALICIOUS URL DETECTION USING MULTI-LAYER FILTERING MODEL**

Malicious URLs are dangerous to each thing of pc users. Detecting of the malicious URL could be very important. Currently, detection of malicious webpages strategies consists of black-listing and white-listing technique and system mastering category algorithms are used. However, the black-listing and white-listing generation is vain if a specific URL isn't always in listing. In this paper, we endorse a multi-layer version for detecting malicious URL. The clear out out can without delay decide the URL through training the edge of every layer clear out out while it reaches the edge. Otherwise, the clear out out leaves the URL to subsequent layer. We extensively utilized an instance to affirm that the version can enhance the accuracy of URL detection.

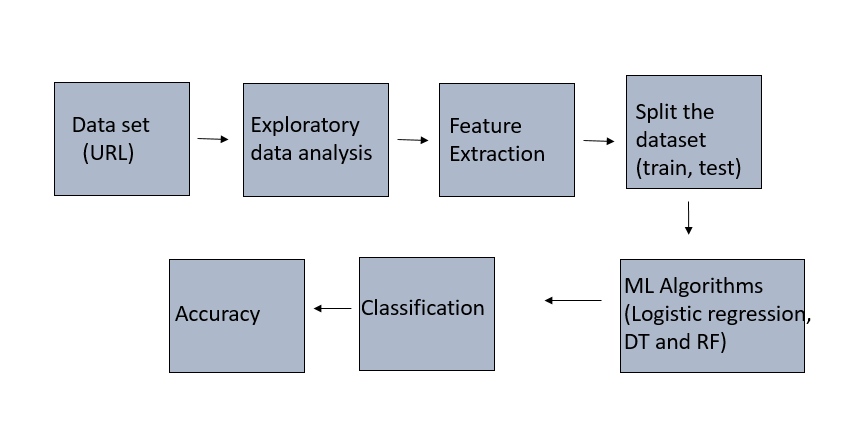
1. **PROPOSED APPROACH**

**4.1. System Component:**

The system for malicious website detection integrates data collection from various sources like web traffic logs and domain registries. After preprocessing, relevant features are extracted, encompassing URL characteristics and other metadata. Machine learning algorithms are then employed to train models for detection. Continuous monitoring of model performance helps detect concept drift, triggering retraining when significant changes occur in data distribution. The system evaluates model performance using metrics like accuracy and precision. Once deployed, it continuously learns from new data to adapt to evolving threats, while also providing reporting and visualization capabilities for easy interpretation of results. Integration and scalability ensure the system seamlessly fits into existing security infrastructure and efficiently handles large volumes of data.



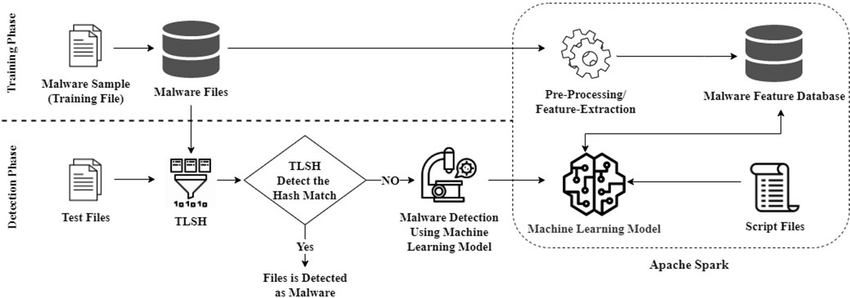
**Fig 4.1 Front End Module Diagram**



**Fig 4.2 Back End Module Diagram**

**4.2. Application Model:**

The application model for malicious website detection follows a systematic approach: data collection from diverse sources, preprocessing to refine and organize the data, feature extraction to identify relevant characteristics, model training using machine learning algorithms, continuous monitoring for concept drift, evaluation of model performance, deployment for real-time detection, continuous learning to adapt to new threats, and reporting/visualization for insights. Integration and scalability are prioritized to seamlessly fit within existing security infrastructure and handle increasing data loads efficiently.



**Fig 4.3 Architecture Diagram**

**4.3. Component Operation:**

Each component of the system operates cohesively to achieve efficient malicious website detection. Data collection sources from web traffic logs and domain registries, while preprocessing ensures data cleanliness and organization. Feature engineering extracts relevant attributes like URL characteristics. Through model training, machine learning algorithms are employed to create effective detection models. Concept drift detection monitors model performance for changes in data distribution, triggering retraining when necessary. Model evaluation assesses effectiveness using metrics like accuracy. Once deployed, the system continuously learns from new data to adapt to emerging threats. Reporting and visualization tools provide insights in a user-friendly format. Integration and scalability ensure seamless incorporation into existing infrastructure and effective handling of increased data volumes.

1. **PROJECT WORK**

**5.1. Implemented Design:**

The implemented design for malicious website detection employs a modular architecture for robust functionality. Each module, from data collection to continuous learning, operates seamlessly to ensure efficient detection. Leveraging machine learning algorithms and feature engineering techniques, the system adapts to evolving threats. Concept drift detection mechanisms trigger model retraining, maintaining effectiveness over time. Real-time deployment integrates with existing infrastructure, while reporting and visualization tools provide insights for informed decision-making. This design prioritizes scalability and integration, facilitating seamless incorporation into diverse security ecosystems while effectively handling growing data demands.

**5.2. Validation Phases**

**5.2.1. Unit Test:**

Unit tests for the malicious website detection system verify the functionality of individual modules. Each test case isolates a specific component, assessing its performance against expected outcomes. For example, data collection tests verify data retrieval accuracy, preprocessing tests confirm data cleanliness, and feature engineering tests validate feature extraction correctness. Model training tests assess the accuracy of machine learning algorithms, while concept drift detection tests ensure timely retraining triggers. Evaluation tests measure model performance metrics like accuracy and precision. Deployment tests validate seamless integration with existing infrastructure, and continuous learning tests verify adaptation to new data. These unit tests collectively ensure the reliability and effectiveness of the system's individual components.

**5.2.2. Integrity Checks:**

Integrity checks are essential to ensure the trustworthiness and reliability of the malicious website detection system. These checks involve verifying the consistency and accuracy of data throughout its lifecycle, from collection to analysis. Techniques such as cryptographic hashing can be employed to detect any unauthorized alterations or tampering of data. Additionally, checksums or digital signatures can be utilized to authenticate the integrity of software components and updates. Regular audits and monitoring mechanisms further bolster integrity assurance, helping to identify and mitigate any potential vulnerabilities or breaches promptly. Overall, integrity checks play a crucial role in maintaining the security and effectiveness of the system, safeguarding against data corruption and unauthorized modifications.

* 1. **Code Used:**

**Backend:**

#list of useful imports that  I will use

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import plotly.graph\_objects as go

from tensorflow.python.util import deprecation

from urllib.parse import urlparse

%matplotlib inline

import os

import numpy as np

import seaborn as sns

import random

import pickle

from sklearn.metrics import confusion\_matrix

from sklearn import metrics

from sklearn.metrics import roc\_curve, accuracy\_score,log\_loss,roc\_auc\_score

from nltk.stem.porter import PorterStemmer

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import warnings

warnings.filterwarnings("ignore")

pip install tldextract

Requirement already satisfied: tldextract in c:\users\st-0008\anaconda3\lib\site-packages (3.2.0)

Requirement already satisfied: requests>=2.1.0 in c:\users\st-0008\anaconda3\lib\site-packages (from tldextract) (2.28.1)

Requirement already satisfied: requests-file>=1.4 in c:\users\st-0008\anaconda3\lib\site-packages (from tldextract) (1.5.1)

Requirement already satisfied: idna in c:\users\st-0008\anaconda3\lib\site-packages (from tldextract) (3.3)

Requirement already satisfied: filelock>=3.0.8 in c:\users\st-0008\anaconda3\lib\site-packages (from tldextract) (3.6.0)

Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\st-0008\anaconda3\lib\site-packages (from requests>=2.1.0->tldextract) (1.26.11)

Requirement already satisfied: certifi>=2017.4.17 in c:\users\st-0008\anaconda3\lib\site-packages (from requests>=2.1.0->tldextract) (2022.9.24)

Requirement already satisfied: charset-normalizer<3,>=2 in c:\users\st-0008\anaconda3\lib\site-packages (from requests>=2.1.0->tldextract) (2.0.4)

Requirement already satisfied: six in c:\users\st-0008\anaconda3\lib\site-packages (from requests-file>=1.4->tldextract) (1.15.0)

Note: you may need to restart the kernel to use updated packages.

WARNING: Ignoring invalid distribution -umpy (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -rotobuf (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution - (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -umpy (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -rotobuf (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution - (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -umpy (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -rotobuf (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution - (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -umpy (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -rotobuf (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution - (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -umpy (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -rotobuf (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution - (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -umpy (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -rotobuf (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution - (c:\users\st-0008\anaconda3\lib\site-packages)

WARNING: Ignoring invalid distribution -ensorflow-intel (c:\users\st-0008\anaconda3\lib\site-packages)

import tldextract

# Load the data using pandas

urldata = pd.read\_csv(r'C:\Users\ST-0008\Desktop\ITML12\_Malicious\_detection(NA)\DATASET\malicious\_website\_detection\data.csv')

urldata.head(5)

urldata.shape

urldata.info()

#Checking Missing Values

urldata.isnull().sum()

urldata['label'].value\_counts()

fig = go.Figure([go.Pie(labels=['good', 'bad'], values=[urldata['label'].value\_counts()[0], urldata['label'].value\_counts()[1]])])

fig.update\_layout(title='Percentage of data points for ecah class')

fig.show()

!pip install tld

#Importing dependencies

from urllib.parse import urlparse

from tld import get\_tld

import os.path

#Length of URL

urldata['url\_length'] = urldata['url'].apply(lambda i: len(str(i)))

#Hostname Length

urldata['hostname\_length'] = urldata['url'].apply(lambda i: len(urlparse(i).netloc))

#Path Length

urldata['path\_length'] = urldata['url'].apply(lambda i: len(urlparse(i).path))

urldata.head()

#First Directory Length

def fd\_length(url):

    urlpath= urlparse(url).path

    try:

        return len(urlpath.split('/')[1])

    except:

        return 0

urldata['fd\_length'] = urldata['url'].apply(lambda i: fd\_length(i))

urldata.head()

#Length of Top Level Domain

urldata['tld'] = urldata['url'].apply(lambda i: get\_tld(i,fail\_silently=True))

def tld\_length(tld):

    try:

        return len(tld)

    except:

        return -1

urldata['tld\_length'] = urldata['tld'].apply(lambda i: tld\_length(i))

urldata.head()

urldata['tld'].value\_counts()

len(urldata['tld']== None)

urldata['tld\_length'].value\_counts()

urldata = urldata.drop("tld",1)

urldata = urldata.drop("tld",1)

#Dataset after extracting length features

urldata.head()

urldata['count-'] = urldata['url'].apply(lambda i: i.count('-'))

urldata['count@'] = urldata['url'].apply(lambda i: i.count('@'))

urldata['count?'] = urldata['url'].apply(lambda i: i.count('?'))

urldata['count%'] = urldata['url'].apply(lambda i: i.count('%'))

urldata['count.'] = urldata['url'].apply(lambda i: i.count('.'))

urldata['count='] = urldata['url'].apply(lambda i: i.count('='))

urldata['count-http'] = urldata['url'].apply(lambda i : i.count('http'))

urldata['count-https'] = urldata['url'].apply(lambda i : i.count('https'))

urldata['count-www'] = urldata['url'].apply(lambda i: i.count('www'))

def digit\_count(url):

    digits = 0

    for i in url:

        if i.isnumeric():

            digits = digits + 1

    return digits

urldata['count-digits']= urldata['url'].apply(lambda i: digit\_count(i))

def letter\_count(url):

    letters = 0

    for i in url:

        if i.isalpha():

            letters = letters + 1

    return letters

urldata['count-letters']= urldata['url'].apply(lambda i: letter\_count(i))

def no\_of\_dir(url):

    urldir = urlparse(url).path

    return urldir.count('/')

urldata['count\_dir'] = urldata['url'].apply(lambda i: no\_of\_dir(i))

# Data after extracting Count Features

urldata.head()

import re

#Use of IP or not in domain

def having\_ip\_address(url):

    match = re.search(

        '(([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\.'

        '([01]?\\d\\d?|2[0-4]\\d|25[0-5])\\/)|'  # IPv4

        '((0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\.(0x[0-9a-fA-F]{1,2})\\/)' # IPv4 in hexadecimal

        '(?:[a-fA-F0-9]{1,4}:){7}[a-fA-F0-9]{1,4}', url)  # Ipv6

    if match:

        # print match.group()

        return -1

    else:

        # print 'No matching pattern found'

        return 1

urldata['use\_of\_ip'] = urldata['url'].apply(lambda i: having\_ip\_address(i))

def shortening\_service(url):

    match = re.search('bit\.ly|goo\.gl|shorte\.st|go2l\.ink|x\.co|ow\.ly|t\.co|tinyurl|tr\.im|is\.gd|cli\.gs|'

                      'yfrog\.com|migre\.me|ff\.im|tiny\.cc|url4\.eu|twit\.ac|su\.pr|twurl\.nl|snipurl\.com|'

                      'short\.to|BudURL\.com|ping\.fm|post\.ly|Just\.as|bkite\.com|snipr\.com|fic\.kr|loopt\.us|'

                      'doiop\.com|short\.ie|kl\.am|wp\.me|rubyurl\.com|om\.ly|to\.ly|bit\.do|t\.co|lnkd\.in|'

                      'db\.tt|qr\.ae|adf\.ly|goo\.gl|bitly\.com|cur\.lv|tinyurl\.com|ow\.ly|bit\.ly|ity\.im|'

                      'q\.gs|is\.gd|po\.st|bc\.vc|twitthis\.com|u\.to|j\.mp|buzurl\.com|cutt\.us|u\.bb|yourls\.org|'

                      'x\.co|prettylinkpro\.com|scrnch\.me|filoops\.info|vzturl\.com|qr\.net|1url\.com|tweez\.me|v\.gd|'

                      'tr\.im|link\.zip\.net',

                      url)

    if match:

        return -1

    else:

        return 1

urldata['short\_url'] = urldata['url'].apply(lambda i: shortening\_service(i))

#Data after extracting Binary Features

urldata.head()

urldata['label'].value\_counts()[0]

# Upsampling

from sklearn.utils import resample

df\_majority = urldata[urldata.label == 'good']

df\_minarity = urldata[urldata.label == 'bad']

df\_upsample = resample(df\_minarity, n\_samples = 100000, replace = True)

df\_downsample = resample(df\_majority, n\_samples = 100000, replace = False)

urldata = pd.concat([df\_upsample,df\_downsample],axis =0)

urldata.label.value\_counts()

# shuffle the DataFrame rows

urldata = urldata.sample(frac = 1)

fig = go.Figure([go.Pie(labels=['good', 'bad'], values=[urldata['label'].value\_counts()[0], urldata['label'].value\_counts()[1]])])

fig.update\_layout(title='Percentage of data points for ecah class after upsampling')

fig.show()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

urldata['label'] = le.fit\_transform(urldata['label'])

#Heatmap

corrmat = urldata.corr()

f, ax = plt.subplots(figsize=(25,19))

sns.heatmap(corrmat, square=True, annot = True, annot\_kws={'size':10})

plt.figure(figsize=(15,5))

sns.countplot(x='label',data=urldata)

plt.title("Count Of URLs",fontsize=20)

plt.xlabel("Type Of URLs",fontsize=18)

plt.ylabel("Number Of URLs",fontsize=18)

print("Percent Of Bad URLs:{:.2f} %".format(len(urldata[urldata['label']=='bad'])/len(urldata['label'])\*100))

print("Percent Of Good URLs:{:.2f} %".format(len(urldata[urldata['label']=='good'])/len(urldata['label'])\*100))

plt.figure(figsize=(20,5))

plt.hist(urldata['url\_length'],bins=50,color='LightBlue')

plt.title("URL-Length",fontsize=20)

plt.xlabel("Url-Length",fontsize=18)

plt.ylabel("Number Of Urls",fontsize=18)

plt.figure(figsize=(20,5))

plt.hist(urldata['hostname\_length'],bins=50,color='Lightgreen')

plt.title("Hostname-Length",fontsize=20)

plt.xlabel("Length Of Hostname",fontsize=18)

plt.ylabel("Number Of Urls",fontsize=18)

plt.ylim(0,500)

plt.figure(figsize=(15,5))

plt.title("Number Of Directories In Url",fontsize=20)

sns.countplot(x='count\_dir',data=urldata)

plt.xlabel("Number Of Directories",fontsize=18)

plt.ylabel("Number Of URLs",fontsize=18)

plt.figure(figsize=(15,5))

plt.title("Number Of Directories In Url",fontsize=20)

sns.countplot(x='count\_dir',data=urldata,hue='label')

plt.xlabel("Number Of Directories",fontsize=18)

plt.ylabel("Number Of URLs",fontsize=18)

plt.figure(figsize=(15,5))

plt.title("Use Of IP In Url",fontsize=20)

plt.xlabel("Use Of IP",fontsize=18)

**5.4 Steps For Implementation :**

step1: Open command prompt

step2: cd C:\Users\gunan\Desktop\Malicious\_url - Copy\FINAL CODE\FINAL CODE\FRONT END\Malicious\new\_project

step3: python manage.py runserver

Link will be generated

step4: Copy and paste the link: http://127.0.0.1:8000/

Hence the application ::::::::::::::::::::::::::::

Use Logistic regression for better performance

Sample URLs:

------------

Bad:

----

campfireusa-emass.org/

espdesign.com.au

Good:

-----

https://chat.openai.com/

www.google.com

www.linked.com

1. **FEATURE ENGINEERING**

Feature engineering plays a crucial role in capturing the characteristics of malicious websites. We extract features from various sources, including website content, HTML structure, HTTP headers, and network traffic patterns. These features are carefully selected to capture both static and dynamic aspects of malicious behavior.

1. **CONCLUSION**

In conclusion, this project introduces an advanced system for malicious URL detection, overcoming the limitations of simplistic feature sets and a singular algorithmic approach. By incorporating diverse feature extraction methods and exploring multiple machine learning models, the system aims to enhance accuracy and adaptability to emerging cyber threats. The emphasis on continuous evaluation and refinement ensures the system's resilience in dynamic cybersecurity landscapes. Ultimately, this proactive defense mechanism seeks to mitigate the impact of scams, safeguard user information, and contribute to a more secure digital environment in the face of evolving malicious URL threats.

1. **REFERENCE**

[1] Alsharnouby, M.; Alaca, F.; Chiasson, S.; Why phishing still works: User strategies for combating phishing attacks, International Journal of Human-Computer Studies, Volume 82, pp. 69-82, ISSN1071 - 5819, <https://doi.org/10.1016/j.ijhcs.2015.05.005.(2015)>

[2]Ataque Homografico, https://www.techtudo.com.br/noticias/2017/11/ataque-homograficotruque-na-url-enganausuarios-com-paginas-falsas.ghtml, last accessed 30/11/2018.

[3]Internet Security Threat Report, Vol.23, Symantec.

[4]Protocols de Rede, <https://www.weblink.com.br/blog/tecnologia/conheca-os-principaisprotocolos-de-internet/>

last accessed 24/11/2018.

[5] Richards, K.; LaSalle, R.; INSIGHTS ON THE SECURITY INVESTMENTS THAT MAKE A DIFFERENCE, COST OF CYBER CRIME STUDY, Accenture. (2017)

[6] Rossouw, S.; Niekerk, J.;From information security to cyber security, Computers & Security, vol. 38, pp. 97- 102, ISSN 0167-4048, <https://doi.org/10.1016/j.cose.2013.04.004.(2013)>

[7] Sahoo, D.; Liu, C.; Steven C. H.; Malicious URL Detection using Machine Learning: A Survey, eprint arXiv:1701.07179. (2017)

[8] Wang, W.; Lu, Z.: Cyber security in the Smart Grid: Survey and Challenges. Computer Networks, vol. 57:5, pp. 1344-1371. ISSN 1389-1286,https://doi.org/10.1016/j.comnet.2012.12.017. (2013)

[9] What is URL? Definition from Techopedia, https://www.techopedia.com/definition/1352/uniform-resourcelocator-url, last accessed 22/11/2018.

[10]What is URL?, https://searchnetworking.techtarget.com/definition/URL, last accessed 22/11/2018.

[11]What is a Drive-by Download, https://www.kaspersky.com/resource-center/definitions/drive-by-download, last acessed 30/11/2018.

[12]What is Social Engineering?, https://searchsecurity.techtarget.com/definition/social-engineering, last acessed 30/11/2018.

[13]What is Social Engineering?, httPs://www.incapsula.com/web-application-security/socialengineeringattack.html, last acessed 1/12/2018.

[14] What’s the difference between a site and a URL?, https://www.lifewire.com/what-is-a-url2626035, last accessed 24/11/2018.

[15] Uma M., Padmavathi G.: A Survey on Various Cyber Attacks and Their Classification, International Journal of Network Security, vol.15:5, PP.390-396. (2013)