PROJECT REPORT

WEB PHISHING DETECTION

TEAM ID: PNT2022TMID49682

TEAM LEADER: S.MUTHULAKSHMI
TEAM MEMBER 1:M.RAHINI
TEAM MEMBER 2:A.ANTONY PRIYA

Project Report Format

CHAPTER I

I INTRODUCTION

Phishing has become a major source of concern for security professionals in recent years since it is relatively easy to develop a phoney website that appears to be identical to a legitimate website. Although experts can detect bogus websites, not all users can, and as a result, they become victims of phishing attacks. The attacker's main goal is to steal bank account credentials. Because of a lack of user awareness, phishing assaults are becoming more successful. Because phishing attacks take advantage of user flaws, it is difficult to mitigate them, but it is critical to improve phishing detection tools. Phishing is a type of wide extortion in which a malicious website imitates a genuine one-time memory with the sole purpose of obtaining sensitive data, such as passwords, account details, or MasterCard numbers. Despite the fact that there are still some anti-phishing programming and strategies for detecting possible phishing attempts in messages and typical phishing

content on websites, phishes devise fresh and crossbred procedures to get around public programming and frameworks. Phishing is a type of fraud that combines social engineering with access to sensitive and personal data, such as passwords and open-end credit unpretentious components by assuming the characteristics of a trustworthy person or business via electronic correspondence. Hacking uses spoof messages that appear legitimate and are instructed to originate from legitimate sources such as financial institutions, online business goals, and so on, to entice users to visit phoney destinations via links provided on phishing websites.

CHAPTER 2

2. LITERATURE SURVEY

Huang et al., (2009) proposed frameworks to distinguish phishing using page section similitude, which breaks down universal resource locator tokens to create forecast precision phishing pages typically keep their CSS look similar to their target pages. This strategy was suggested by Marchal et al., (2017) to differentiate The analysis of authentic site server log knowledge is required for phishing websites. An off-the-shelf programme or the detection of a phishing website. Free has a number of distinguishing characteristics, including high precision, complete autonomy, and beautiful language-freedom, speed of choosing, flexibility to dynamic phishing, and flexibility to advance in phishing methods.

By extracting website URL features and analysing subset based feature selection methods, Mustafa Aydin et al. suggested a classification algorithm for phishing website identification. For the detection of phishing websites, it uses feature extraction and selection approaches.

Alphanumeric Character Analysis, Keyword Analysis, Security Analysis, Domain Identity Analysis, and Rank Based Analysis are five separate analyses of the retrieved features about the URLs of the sites and the built feature matrix. The majority of these elements are textual properties of the URL, with others relying on third-party services. PhishStorm is an automated phishing detection system developed by Samuel Marchal et al. that can analyse any URL in real time to identify probable phishing sites. To protect consumers from phishing content, Phish Storm is presented as an automatic real-time URL phishingness evaluation system. PhishStorm can be used as a Website reputation evaluation system that delivers a phishingness score for URLs.

Fadi Thabtah et al. compared a huge variety of machine learning approaches on real phishing datasets using various metrics. The goal of this comparison is to highlight the benefits and drawbacks of machine learning predictive models, as well as their real effectiveness when it comes to phishing assaults. Covering approach models are more appropriate as antiphishing solutions, according to the experimental data.

Muhemmet Baykara et al. proposed the Anti Phishing Simulator, which provides information about the phishing detection challenge and how to detect phishing emails. The Bayesian algorithm adds spam emails to the database. Phishing attackers utilise JavaScript to insert a valid URL into the address bar of the browser. The study recommends using the e-mail text as a keyword only for advanced word processing.

2.1 Existing system

To detect phishing sites, the existing system employs Classifiers, Fusion Algorithms, and Bayesian Models. Text and visual material can be classified by the classifiers. Text classifiers are used to categorise text material, whereas Image classifiers are used to categorise image content. The threshold value is calculated using a Bayesian model. The Fusion Algorithm uses the results of both classifiers to determine whether or not the site is phishing. Correct classification ratio, F-score, Matthews' correlation coefficient, False negative ratio, and False alarm ratio are used to evaluate the performance of different classifiers.

2.2 REFERENCES

1.M. Korkmaz, O. K. Sahingoz and B. Diri, "Detection of Phishing Websites by Using Machine Learning-Based URL Analysis", (2020 11th International Conference on Computing Communication and Networking Technologies ICCCNT), pp. 1-7, 2020. Show in Context View Article Google Scholar
2.Phishing Activity Trends Report Summary - 3rd Quarter 2020, 2020, [online] Available:

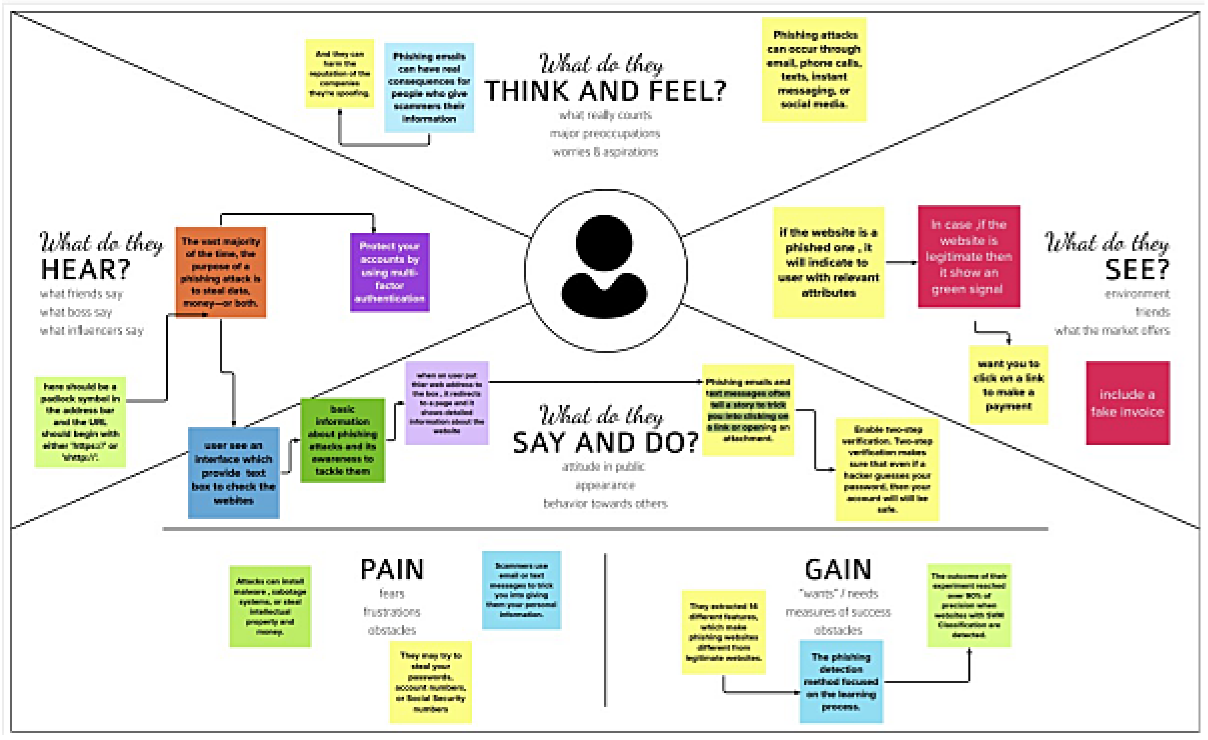
https://apwg.org/trendsreports/ Show in Context Google Scholar
3.A. Das, S. Baki, A. El Aassal, R. Verma and A. Dunbar, "SoK: A Comprehensive Reexamination of Phishing Research From the Security Perspective", IEEE Communications Surveys Tutorials, vol. 22, no. 1, pp. 671-708, First quarter 2020.

2.3 Problem Statement Definition

This section explains the suggested phishing attack detection model. The suggested methodology focuses on detecting phishing attacks using the properties of phishing websites, the Blacklist, and the WHOIS database. Few criteria can be utilised to distinguish between real and faked web pages, according to experts. URLs, domain identification, security & encryption, source code, page style and contents, web address bar, and social human component are only a few of the features that have been chosen. This research is limited to URLs and domain name characteristics. . These characteristics are examined using a set of rules in order to distinguish phishing webpage URLs from authentic website URL

3. IDEATION & PROPOSED SOLUTION

3.1 Empthy map canvas



And they can harm the reputation of the companies they're spoofing.

Phishing emails can have real consequences for people who give scammers their information

Phishing attacks can occur through email, phone calls, texts, instant messaging, or social media.

The vast majority of the time, the purpose of a phishing attack is to steal data, money—or both.

Protect your accounts by using multi-factor authentication

here should be a padlock symbol in the address bar and the URL should begin with either "https://" or "shttp://".

user see an interface which provide text box to check the webites

basic information about phishing attacks and its awareness to tackle them

when an user put thier web address to the box , it redirects to a page and it shows detailed information about the website

Phishing emails and text messages often tell a story to trick you into clicking on a link or opening an attachment.

Enable two-step verification. Two-step verification makes sure that even if a hacker guesses your password, then your account will still be safe.

if the website is a phished one , it will indicate to user with relevant attributes

In case ,if the website is legitimate then it show an green signal

want you to click on a link to make a payment

include a fake invoice

Attacks can install malware , sabotage systems, or steal intellectual property and money.

Scammers use email or text messages to trick you into giving them your personal information.

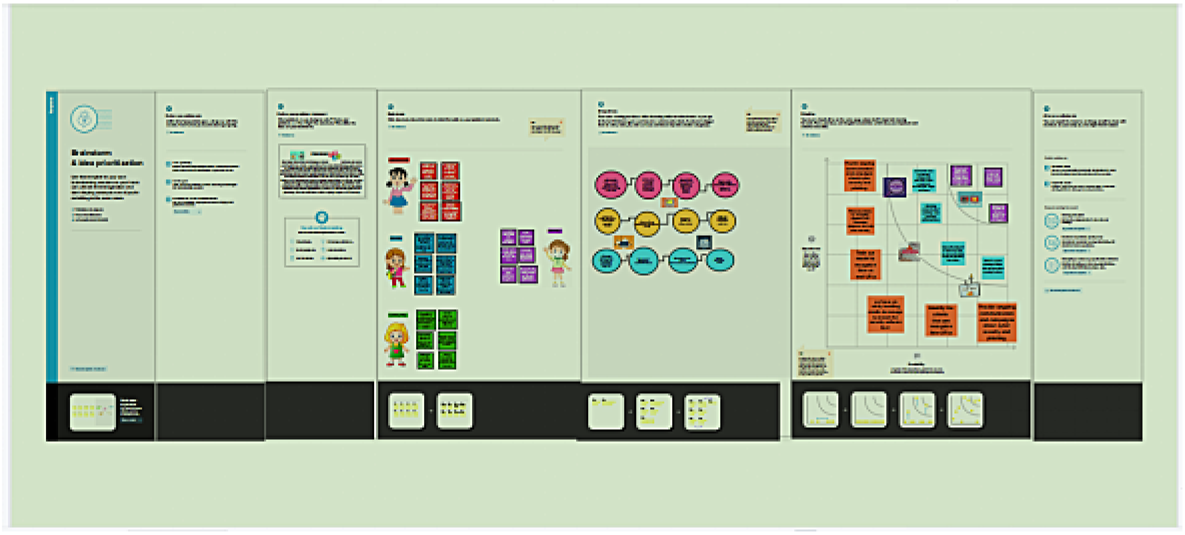
They may try to steal your passwords, account numbers, or Social Security numbers

They extracted 16 different features, which make phishing websites different from legitimate websites.

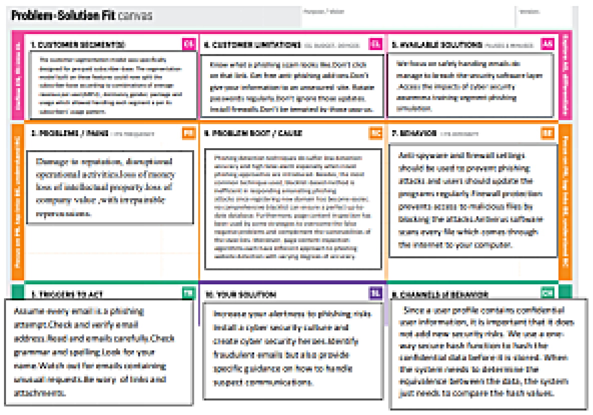
The phishing detection method focused on the learning process.

The outcome of their experiment reached over 80% of precision when websites with SVM Classification are detected.

3.2 Ideation & Brainstorming



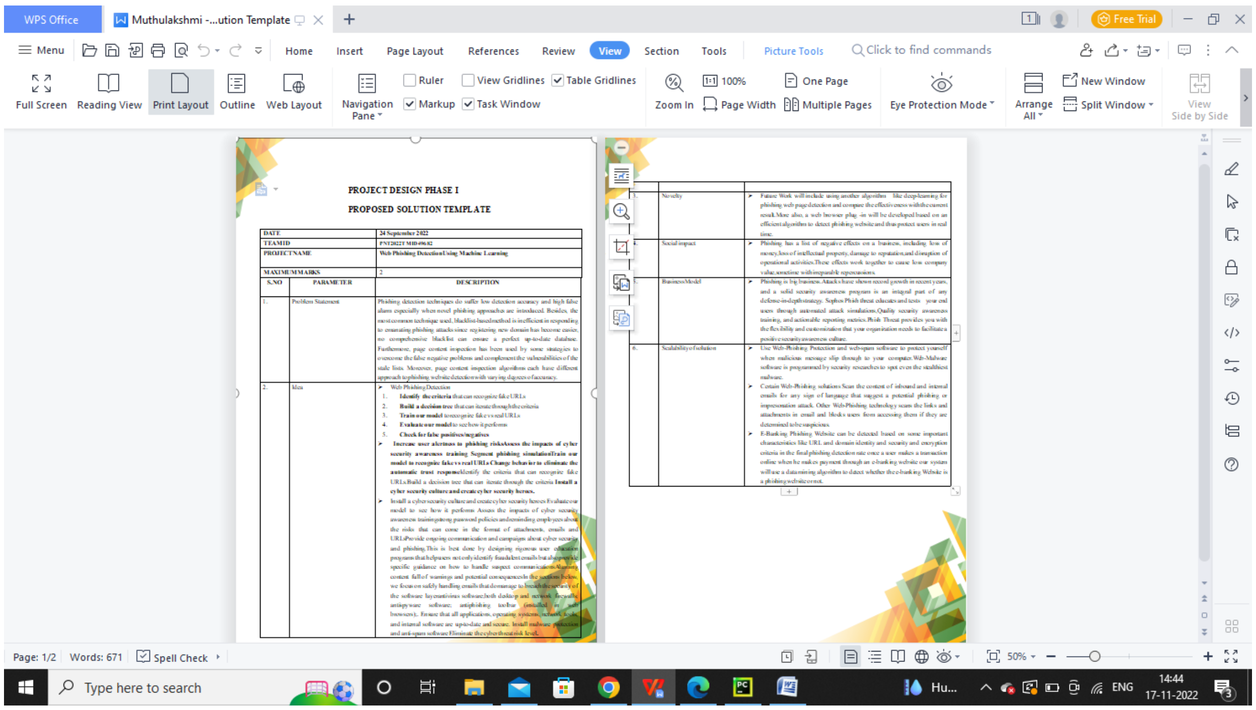
3.3 Proposed Solution



Problem-Solution Fit canvas

1. CUSTOMER SEGMENT(S) CS The customer segmentation model uses specifically designed for pre-paid subscriber base. The segmentation model built on these features could now split the subscriber base according to combinations of average revenue per user(ARPU), tenure, gender, package and usage which allowed handling and segment a per to subscribers' usage pattern.	**6. CUSTOMER LIMITATIONS** CC Know what a phishing scam looks like.Don't click on that link. Get free anti-phishing add-ons.Don't give your information to an unsecured site. Rotate passwords regularly.Don't ignore those updates. Install firewalls.Don't be tempted by those pop-ups.	**5. AVAILABLE SOLUTIONS** AS We focus on safety handling emails do manage to breach the security software layer .Access the impacts of cyber security awareness training segment phishing simulation.
2. PROBLEMS / PAINS PR Damage to reputation, disruptional operational activities,loss of money loss of intellectual property,loss of company value ,with irreparable repercussions.	**9. PROBLEM ROOT / CAUSE** RC Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced. Besides, the most common technique used, blacklist-based method is inefficient in responding emanating phishing attacks since registering new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database. Furthermore, page content inspection has been used by some strategies to overcome the false negative problems and complement the vulnerabilities of the stale lists. Moreover, page content inspection algorithms each have different approach to phishing website detection with varying degrees of accuracy.	**7. BEHAVIOR** BE Anti-spyware and firewall settings should be used to prevent phishing attacks and users should update the programs regularly.Firewall protection prevents access to malicious files by blocking the attacks.Antivirus software scans every file which comes through the internet to your computer.
3. TRIGGERS TO ACT TR Assume every email is a phishing attempt.Check and verify email address.Read and emails carefully.Check grammar and spelling.Look for your name.Watch out for emails containing unusual requests.Be wary of links and attachments.	**10. YOUR SOLUTION** SL Increase your alertness to phishing risks Install a cyber security culture and create cyber security heroes.Identify fraudulent emails but also provide specific guidance on how to handle suspect communications.	**8. CHANNELS of BEHAVIOR** CH Since a user profile contains confidential user information, it is important that it does not add new security risks. We use a one-way secure hash function to hash the confidential data before it is stored. When the system needs to determine the equivalence between the data, the system just needs to compare the hash values.

3.4 Problem Solution fit



PROJECT DESIGN PHASE I

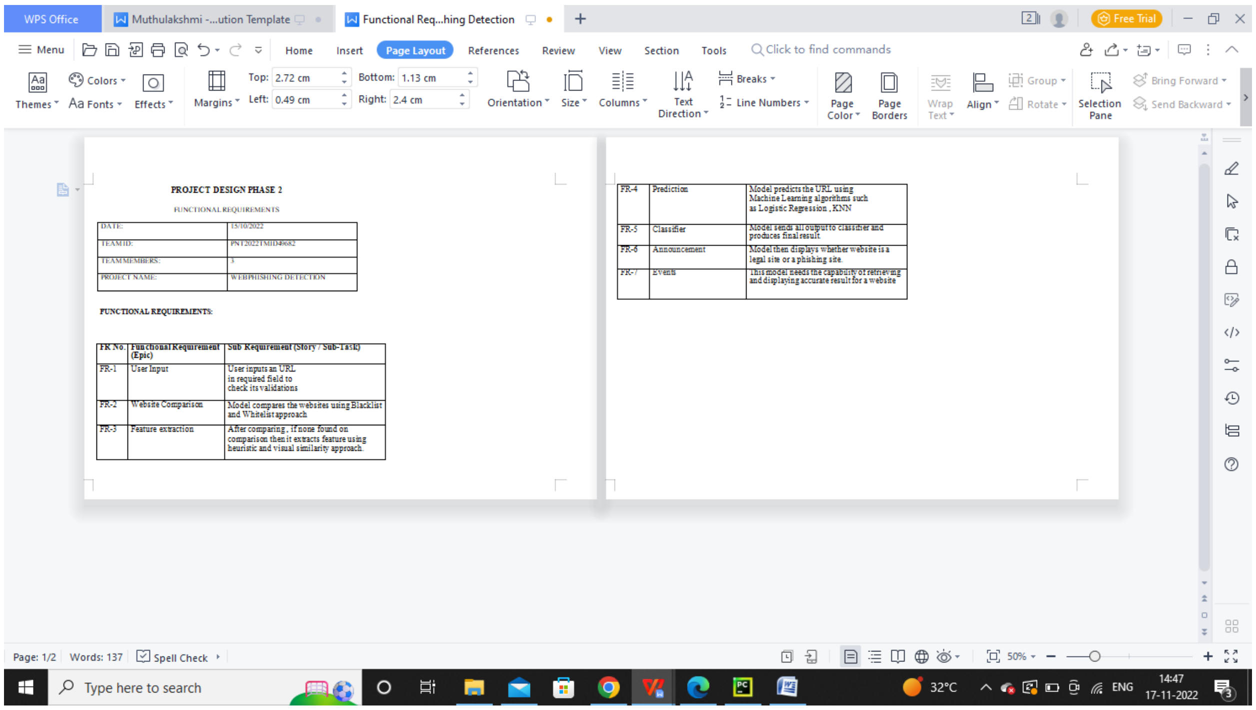
PROPOSED SOLUTION TEMPLATE

DATE	24 September 2022
TEAMID	PNT2022TMID49682
PROJECTNAME	Web Phishing DetectionUsing Machine Learning
MAXIMUMMARKS	2

S.NO	PARAMETER	DESCRIPTION
1.	Problem Statement	Phishing detection techniques do suffer low detection accuracy and high false alarm especially when novel phishing approaches are introduced. Besides, the most common technique used, blacklist-basedmethod is inefficient in responding to emanating phishing attacks since registering new domain has become easier, no comprehensive blacklist can ensure a perfect up-to-date database. Furthermore, page content inspection has been used by some strategies to overcome the false negative problems and complement the vulnerabilities of the stale lists. Moreover, page content inspection algorithms each have different approach to phishing website detection with varying degrees of accuracy.
2.	Idea	➤ Web Phishing Detection 1. **Identify the criteria** that can recognize fake URLs 2. **Build a decision tree** that can iterate through the criteria 3. **Train our model** to recognize fake vs real URLs 4. **Evaluate our model** to see how it performs 5. **Check for false positives/negatives** ➤ **Increase user alertness to phishing risksAssess the impacts of cyber security awareness training Segment phishing simulationTrain our model to recognize fake vs real URLs Change behavior to eliminate the automatic trust response**Identify the criteria that can recognize fake URLs Build a decision tree that can iterate through the criteria **Install a cyber security culture and create cyber security heroes.** ➤ Install a cyber security culture and create cyber security heroes Evaluate our model to see how it performs Assess the impacts of cyber security awareness training strong password policies and reminding employees about the risks that can come in the format of attachments, emails and URLs Provide ongoing communication and campaigns about cyber security and phishing.This is best done by designing rigorous user education programs that help users not only identify fraudulent emails but also provide specific guidance on how to handle suspect communications Alarming content full of warnings and potential consequences In the sections below, we focus on safely handling emails that do manage to breach the security of the software layer antivirus software,both desktop and network firewalls antispyware software; antiphishing toolbar (installed in web browsers);. Ensure that all applications, operating systems, network tools, and internal software are up-to-date and secure. Install malware protection and anti-spam software Eliminate the cyber threat risk level.
3.	Novelty	➤ Future Work will include using another algorithm like deep-learning for phishing web page detection and compare the effectiveness with the current result.More also, a web browser plug -in will be developed based on an efficient algorithm to detect phishing website and thus protect users in real time.
4.	Social impact	➤ Phishing has a list of negative effects on a business, including loss of money,loss of intellectual property, damage to reputation,and disruption of operational activities.These effects work together to cause loss company value,sometime with irreparable repercussions.
5.	BusinessModel	➤ Phishing is big business.Attacks have shown record growth in recent years, and a solid security awareness program is an integral part of any defense-in-depth strategy. Sophos Phish threat educates and tests your end users through automated attack simulations,Quality security awareness training, and actionable reporting metrics.Phish Threat provides you with the flexibility and customization that your organization needs to facilitate a positive security awareness culture.
6.	Scalability of solution	➤ Use Web-Phishing Protection and web-spam software to protect yourself when malicious message slip through to your computer.Web-Malware software is programmed by security researches to spot even the stealthiest malware. ➤ Certain Web-Phishing solutions Scan the content of inbound and internal emails for any sign of language that suggest a potential phishing or impersonation attack. Other Web-Phishing technology scans the links and attachments in email and blocks users from accessing them if they are determined to be suspicious. ➤ E-Banking Phishing Website can be detected based on some important characteristics like URL and domain identity and security and encryption criteria in the final phishing detection rate once a user makes a transaction online when he makes payment through an e-banking website our system will use a data mining algorithm to detect whether the e-banking Website is a phishing website or not.

4. REQUIREMENT ANALYSIS

4.1 Functional requirement



PROJECT DESIGN PHASE 2

FUNCTIONAL REQUIREMENTS

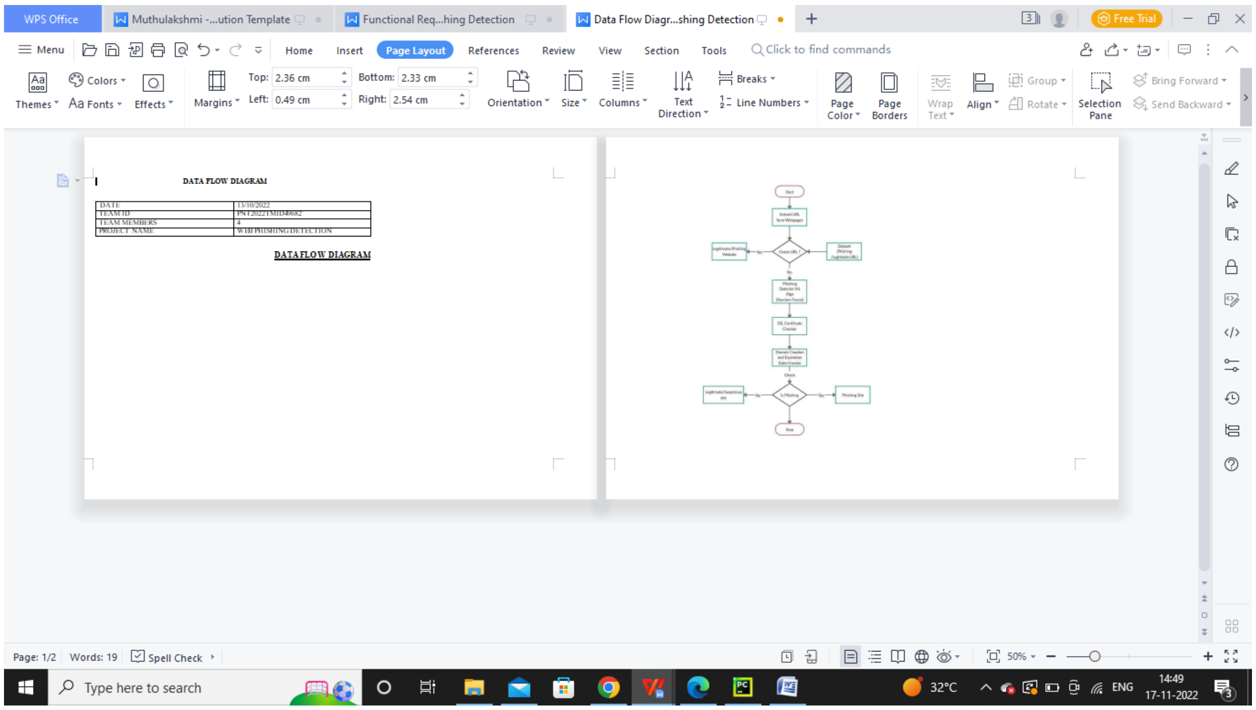
DATE:	15/10/2022
TEAMID:	PNT2022TMID49682
TEAMMEMBERS:	3
PROJECT NAME:	WEBPHISHING DETECTION

FUNCTIONAL REQUIREMENTS:

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Input	User inputs an URL in required field to check its validations
FR-2	Website Comparison	Model compares the websites using Blacklist and Whitelist approach
FR-3	Feature extraction	After comparing , if none found on comparison then it extracts feature using heuristic and visual similarity approach.
FR-4	Prediction	Model predicts the URL using Machine Learning algorithms such as Logistic Regression , KNN
FR-5	Classifier	Model sends all output to classifier and produces final result.
FR-6	Announcement	Model then displays whether website is a legal site or a phishing site.
FR-7	Events	This model needs the capability of retrieving and displaying accurate result for a website

5.PROJECT DESIGN

5.1 Data Flow Diagram



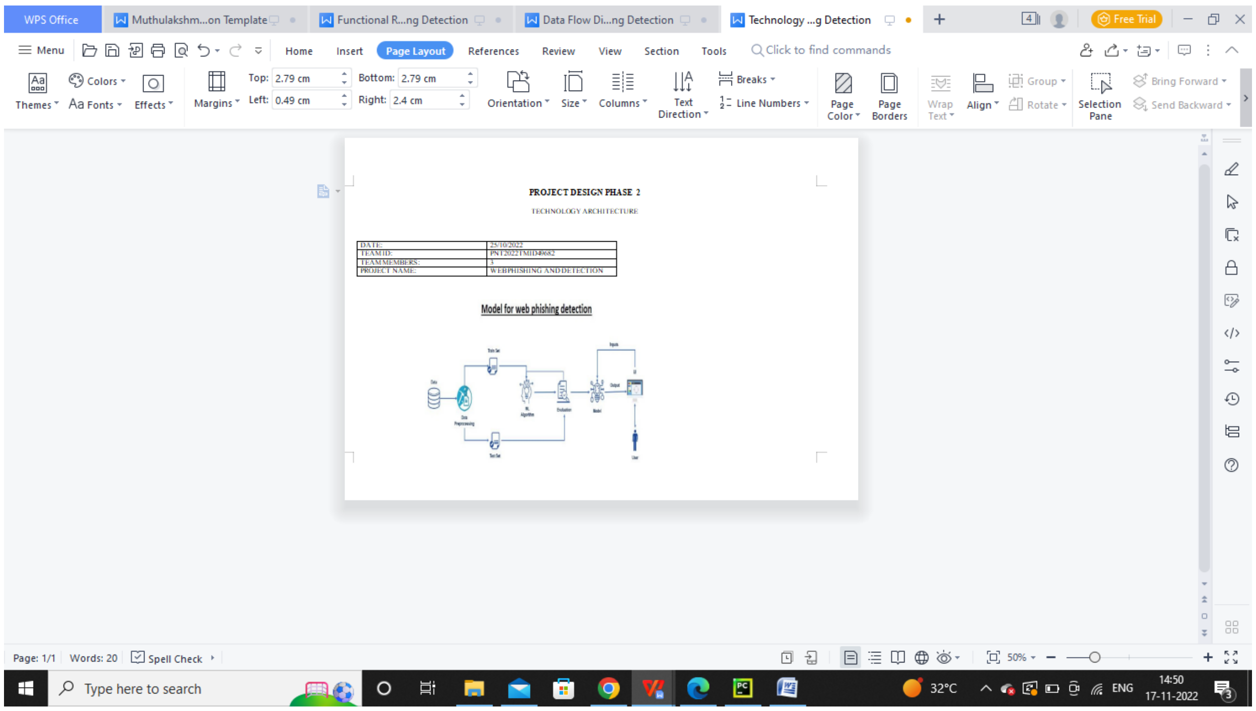
DATA FLOW DIAGRAM

DATE	13/10/2022
TEAM ID	PNT2022TMID49682
TEAM MEMBERS	4
PROJECT NAME	WEB PHISHING DETECTION

DATA FLOW DIAGRAM

Start
Extract URL form Webpages
Check URL ?
No
Legitimate/Phishing Website
Dataset (Phishing /Legitimate URL)
No
Phishing Detector ML Algo (Random Forest)
SSL Certificate Checker
Domain Creation and Expiration Date Checker
Check
Is Phishing
No
Legitimate/Suspicious site
Yes
Phishing Site
Stop

5.2 Solution & Technical Architecture

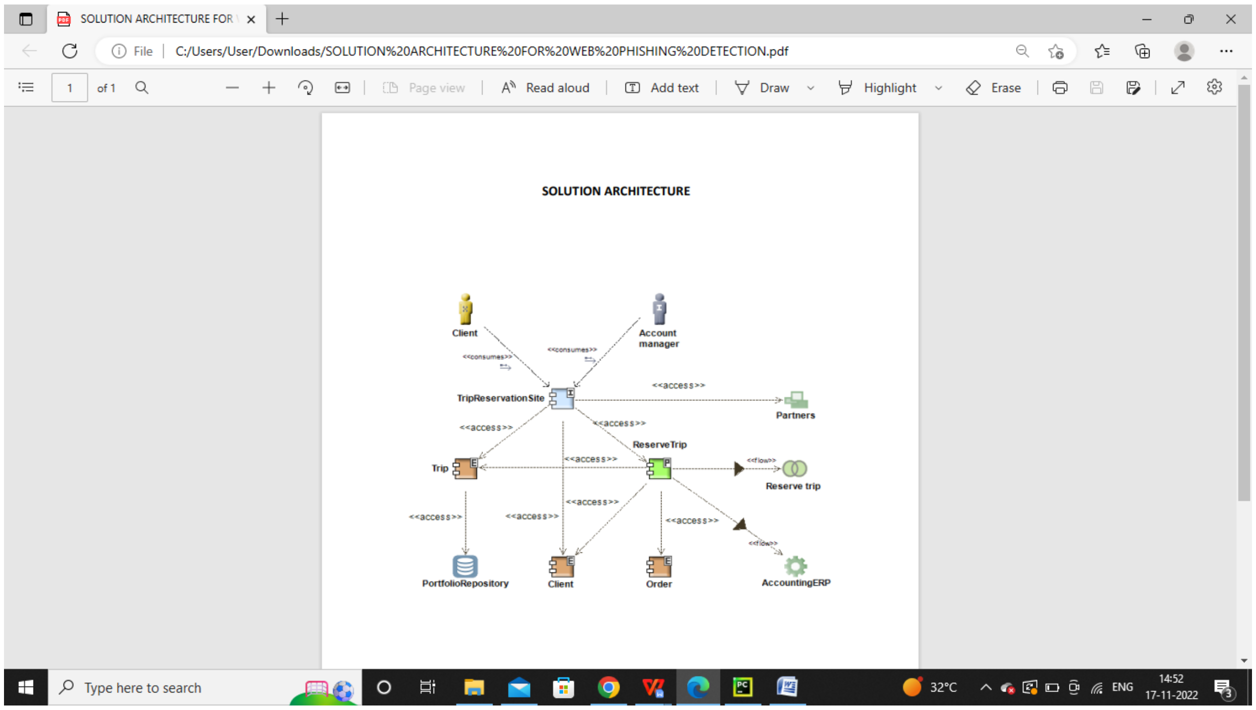


PROJECT DESIGN PHASE 2

TECHNOLOGY ARCHITECTURE

DATE:	25/10/2022
TEAM ID:	PNT2022TMID49682
TEAM MEMBERS:	3
PROJECT NAME:	WEBPHISHING AND DETECTION

Model for web phishing detection



SOLUTION ARCHITECTURE

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv("phishing_detection.csv")
df.head()
# result is the target attribute
df.head()
# result is the target attribute
df.columns
df.info()
# no null values
df.isnull().sum()
plt.figure(figsize=(7,6))
# count numbers of class records for 'Result' target attribute
sns.countplot('Result', data = df)
df['Result'].value_counts()
#we can see the target class count is almost equally balanced. Hence Data
augmentation is necessary

#Module1
# creating correlation matrix on the features
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(30, 25))
# representation of correalation matrix through heatmap
g = sns.heatmap(df[top_corr_features].corr(), annot = True, cmap = "RdYlGn")
col_corr = set() # Set of all the names of deleted columns
def correlation(dataset, threshold):
  corr_matrix = dataset.corr()
  for i in range(len(corr_matrix.columns)):
      for j in range(i):
          if (abs(corr_matrix.iloc[i, j]) >= threshold) and
(corr_matrix.columns[j] not in col_corr):
              colname = corr_matrix.columns[i] # getting the name of column
              col_corr.add(colname)
# remove multicollinear column with collinearity greater than 0.85
correlation(df, 0.85)
col_corr
{'popUpWidnow'}
# identifying weakly correlated features with target attribute
weak_col_corr = set()
def weakcorrelation(dataset, threshold):
  corr_matrix = dataset.corr()
  idx = 0
  for feature in corr_matrix['Result']:
    if(feature < threshold):
      weak_col_corr.add(corr_matrix.columns[idx])
    idx += 1

 # dropping features with correlation less than 0.01
weakcorrelation(df, 0.01)
print(weak_col_corr)
# gathering all columns that were identified to be deleted
del_col = col_corr.union(weak_col_corr)
del_col
# dropping columns permanantly
df.drop(del_col, axis = 1, inplace = True)
df.isnull().sum()
#splitting dataset into train and test
# Load libraries
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split
function
from sklearn import metrics #Import scikit-learn metrics module for accuracy
calculation

# input attribute and target attribute
X = df.iloc[: , :-1]
y = df.iloc[:, -1:]

y
# train test split with test size as 0.25
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1)
#Module 2: Decision tree classifier
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
clf.score(X_train, y_train)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#visualize decision tree
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file= dot_data, feature_names=list(X.columns), filled =
True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png())
from sklearn.metrics import confusion_matrix, classification_report
# classification report
print(classification_report(y_test, y_pred))
#Confusion matrix for decision tree model

#A confusion matrix is a summary of prediction results on a classification
problem.

#The number of correct and incorrect predictions are summarized with count
values and broken down by each class. This is the key to the confusion matrix.
from google.colab.patches import cv2_imshow
import cv2
img = cv2.imread('confusion.png', cv2.IMREAD_UNCHANGED)
cv2_imshow(img)
```

```
# the confusion matrix for our model
confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_matrix(y_test, y_pred), annot = True, fmt='0.0f')
#Random Forest
# prediction on validation dataset
y_pred = rfc.predict(X_test)

# prediction on training dataset
y_pred_train = rfc.predict(X_train)

from sklearn import metrics
print("Train ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_train,
y_pred_train))
from sklearn import metrics
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_matrix(y_test, y_pred), annot = True, fmt='0.0f')
SVM
from sklearn.svm import SVC
# Building a Support Vector Machine on train data
svc_model = SVC(C= .1, gamma= 1, kernel='sigmoid', random_state=42)
```

6.2 Sprint Delivery Schedule

```
svc_model.fit(X_train, y_train)
rediction = svc_model .predict(X_test)
# check the accuracy on the training set
print('Accuracy of training data: ', svc_model.score(X_train, y_train))
print('Accuracy of validation data: ',svc_model.score(X_test, y_test))
from sklearn.metrics import confusion_matrix, classification_report
# generating classification report
print(classification_report(y_test, prediction))
confusion_matrix(y_test, prediction)
sns.heatmap(confusion_matrix(y_test, prediction), annot = True, fmt='0.0f')
#K-nearest Neighbors
```

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
#choosing the best values of k (neighbours)
neighbour = []
accuracy = []
for k in range(1, 20):
  k_near = KNeighborsClassifier(n_neighbors=k)
  k_near.fit(X,y)
  Y_pre_test = k_near.predict(X_test)
  Y_pre_train = k_near.predict(X_train)
  test_accurry = accuracy_score(Y_pre_test, y_test)
  neighbour.append(k)
  accuracy.append(test_accurry)
# plotting for n neighbour vs accuracy
plt.plot(neighbour, accuracy)
plt.title('n neighbour vs accuracy')
plt.xlabel('n neighbour')
plt.ylabel('accuracy')
#KNN with n_neighbour = 1
k_near = KNeighborsClassifier(n_neighbors=1)
k_near.fit(X_train,y_train)
Y_pre_test = k_near.predict(X_test)
Y_pre_train = k_near.predict(X_train)
from sklearn.metrics import accuracy_score
train_accurry = accuracy_score(Y_pre_train, y_train)
test_accurry = accuracy_score(Y_pre_test, y_test)
print('Accuracy for train dataset for K-neariest : ', train_accurry)
print('Accuracy for test dataset for K-neariest : ', test_accurry)
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, Y_pre_test ))
confusion_matrix(y_test, Y_pre_test )
sns.heatmap(confusion_matrix(y_test, Y_pre_test), annot = True, fmt='0.0f')
Logistic Regression
from sklearn.linear_model import LogisticRegression
lgr = LogisticRegression(random_state=0)
lgr.fit(X_train,y_train)
y_pre_test = lgr.predict(X_test)
y_pre_train = lgr.predict(X_train)
from sklearn.metrics import accuracy_score
```

```
train_accurry = accuracy_score(y_pre_train, y_train)
test_accurry = accuracy_score(y_pre_test, y_test)
print('Accuracy for train dataset for logistic reg : ', train_accurry)
print('Accuracy for test dataset for logistic reg : ', test_accurry)
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_pre_test ))
confusion_matrix(y_test, y_pre_test )
sns.heatmap(confusion_matrix(y_test,y_pre_test), annot = True, fmt='0.0f')
#Naive Bayes
from sklearn.naive_bayes import GaussianNB, MultinomialNB, CategoricalNB,
BernoulliNB, ComplementNB
# Bernoullis Navaive bayes classifier
nvb = BernoulliNB()
nvb.fit(X_train,y_train)
y_pre_test = nvb.predict(X_test)
y_pre_train = nvb.predict(X_train)
from sklearn.metrics import accuracy_score
train_accurry = accuracy_score(y_pre_train, y_train)
test_accurry = accuracy_score(y_pre_test, y_test)
print('Accuracy for train dataset for naive bayes  reg : ', train_accurry)
print('Accuracy for test dataset for naive bayes reg : ', test_accurry)
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_pre_test ))
confusion_matrix(y_test, y_pre_test)
sns.heatmap(confusion_matrix(y_test, y_pre_test), annot = True, fmt='0.0f')
#Adaboost
from sklearn.ensemble import AdaBoostClassifier

# Create Decision Tree classifer object
clf = AdaBoostClassifier(n_estimators=100, random_state=42)

# Train Decision Tree Classifer
clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
clf.score(X_train, y_train)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_matrix(y_test, y_pred), annot = True, fmt='0.0f')
#Hybrid Ensembler
#Ensembler method used - Max voting: It is mainly used for classification
problems.
#The method consists of building multiple models independently and getting
their individual output called 'vote'.
#The class with maximum votes is returned as output.
# importing voting classifier
from sklearn.ensemble import VotingClassifier
model_1 = RandomForestClassifier(n_estimators=100)
model_2 = KNeighborsClassifier(n_neighbors=1)
model_3 = LogisticRegression(random_state=0)
model_4 = BernoulliNB()
model_5 = DecisionTreeClassifier()
ensemble = VotingClassifier(estimators=[('RandomForest', model_1), ('KNN',
model_2), ('LogisticRegression', model_3), ('NaiveBayes', model_4),
                                        ('DT', model_5)], voting='hard')
ensemble.fit(X_train, y_train)
y_pred_test = ensemble.predict(X_test)
y_pred_train = ensemble.predict(X_train)
from sklearn.metrics import accuracy_score
train_accurry = accuracy_score(y_pred_train, y_train)
test_accurry = accuracy_score(y_pred_test, y_test)
print('Accuracy for train dataset for naive bayes  reg : ', train_accurry)
print('Accuracy for test dataset for naive bayes reg : ', test_accurry)
import pickle
filename = 'finalized_model.sav'
pickle.dump(ensemble, open(filename, 'wb'))
phising = ensemble.predict(input)
```

7. CODING & SOLUTIONING

(Explain the features added in the project along with code)

7.1 Feature 1

```
# importing necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
df = pd.read_csv("phishing_detection.csv")
df.head()
# result is the target attribute
df.head()
# result is the target attribute
df.columns
df.info()
# no null values
df.isnull().sum()
plt.figure(figsize=(7,6))
# count numbers of class records for 'Result' target attribute
sns.countplot('Result', data = df)
df['Result'].value_counts()
#we can see the target class count is almost equally balanced. Hence Data
augmentation is necessary

#Module1
# creating correlation matrix on the features
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(30, 25))
# representation of correalation matrix through heatmap
g = sns.heatmap(df[top_corr_features].corr(), annot = True, cmap = "RdYlGn")
col_corr = set() # Set of all the names of deleted columns
def correlation(dataset, threshold):
  corr_matrix = dataset.corr()
  for i in range(len(corr_matrix.columns)):
      for j in range(i):
          if (abs(corr_matrix.iloc[i, j]) >= threshold) and
(corr_matrix.columns[j] not in col_corr):
              colname = corr_matrix.columns[i] # getting the name of column
              col_corr.add(colname)
# remove multicollinear column with collinearity greater than 0.85
correlation(df, 0.85)
col_corr
{'popUpWidnow'}
# identifying weakly correlated features with target attribute
weak_col_corr = set()
def weakcorrelation(dataset, threshold):
  corr_matrix = dataset.corr()
  idx = 0
  for feature in corr_matrix['Result']:
    if(feature < threshold):
      weak_col_corr.add(corr_matrix.columns[idx])
    idx += 1

 # dropping features with correlation less than 0.01
weakcorrelation(df, 0.01)
print(weak_col_corr)
# gathering all columns that were identified to be deleted
del_col = col_corr.union(weak_col_corr)
del_col
# dropping columns permanantly
df.drop(del_col, axis = 1, inplace = True)
df.isnull().sum()
#splitting dataset into train and test
# Load libraries
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.model_selection import train_test_split # Import train_test_split
function
from sklearn import metrics #Import scikit-learn metrics module for accuracy
calculation

# input attribute and target attribute
X = df.iloc[: , :-1]
y = df.iloc[:, -1:]
```

```
y
# train test split with test size as 0.25
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=1)
#Module 2: Decision tree classifier
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
clf.score(X_train, y_train)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
#visualize decision tree
from six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dot_data = StringIO()
export_graphviz(clf, out_file= dot_data, feature_names=list(X.columns), filled =
True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png())
from sklearn.metrics import confusion_matrix, classification_report
# classification report
print(classification_report(y_test, y_pred))
#Confusion matrix for decision tree model

#A confusion matrix is a summary of prediction results on a classification
problem.

#The number of correct and incorrect predictions are summarized with count
values and broken down by each class. This is the key to the confusion matrix.
from google.colab.patches import cv2_imshow
```

```
import cv2
img = cv2.imread('confusion.png', cv2.IMREAD_UNCHANGED)
cv2_imshow(img)
# the confusion matrix for our model
confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_matrix(y_test, y_pred), annot = True, fmt='0.0f')
#Random Forest
# prediction on validation dataset
y_pred = rfc.predict(X_test)

# prediction on training dataset
y_pred_train = rfc.predict(X_train)

from sklearn import metrics
print("Train ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_train,
y_pred_train))
from sklearn import metrics
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_matrix(y_test, y_pred), annot = True, fmt='0.0f')
SVM
from sklearn.svm import SVC
# Building a Support Vector Machine on train data
svc_model = SVC(C= .1, gamma= 1, kernel='sigmoid', random_state=42)

svc_model.fit(X_train, y_train)
rediction = svc_model .predict(X_test)
# check the accuracy on the training set
print('Accuracy of training data: ', svc_model.score(X_train, y_train))
print('Accuracy of validation data: ',svc_model.score(X_test, y_test))
from sklearn.metrics import confusion_matrix, classification_report
# generating classification report
print(classification_report(y_test, prediction))
confusion_matrix(y_test, prediction)
sns.heatmap(confusion_matrix(y_test, prediction), annot = True, fmt='0.0f')
#K-nearest Neighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
#choosing the best values of k (neighbours)
neighbour = []
accuracy = []
for k in range(1, 20):
  k_near = KNeighborsClassifier(n_neighbors=k)
  k_near.fit(X,y)
  Y_pre_test = k_near.predict(X_test)
  Y_pre_train = k_near.predict(X_train)
  test_accurry = accuracy_score(Y_pre_test, y_test)
  neighbour.append(k)
  accuracy.append(test_accurry)
# plotting for n neighbour vs accuracy
plt.plot(neighbour, accuracy)
plt.title('n neighbour vs accuracy')
plt.xlabel('n neighbour')
plt.ylabel('accuracy')
#KNN with n_neighbour = 1
k_near = KNeighborsClassifier(n_neighbors=1)
k_near.fit(X_train,y_train)
Y_pre_test = k_near.predict(X_test)
Y_pre_train = k_near.predict(X_train)
from sklearn.metrics import accuracy_score
train_accurry = accuracy_score(Y_pre_train, y_train)
test_accurry = accuracy_score(Y_pre_test, y_test)
print('Accuracy for train dataset for K-neariest : ', train_accurry)
print('Accuracy for test dataset for K-neariest : ', test_accurry)
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, Y_pre_test ))
confusion_matrix(y_test, Y_pre_test )
sns.heatmap(confusion_matrix(y_test, Y_pre_test), annot = True, fmt='0.0f')
Logistic Regression
from sklearn.linear_model import LogisticRegression
lgr = LogisticRegression(random_state=0)
lgr.fit(X_train,y_train)
y_pre_test = lgr.predict(X_test)
y_pre_train = lgr.predict(X_train)
from sklearn.metrics import accuracy_score
train_accurry = accuracy_score(y_pre_train, y_train)
```

```
test_accurry = accuracy_score(y_pre_test, y_test)
print('Accuracy for train dataset for logistic reg : ', train_accurry)
print('Accuracy for test dataset for logistic reg : ', test_accurry)
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_pre_test ))
confusion_matrix(y_test, y_pre_test )
sns.heatmap(confusion_matrix(y_test,y_pre_test), annot = True, fmt='0.0f')
#Naive Bayes
from sklearn.naive_bayes import GaussianNB, MultinomialNB, CategoricalNB,
BernoulliNB, ComplementNB
# Bernoullis Navaive bayes classifier
nvb = BernoulliNB()
nvb.fit(X_train,y_train)
y_pre_test = nvb.predict(X_test)
y_pre_train = nvb.predict(X_train)
from sklearn.metrics import accuracy_score
train_accurry = accuracy_score(y_pre_train, y_train)
test_accurry = accuracy_score(y_pre_test, y_test)
print('Accuracy for train dataset for naive bayes  reg : ', train_accurry)
print('Accuracy for test dataset for naive bayes reg : ', test_accurry)
from sklearn.metrics import confusion_matrix, classification_report
print(classification_report(y_test, y_pre_test ))
confusion_matrix(y_test, y_pre_test)
sns.heatmap(confusion_matrix(y_test, y_pre_test), annot = True, fmt='0.0f')
#Adaboost
from sklearn.ensemble import AdaBoostClassifier

# Create Decision Tree classifer object
clf = AdaBoostClassifier(n_estimators=100, random_state=42)

# Train Decision Tree Classifer
clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
clf.score(X_train, y_train)
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
from sklearn.metrics import confusion_matrix, classification_report
```

```
print(classification_report(y_test, y_pred))
confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_matrix(y_test, y_pred), annot = True, fmt='0.0f')
#Hybrid Ensembler
#Ensembler method used - Max voting: It is mainly used for classification
problems.
#The method consists of building multiple models independently and getting
their individual output called 'vote'.
#The class with maximum votes is returned as output.
# importing voting classifier
from sklearn.ensemble import VotingClassifier
model_1 = RandomForestClassifier(n_estimators=100)
model_2 = KNeighborsClassifier(n_neighbors=1)
model_3 = LogisticRegression(random_state=0)
model_4 = BernoulliNB()
model_5 = DecisionTreeClassifier()
ensemble = VotingClassifier(estimators=[('RandomForest', model_1), ('KNN',
model_2), ('LogisticRegression', model_3), ('NaiveBayes', model_4),
                                        ('DT', model_5)], voting='hard')
ensemble.fit(X_train, y_train)
y_pred_test = ensemble.predict(X_test)
y_pred_train = ensemble.predict(X_train)
from sklearn.metrics import accuracy_score
train_accurry = accuracy_score(y_pred_train, y_train)
test_accurry = accuracy_score(y_pred_test, y_test)
print('Accuracy for train dataset for naive bayes  reg : ', train_accurry)
print('Accuracy for test dataset for naive bayes reg : ', test_accurry)
import pickle
filename = 'finalized_model.sav'
pickle.dump(ensemble, open(filename, 'wb'))
phising = ensemble.predict(input)
```

7.2 Feature 2

```
def having_ip_address(url):
  import re
  x = re.search('^(http|https)://\d+\.\d+\.\d+\.\d+\.*', url)
  if x:
    return 1
  else:
    return -1

having_ip_address('https://128.36.54.192/questions/59020008/how-to-import-
functions-of-a-jupyter-notebook-into-another-jupyter-notebook-in-g')


1
def URL_Length(url):
  if len(url) < 54:
    return -1
  elif len(url) >= 54 and len(url) <=75:
    return 0
  else:
    return 1
URL_Length('https://stackoverflow.com/questions/59020008/how-to-import-
functions-of-a-jupyter-notebook-into-another-jupyter-notebook-in-g')
1
```

###Module3

```
def haveAtSign(url):
  if "@" in url:
    at = 1
  else:
    at = -1
  return at
haveAtSign('https://stackoverflow.com/questions/59020008/how-to-import-
functions-of-a-jupyter-notebook-into-another-jupyter-notebook-in-g')
-1
```

Prefix_Suffix

```
def prefixSuffix(url):
    from urllib.parse import urlparse
    if '-' in urlparse(url).netloc:
        return 1
    else:
        return -1
prefixSuffix('https://dfhdfhdfgs.tokyo/ja-jp/account/login')
-1
```

Having sub domain

```
def sub_domain_count(url):
  !pip install tld
  from urllib.parse import urlparse
  from tld import get_tld
  domain = urlparse(url).netloc
  domain = domain.split('.')
  top_domain = get_tld(url)
  top_domain = top_domain.split('.')
  sub_domain = set(domain) - set(top_domain)
  count = len(sub_domain)
  if(count == 1):
    return -1
  if(count == 2):
    return 0
  else:
    return 1
sub_domain_count('https://stackoverflow.com/questions/59020008/how-to-import-
functions-of-a-jupyter-notebook-into-another-jupyter-notebook-in-g')
Collecting tld
  Downloading tld-0.12.6-py37-none-any.whl (412 kB)
?25l
K     |█                               | 10 kB 26.3 MB/s eta 0:00:01
K     |██                              | 20 kB 9.8 MB/s eta 0:00:01
K     |██                              | 30 kB 8.4 MB/s eta 0:00:01
K     |███                             | 40 kB 7.6 MB/s eta 0:00:01
K     |████                            | 51 kB 4.1 MB/s eta 0:00:01
K     |█████                           | 61 kB 4.4 MB/s eta 0:00:01
K     |██████                          | 71 kB 4.3 MB/s eta 0:00:01
K     |██████                          | 81 kB 4.9 MB/s eta 0:00:01
K     |███████                         | 92 kB 5.1 MB/s eta 0:00:01
K     |████████                        | 102 kB 4.1 MB/s eta 0:00:01
K     |█████████                       | 112 kB 4.1 MB/s eta 0:00:01
K     |█████████                       | 122 kB 4.1 MB/s eta 0:00:01
K     |██████████                      | 133 kB 4.1 MB/s eta 0:00:01
K     |███████████                     | 143 kB 4.1 MB/s eta 0:00:01
K     |████████████                    | 153 kB 4.1 MB/s eta 0:00:01
K     |█████████████                   | 163 kB 4.1 MB/s eta 0:00:01
K     |█████████████                   | 174 kB 4.1 MB/s eta 0:00:01
K     |██████████████                  | 184 kB 4.1 MB/s eta 0:00:01
```

```
K     |███████████████                 | 194 kB 4.1 MB/s eta 0:00:01
K     |████████████████                | 204 kB 4.1 MB/s eta 0:00:01
K     |█████████████████               | 215 kB 4.1 MB/s eta 0:00:01
K     |█████████████████               | 225 kB 4.1 MB/s eta 0:00:01
K     |██████████████████              | 235 kB 4.1 MB/s eta 0:00:01
K     |███████████████████             | 245 kB 4.1 MB/s eta 0:00:01
K     |████████████████████            | 256 kB 4.1 MB/s eta 0:00:01
K     |█████████████████████           | 266 kB 4.1 MB/s eta 0:00:01
K     |█████████████████████           | 276 kB 4.1 MB/s eta 0:00:01
K     |██████████████████████          | 286 kB 4.1 MB/s eta 0:00:01
K     |███████████████████████         | 296 kB 4.1 MB/s eta 0:00:01
K     |████████████████████████        | 307 kB 4.1 MB/s eta 0:00:01
K     |█████████████████████████       | 317 kB 4.1 MB/s eta 0:00:01
K     |█████████████████████████       | 327 kB 4.1 MB/s eta 0:00:01
K     |██████████████████████████      | 337 kB 4.1 MB/s eta 0:00:01
K     |███████████████████████████     | 348 kB 4.1 MB/s eta 0:00:01
K     |████████████████████████████    | 358 kB 4.1 MB/s eta 0:00:01
K     |█████████████████████████████   | 368 kB 4.1 MB/s eta 0:00:01
K     |█████████████████████████████   | 378 kB 4.1 MB/s eta 0:00:01
K     |██████████████████████████████  | 389 kB 4.1 MB/s eta 0:00:01
K     |███████████████████████████████ | 399 kB 4.1 MB/s eta 0:00:01
K     |████████████████████████████████| 409 kB 4.1 MB/s eta 0:00:01
K     |████████████████████████████████| 412 kB 4.1 MB/s
?25hInstalling collected packages: tld
Successfully installed tld-0.12.6

-1
```

###SSL final state

```python
def sslVerify(url):
  from urllib.parse import urlparse,urlencode,urlsplit, urlunsplit
  from urllib.request import Request, urlopen, ssl, socket
  import json
  from datetime import datetime
  split_url = urlsplit(url)
  #some site without http/https in the path
  port = '443'

  hostname = split_url.netloc
  context = ssl.create_default_context()
  try:
    with socket.create_connection((hostname, port)) as sock:
      with context.wrap_socket(sock, server_hostname=hostname) as ssock:
          data = json.dumps(ssock.getpeercert())
          res = json.loads(data)
    notBefore = datetime.strptime(res["notBefore"],'%b  %d %H:%M:%S %Y
%Z').date()
    notAfter = datetime.strptime(res["notAfter"],'%b  %d %H:%M:%S %Y
%Z').date()
    # print(notBefore, notAfter)
    if(ssl.SSLCertVerificationError(ssock) == True):
      return 1
    elif (notAfter.year-notBefore.year)+(notAfter.month-notBefore.month)*0.1 >=
1:
      return -1
    else:
      return 0
  except:
    return 1
sslVerify("http://postdebanks.com/DIE/POST/diepost/")
1
```

Port

```
def port(domain):
  import requests
  import json
  response =
requests.get("https://api.viewdns.info/portscan/?host="+domain+"&apikey=1bf03196
763a201bcc66a59bf88ed8ddf7a9432f&output=json")
  myjson = response.json()
  # print(myjson)
  pref_stat = {21:'closed', 22:'closed', 23:'closed', 80:'open', 443:'open',
445 : 'closed', 1433:'closed', 1521:'closed', 3306 : 'closed', 3389:'closed' }
  flag = -1
  for i in range(len(myjson['response']['port'])):
    if int(myjson['response']['port'][i]['number']) in pref_stat:
      if(myjson['response']['port'][0]['status'] !=
pref_stat[int(myjson['response']['port'][0]['number'])]):
        flag = 1
  return flag
# port('postdebanks.com')
```

Request_URL

```
def request_url(url):
  from urllib.parse import urlparse,urlencode,urlsplit, urlunsplit
  from bs4 import BeautifulSoup
  import requests
  import re
  !pip install tldextract
  import tldextract
  Null_format = ["", "#", "#nothing", "#doesnotexist", "#null", "#void",
"#whatever",
              "#content", "javascript::void(0)", "javascript::void(0);",
"javascript::;", "javascript"]

  def is_URL_accessible(url):
    page = None
    try:
        page = requests.get(url, timeout=5)
    except:
        parsed = urlparse(url)
        url = parsed.scheme+'://'+parsed.netloc
        if not parsed.netloc.startswith('www'):
            url = parsed.scheme+'://www.'+parsed.netloc
            try:
                page = requests.get(url, timeout=5)
            except:
                page = None
                pass
    if page and page.status_code == 200 and page.content not in ["b''", "b'
'"]:
        return True, url, page
    else:
        return False, None, None

  state, iurl, page = is_URL_accessible(url)
  def get_domain(url):
      o = urlsplit(url)
      return o.hostname, tldextract.extract(url).domain, o.path

  if state:
        print('Yes')
        content = page.content
        hostname, domain, path = get_domain(url)
  else:
        print('No state')

  Media = {'internals':[], 'externals':[], 'null':[]}

  def external_media(Media):
    total = len(Media['internals']) + len(Media['externals'])
    externals = len(Media['externals'])
    try:
        percentile = externals / float(total) * 100
    except:
        return 0
    return percentile

  def findMedia(Media,domain, hostname ):
    Null_format = ["", "#", "#nothing", "#doesnotexist", "#null", "#void",
"#whatever",
               "#content", "javascript::void(0)", "javascript::void(0);",
"javascript::;", "javascript"]
    soup = BeautifulSoup(content, 'html.parser', from_encoding='iso-8859-1')
    for img in soup.find_all('img', src=True):
          dots = [x.start(0) for x in re.finditer('\.', img['src'])]
          if hostname in img['src'] or domain in img['src'] or len(dots) == 1
or not img['src'].startswith('http'):
              if not img['src'].startswith('http'):
                  if not img['src'].startswith('/'):
                      Media['internals'].append(hostname+'/'+img['src'])
                  elif img['src'] in Null_format:
                      Media['null'].append(img['src'])
                  else:
                      Media['internals'].append(hostname+img['src'])
          else:
              Media['externals'].append(img['src'])
    for audio in soup.find_all('audio', src=True):
      dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
      if hostname in audio['src'] or domain in audio['src'] or len(dots) == 1
or not audio['src'].startswith('http'):
          if not audio['src'].startswith('http'):
              if not audio['src'].startswith('/'):
                  Media['internals'].append(hostname+'/'+audio['src'])
              elif audio['src'] in Null_format:
                  Media['null'].append(audio['src'])
              else:
                  Media['internals'].append(hostname+audio['src'])
      else:
          Media['externals'].append(audio['src'])

    for embed in soup.find_all('embed', src=True):
      dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
      if hostname in embed['src'] or domain in embed['src'] or len(dots) == 1
or not embed['src'].startswith('http'):
          if not embed['src'].startswith('http'):
              if not embed['src'].startswith('/'):
                  Media['internals'].append(hostname+'/'+embed['src'])
              elif embed['src'] in Null_format:
                  Media['null'].append(embed['src'])
              else:
                  Media['internals'].append(hostname+embed['src'])
      else:
          Media['externals'].append(embed['src'])

    for i_frame in soup.find_all('iframe', src=True):
      dots = [x.start(0) for x in re.finditer('\.', i_frame['src'])]
      if hostname in i_frame['src'] or domain in i_frame['src'] or len(dots) ==
1 or not i_frame['src'].startswith('http'):
          if not i_frame['src'].startswith('http'):
              if not i_frame['src'].startswith('/'):
                  Media['internals'].append(hostname+'/'+i_frame['src'])
              elif i_frame['src'] in Null_format:
                  Media['null'].append(i_frame['src'])
              else:
                  Media['internals'].append(hostname+i_frame['src'])
      else:
          Media['externals'].append(i_frame['src'])

  findMedia(Media, domain, hostname)
  if external_media(Media) < 22:
    return -1
  elif external_media(Media) >= 22 and external_media(Media) < 61:
    return 0
  else:
    return 1
#
request_url('http://www.budgetbots.com/server.php/Server%20update/index.php?emai
l=USER@DOMAIN.com')
```

Safe anchors

```
def url_anchor(url):
  from urllib.parse import urlparse,urlencode,urlsplit, urlunsplit
  from bs4 import BeautifulSoup
  import requests
  import re
  !pip install tldextract
  import tldextract
  Null_format = ["", "#", "#nothing", "#doesnotexist", "#null", "#void",
"#whatever",
               "#content", "javascript::void(0)", "javascript::void(0);",
"javascript::;", "javascript"]

  def is_URL_accessible(url):
    page = None
    try:
        page = requests.get(url, timeout=5)
    except:
        parsed = urlparse(url)
        url = parsed.scheme+'://'+parsed.netloc
        if not parsed.netloc.startswith('www'):
            url = parsed.scheme+'://www.'+parsed.netloc
            try:
                page = requests.get(url, timeout=5)
            except:
                page = None
                pass
    if page and page.status_code == 200 and page.content not in ["b''", "b'
'"]:
        return True, url, page
    else:
        return False, None, None

  state, iurl, page = is_URL_accessible(url)


  def get_domain(url):
      o = urlsplit(url)
      return o.hostname, tldextract.extract(url).domain, o.path


  if state:
        content = page.content
        hostname, domain, path = get_domain(url)


  Anchor = {'safe':[], 'unsafe':[], 'null':[]}


  def anchor(Anchor, domain, hostname):
      soup = BeautifulSoup(content, 'html.parser', from_encoding='iso-8859-1')
      for href in soup.find_all('a', href=True):
        dots = [x.start(0) for x in re.finditer('\.', href['href'])]
        if hostname in href['href'] or domain in href['href'] or len(dots) == 1
or not href['href'].startswith('http'):
            if "#" in href['href'] or "javascript" in href['href'].lower() or
"mailto" in href['href'].lower():
                 Anchor['unsafe'].append(href['href'])
        else:
            Anchor['safe'].append(href['href'])


  anchor(Anchor, domain, url)


  def safe_anchor(Anchor):
      total = len(Anchor['safe']) +  len(Anchor['unsafe'])
      unsafe = len(Anchor['unsafe'])
      try:
        percentile = unsafe / float(total) * 100
      except:
        return 0
      return percentile
  #print(safe_anchor(Anchor))
  if safe_anchor(Anchor) < 31:
       return -1
  elif safe_anchor(Anchor) >= 31 and safe_anchor(Anchor) <= 67:
       return 0
  else:
       return 1

url_anchor('https://www.xiaoji.com/')
Collecting tldextract
  Downloading tldextract-3.1.2-py2.py3-none-any.whl (87 kB)
?25l
K     |████                              | 10 kB 21.4 MB/s eta 0:00:01
K     |█████████                         | 20 kB 10.9 MB/s eta 0:00:01
K     |█████████████                     | 30 kB 9.0 MB/s eta 0:00:01
K     |██████████████████                | 40 kB 8.2 MB/s eta 0:00:01
K     |███████████████████████           | 51 kB 4.0 MB/s eta 0:00:01
K     |███████████████████████████       | 61 kB 4.4 MB/s eta 0:00:01
K     |████████████████████████████████  | 71 kB 4.6 MB/s eta 0:00:01
K     |████████████████████████████████████| 81 kB 5.2 MB/s eta 0:00:01
K     |██████████████████████████████████████| 87 kB 2.8 MB/s
?25hRequirement already satisfied: requests>=2.1.0 in
/usr/local/lib/python3.7/dist-packages (from tldextract) (2.23.0)
Requirement already satisfied: idna in /usr/local/lib/python3.7/dist-packages
(from tldextract) (2.10)
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (3.4.0)
Collecting requests-file>=1.4
  Downloading requests_file-1.5.1-py2.py3-none-any.whl (3.7 kB)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(2021.10.8)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
```

```
(1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(3.0.4)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from requests-file>=1.4->tldextract) (1.15.0)
Installing collected packages: requests-file, tldextract
Successfully installed requests-file-1.5.1 tldextract-3.1.2

-1
```

Module4

```
def links_tag(url):
  from urllib.parse import urlparse,urlencode,urlsplit, urlunsplit
  from bs4 import BeautifulSoup
  import requests
  import re
  !pip install tldextract
  import tldextract
  Null_format = ["", "#", "#nothing", "#doesnotexist", "#null", "#void",
"#whatever",
               "#content", "javascript::void(0)", "javascript::void(0);",
"javascript::;", "javascript"]

  def is_URL_accessible(url):
    page = None
    try:
        page = requests.get(url, timeout=5)
    except:
        parsed = urlparse(url)
        url = parsed.scheme+'://'+parsed.netloc
        if not parsed.netloc.startswith('www'):
            url = parsed.scheme+'://www.'+parsed.netloc
            try:
                page = requests.get(url, timeout=5)
            except:
                page = None
                pass
    if page and page.status_code == 200 and page.content not in ["b''", "b'
'"]:
        return True, url, page
    else:
        return False, None, None

  state, iurl, page = is_URL_accessible(url)

  def get_domain(url):
      o = urlsplit(url)
      return o.hostname, tldextract.extract(url).domain, o.path

  if state:
        content = page.content
        hostname, domain, path = get_domain(url)

  Link = {'internals':[], 'externals':[], 'null':[]}

  def find_links(Link, domain, hostname):
    soup = BeautifulSoup(content, 'html.parser', from_encoding='iso-8859-1')
    for link in soup.findAll('link', href=True):
        dots = [x.start(0) for x in re.finditer('\.', link['href'])]
        if hostname in link['href'] or domain in link['href'] or len(dots) == 1
or not link['href'].startswith('http'):
            if not link['href'].startswith('http'):
                if not link['href'].startswith('/'):
                    Link['internals'].append(hostname+'/'+link['href'])
                elif link['href'] in Null_format:
                    Link['null'].append(link['href'])
                else:
                    Link['internals'].append(hostname+link['href'])
        else:
            Link['externals'].append(link['href'])

    for script in soup.find_all('script', src=True):
        dots = [x.start(0) for x in re.finditer('\.', script['src'])]
        if hostname in script['src'] or domain in script['src'] or len(dots) ==
1 or not script['src'].startswith('http'):
            if not script['src'].startswith('http'):
                if not script['src'].startswith('/'):
                    Link['internals'].append(hostname+'/'+script['src'])
                elif script['src'] in Null_format:
                    Link['null'].append(script['src'])
                else:
                    Link['internals'].append(hostname+script['src'])
        else:
            Link['externals'].append(link['href'])

  def links_in_tags(Link):
    total = len(Link['internals']) +  len(Link['externals'])
    internals = len(Link['internals'])
    try:
        percentile = internals / float(total) * 100
    except:
        return 0
    return percentile
  #print(links_in_tags(Link))
  if links_in_tags(Link) < 17:
    return -1
  elif links_in_tags(Link) >= 17 and links_in_tags(Link) <= 81:
    return 0
  else:
    return 1
links_tag('https://www.xiaoji.com/')
Requirement already satisfied: tldextract in /usr/local/lib/python3.7/dist-
packages (3.1.2)
Requirement already satisfied: requests-file>=1.4 in
/usr/local/lib/python3.7/dist-packages (from tldextract) (1.5.1)
Requirement already satisfied: requests>=2.1.0 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (2.23.0)
Requirement already satisfied: idna in /usr/local/lib/python3.7/dist-packages
(from tldextract) (2.10)
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (3.4.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
```

```
(2021.10.8)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(3.0.4)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from requests-file>=1.4->tldextract) (1.15.0)

-1
```

###SFH

```
def sfh(url):
  from urllib.parse import urlparse,urlencode,urlsplit, urlunsplit
  from bs4 import BeautifulSoup
  import requests
  import re
  !pip install tldextract
  import tldextract
  Null_format = ["", "#", "#nothing", "#doesnotexist", "#null", "#void",
"#whatever",
               "#content", "javascript::void(0)", "javascript::void(0);",
"javascript::;", "javascript"]

  def is_URL_accessible(url):
    page = None
    try:
        page = requests.get(url, timeout=5)
    except:
        parsed = urlparse(url)
        url = parsed.scheme+'://'+parsed.netloc
        if not parsed.netloc.startswith('www'):
            url = parsed.scheme+'://www.'+parsed.netloc
            try:
                page = requests.get(url, timeout=5)
            except:
                page = None
                pass
    if page and page.status_code == 200 and page.content not in ["b''", "b'
'"]:
        return True, url, page
    else:
        return False, None, None

  state, iurl, page = is_URL_accessible(url)

  def get_domain(url):
      o = urlsplit(url)
      return o.hostname, tldextract.extract(url).domain, o.path

  if state:
        content = page.content
        hostname, domain, path = get_domain(url)

  Form = {'internals':[], 'externals':[], 'null':[]}

  def findForm(Form, domain, hostname):
    for form in soup.findAll('form', action=True):
      dots = [x.start(0) for x in re.finditer('\.', form['action'])]
      if hostname in form['action'] or domain in form['action'] or len(dots) ==
1 or not form['action'].startswith('http'):
          if not form['action'].startswith('http'):
              if not form['action'].startswith('/'):
                  Form['internals'].append(hostname+'/'+form['action'])
              elif form['action'] in Null_format or form['action'] ==
'about:blank':
                  Form['null'].append(form['action'])
              else:
                  Form['internals'].append(hostname+form['action'])
      else:
          Form['externals'].append(form['action'])

  def sf(hostname, Form):
    if len(Form['null'])==0:
        return 1
    elif len(Form['null'])>0:
        return 0
    else:
      return -1
  return(sf(hostname, Form))
sfh('https://stackoverflow.com/questions/59020008/how-to-import-functions-of-a-
jupyter-notebook-into-another-jupyter-notebook-in-g')
Requirement already satisfied: tldextract in /usr/local/lib/python3.7/dist-
packages (3.1.2)
Requirement already satisfied: requests>=2.1.0 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (2.23.0)
Requirement already satisfied: requests-file>=1.4 in
/usr/local/lib/python3.7/dist-packages (from tldextract) (1.5.1)
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (3.4.0)
Requirement already satisfied: idna in /usr/local/lib/python3.7/dist-packages
(from tldextract) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(2021.10.8)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(1.24.3)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from requests-file>=1.4->tldextract) (1.15.0)

1
```

Submitting to emial

```
def sub_email(url):
  from urllib.parse import urlparse,urlencode,urlsplit, urlunsplit
  from bs4 import BeautifulSoup
  import requests
  import re
  !pip install tldextract
  import tldextract
  Null_format = ["", "#", "#nothing", "#doesnotexist", "#null", "#void",
"#whatever",
               "#content", "javascript::void(0)", "javascript::void(0);",
"javascript::;", "javascript"]
```

```
  def is_URL_accessible(url):
    page = None
    try:
        page = requests.get(url, timeout=5)
    except:
        parsed = urlparse(url)
        url = parsed.scheme+'://'+parsed.netloc
        if not parsed.netloc.startswith('www'):
            url = parsed.scheme+'://www.'+parsed.netloc
            try:
                page = requests.get(url, timeout=5)
            except:
                page = None
                pass
    if page and page.status_code == 200 and page.content not in ["b''", "b'
'"]:
        return True, url, page
    else:
        return False, None, None

  state, iurl, page = is_URL_accessible(url)

  def get_domain(url):
      o = urlsplit(url)
      return o.hostname, tldextract.extract(url).domain, o.path

  if state:
        content = page.content
        hostname, domain, path = get_domain(url)

  Form = {'internals':[], 'externals':[], 'null':[]}
  def findForm(Form, domain, hostname):
    for form in soup.findAll('form', action=True):
      dots = [x.start(0) for x in re.finditer('\.', form['action'])]
      if hostname in form['action'] or domain in form['action'] or len(dots) ==
1 or not form['action'].startswith('http'):
          if not form['action'].startswith('http'):
              if not form['action'].startswith('/'):
                  Form['internals'].append(hostname+'/'+form['action'])
              elif form['action'] in Null_format or form['action'] ==
'about:blank':
                  Form['null'].append(form['action'])
              else:
                  Form['internals'].append(hostname+form['action'])
      else:
          Form['externals'].append(form['action'])

  def submitting_to_email(Form):
    # print(Form)
    for form in (Form['internals'] + Form['externals']):
        #print('inside for')
        if "mailto:" in form or "mail()" in form:
            return 1
        else:
            return -1
    return 1

  return(submitting_to_email(Form))
# print('ans is ',  sub_email('https://www.xiaoji.com/'))
```

On Mouse-over

```
def mouse_over(url):
  from urllib.parse import urlparse,urlencode,urlsplit, urlunsplit
  from bs4 import BeautifulSoup
  import requests
  import re
  !pip install tldextract
  import tldextract
  Null_format = ["", "#", "#nothing", "#doesnotexist", "#null", "#void",
"#whatever",
               "#content", "javascript::void(0)", "javascript::void(0);",
"javascript::;", "javascript"]

  def is_URL_accessible(url):
    page = None
    try:
        page = requests.get(url, timeout=5)
    except:
        parsed = urlparse(url)
        url = parsed.scheme+'://'+parsed.netloc
        if not parsed.netloc.startswith('www'):
            url = parsed.scheme+'://www.'+parsed.netloc
            try:
                page = requests.get(url, timeout=5)
            except:
                page = None
                pass
    if page and page.status_code == 200 and page.content not in ["b''", "b'
'"]:
        return True, url, page
    else:
        return False, None, None

  state, iurl, page = is_URL_accessible(url)

  def get_domain(url):
      o = urlsplit(url)
      return o.hostname, tldextract.extract(url).domain, o.path

  if state:
        content = page.content
        hostname, domain, path = get_domain(url)

  def onmouseover(content):
    if 'onmouseover="window.status=' in str(content).lower().replace(" ",""):
        return 1
    else:
        return -1
  return(onmouseover(content.decode('latin-1')))
mouse_over('https://www.xiaoji.com/')
Requirement already satisfied: tldextract in /usr/local/lib/python3.7/dist-
packages (3.1.2)
Requirement already satisfied: idna in /usr/local/lib/python3.7/dist-packages
(from tldextract) (2.10)
Requirement already satisfied: requests>=2.1.0 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (2.23.0)
```

```
Requirement already satisfied: requests-file>=1.4 in
/usr/local/lib/python3.7/dist-packages (from tldextract) (1.5.1)
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (3.4.0)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(2021.10.8)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from requests-file>=1.4->tldextract) (1.15.0)

-1
```

Right Click

```
def right_click(url):
  from urllib.parse import urlparse,urlencode,urlsplit, urlunsplit
  from bs4 import BeautifulSoup
  import requests
  import re
  !pip install tldextract
  import tldextract
  Null_format = ["", "#", "#nothing", "#doesnotexist", "#null", "#void",
"#whatever",
               "#content", "javascript::void(0)", "javascript::void(0);",
"javascript::;", "javascript"]

  def is_URL_accessible(url):
    page = None
    try:
        page = requests.get(url, timeout=5)
    except:
        parsed = urlparse(url)
        url = parsed.scheme+'://'+parsed.netloc
        if not parsed.netloc.startswith('www'):
            url = parsed.scheme+'://www.'+parsed.netloc
            try:
                page = requests.get(url, timeout=5)
            except:
                page = None
                pass
    if page and page.status_code == 200 and page.content not in ["b''", "b'
'"]:
        return True, url, page
    else:
        return False, None, None

  state, iurl, page = is_URL_accessible(url)


  def get_domain(url):
      o = urlsplit(url)
      return o.hostname, tldextract.extract(url).domain, o.path


  if state:
        content = page.content
        hostname, domain, path = get_domain(url)
  def right_clic(content):
    if re.findall(r"event.button ?== ?2", content.decode('latin-1')):
        return 1
    else:
        return -1

  return right_clic(content)
right_click('http://walletconnectbits.com/')
Requirement already satisfied: tldextract in /usr/local/lib/python3.7/dist-
packages (3.1.2)
Requirement already satisfied: filelock>=3.0.8 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (3.4.0)
Requirement already satisfied: idna in /usr/local/lib/python3.7/dist-packages
(from tldextract) (2.10)
Requirement already satisfied: requests>=2.1.0 in /usr/local/lib/python3.7/dist-
packages (from tldextract) (2.23.0)
Requirement already satisfied: requests-file>=1.4 in
/usr/local/lib/python3.7/dist-packages (from tldextract) (1.5.1)
```

```
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests>=2.1.0->tldextract)
(2021.10.8)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from requests-file>=1.4->tldextract) (1.15.0)
```

```
-1
```

Domain Age

```python
def domain_age(domain):
    import json
    import requests
    url = domain.split("//")[-1].split("/")[0].split('?')[0]
    show = "https://input.payapi.io/v1/api/fraud/domain/age/" + url
    r = requests.get(show)

    if r.status_code == 200:
        data = r.text
        jsonToPython = json.loads(data)
        result = jsonToPython['result']
        if result/30 >= 6:
            return -1
        else:
            return 1
    else:
        return -1
domain_age('http://walletconnectbits.com/')
```

```
1
```

DNS Record

```python
def dns_record(domain):
    !pip install dnspython
    import dns.resolver
    try:
        nameservers = dns.resolver.resolve(domain,'NS')
        if len(nameservers) > 0:
            #print('len is', nameservers)
            return -1
        else:
            return 1
    except:
        return 1
# dns_record('http://postdebanks.com/DIE/POST/diepost/')
```

###Website Traffic

```
def web_traffic(url):
  from urllib.parse import quote
  from urllib.request import urlopen
  from bs4 import BeautifulSoup
  try:
    #Filling the whitespaces in the URL if any
    url = quote(url)
    rank = BeautifulSoup(urlopen("http://data.alexa.com/data?cli=10&dat=s&url="
+ url).read(), "xml").find(
        "REACH")['RANK']
    rank = int(rank)
  except TypeError:
        return 1
  if rank <= 100000:
    return -1
  elif rank > 100000:
    return 0
  else:
    return 1
#web_traffic('http://postdebanks.com/DIE/POST/diepost/')
```

Page Rank

```
def page_rank(domain):
    import requests
    key = '8o0gg0804g4k0gwkk4oocws04oc0sg88gg844o4k'
    url = 'https://openpagerank.com/api/v1.0/getPageRank?domains%5B0%5D=' +
str(domain)
    try:
        request = requests.get(url, headers={'API-OPR': key})
        result = request.json()
        result = result['response'][0]['page_rank_integer']
        if result / 10 < 0.2 :
            return 1
        else:
            return -1
    except:
        return 1
#page_rank('http://postdebanks.com/DIE/POST/diepost/')
```

Google Index

```
def google_index(url):
    from urllib.parse import urlencode
    from bs4 import BeautifulSoup
    import requests
    #time.sleep(.6)
    user_agent =  'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/48.0.2564.116 Safari/537.36'
    headers = {'User-Agent' : user_agent}
    query = {'q': 'site:' + url}
    google = "https://www.google.com/search?" + urlencode(query)
    data = requests.get(google, headers=headers)
    data.encoding = 'ISO-8859-1'
    soup = BeautifulSoup(str(data.content), "html.parser")
    try:
      if 'Our systems have detected unusual traffic from your computer network.'
in str(soup):
        return -1
      check = soup.find(id="rso").find("div").find("div").find("a")
      #print(soup.prettify())
      if check and check['href']:
        return -1
      else:
        return 1

    except AttributeError:
        return 1
#google_index('http://walletconnectbits.com/')
```

Links pointing to page (No of Hyperlinks)

```
def urlsCount(url):
  from bs4 import BeautifulSoup
  from collections import Counter
  import requests
  soup = BeautifulSoup(requests.get(url).text, "html.parser")
  foundUrls = Counter([link["href"] for link in soup.find_all("a", href=lambda
href: href and not href.startswith("#"))])
  count = len(foundUrls)
  if(count == 0):
    return 1
  elif(count > 0 and count <= 2):
    return 0
  else:
    return -1
#urlsCount('https://www.hokabutypolska.pl/')
```

statistical report

```
def statistical_report(url, domain):
    import re
    import socket

url_match=re.search('at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|ho
l\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.ly',url)
    try:
        ip_address=socket.gethostbyname(domain)

ip_match=re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.1
85\.217\.116|78\.46\.211\.158|181\.174\.165\.13|46\.242\.145\.103|121\.50\.168\.
40|83\.125\.22\.219|46\.242\.145\.98|'

'107\.151\.148\.44|107\.151\.148\.107|64\.70\.19\.203|199\.184\.144\.27|107\.151
\.148\.108|107\.151\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|21
6\.58\.192\.225|'

'118\.184\.25\.86|67\.208\.74\.71|23\.253\.126\.58|104\.239\.157\.210|175\.126\.
123\.219|141\.8\.224\.221|10\.10\.10\.10|43\.229\.108\.32|103\.232\.215\.140|69\
.172\.201\.153|'

'216\.218\.185\.162|54\.225\.104\.146|103\.243\.24\.98|199\.59\.243\.120|31\.170
\.160\.61|213\.19\.128\.77|62\.113\.226\.131|208\.100\.26\.234|195\.16\.127\.102
|195\.16\.127\.157|'

'34\.196\.13\.28|103\.224\.212\.222|172\.217\.4\.225|54\.72\.9\.51|192\.64\.147\
.141|198\.200\.56\.183|23\.253\.164\.103|52\.48\.191\.26|52\.214\.197\.72|87\.98
\.255\.18|209\.99\.17\.27|'

'216\.38\.62\.18|104\.130\.124\.96|47\.89\.58\.141|78\.46\.211\.158|54\.86\.225\
.156|54\.82\.156\.19|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42',ip_addr
ess)
        if url_match or ip_match:
            return 1
        else:
            return -1
    except:
        return 1
# statistical_report('http://postdebanks.com/DIE/POST/diepost/',
'postdebanks.com')
```

Input features

```
def getInput(url):
  from urllib.parse import urlparse
  domain = urlparse(url).netloc
  input = []
  print(domain)
  input.append(having_ip_address(url))
  input.append(URL_Length(url))
  input.append(haveAtSign(url))
  input.append(prefixSuffix(url))
  input.append(sub_domain_count(url))
  input.append(sslVerify(url))
  input.append(port(domain))
  input.append(request_url(url))
  input.append(url_anchor(url))
  input.append(links_tag(url))
  input.append(sfh(url))
  input.append(sub_email(url))
  input.append(mouse_over(url))
  input.append(right_click(url))
  input.append(domain_age(url))
  input.append(dns_record(url))
  input.append(web_traffic(url))
  input.append(page_rank(domain))
  input.append(google_index(url))
  input.append(urlsCount(url))
  input.append(statistical_report(url, domain))
  return (input)
input = getInput('https://post-u8719-sufficient-
159.sites.qsandbox.com/swisse/Home/')
input
[-1, 0, -1, 1, 1, 0, 1, -1, 1, -1, 1, -1, -1, -1, -1, 1, 0, 1, 1, 1, -1]
[input]
[[-1, 0, -1, 1, 1, 0, 1, -1, 1, -1, 1, -1, -1, -1, -1, 1, 0, 1, 1, 1, -1]]
import pickle
filename = '/content/finalized_model.sav'
loaded_model = pickle.load(open(filename, 'rb'))
import warnings
warnings.filterwarnings("ignore")
result = loaded_model.predict([input])
if result == -1:
  print("A legitimate website")
else:
  print("A Phishing website!!")
A Phishing website!!
```

8. RESULTS

Although the proposed method in this paper has achieved some good results, there are still some shortcomings. The main disadvantage is that it takes longer to train. However, the trained model is better than the others in terms of accuracy of phishing website detection. Another disadvantage is that the model cannot determine whether the URL is active or not, so it is necessary to test whether the URL is active or not before detection to ensure the effectiveness of detection. In addition, some attackers use URLs that are not imitations of other websites, and such URLs will not be detected. The next step of our work aims to use new techniques to automatically extract other features for detecting phishing sites, such as web code features, web text features, and web icon features.

9. ADVANTAGES

- Build secure connection between user's mail transfer agent(MTA) and Mail User Agent(MUA)
- Provide clear idea about the effective level of each classifier on phishing email detection.
- High level of accuracy by take the advantages of many classifiers.
- Create a new type of features like Markov Features.

DISADVANTAGES

- **Time Consuming**
- **Memory Consuming**
- **Huge number of features**
- **Non Standard classifier**
- **Time consuming because this technique has many layers to make the final result.**
- **Many algorithm for classification which mean time consuming.**
- **Higher cost need large mail server and high memory requirement.**

10. CONCLUSION

Education awareness is the most significant strategy to protect users from phishing attacks. Internet users should be aware of all security recommendations made by professionals. Every user should also be taught not to mindlessly follow links to websites where sensitive information must be entered. Before visiting a website, make sure to check the URL. In the future, the system could be upgraded to automatically detect the web page and the application's compatibility with the web browser. Additional work can be done to distinguish fraudulent web pages from authentic web pages by adding certain additional characteristics. In order to detect phishing on the mobile platform, the PhishChecker programme can be upgraded into a web phone application.

11. FUTURE SCOPE

We'll look into the links between phishing sites and hosting and DNS registration providers in more detail. We'll also look at other features like Content Security Policies, certificate authorities, and TLS fingerprinting that can be used. In addition, we will compare SVMs and neural networks to other machine learning techniques such as random forest classifiers for speed and accuracy.

12. APPENDIX

12.1 Github &Project Demo Link

https://youtu.be/04FB-R_eYE