A PROJECT REPORT ON

WEB PHISHING DETECTION

Domain : APPLIED DATA SCIENCE

Team ID : PNT2022TMID01355

Dharanya S(211419205040)

Department of Information Technology

Panimalar Engineering College

Kiruthika S(211419205094)

Department of Information Technology

Panimalar Engineering College

Nithya S(211419205120)

Department of Information Technology

Panimalar Engineering College

Poornima B(211419205128)

Department of Information Technology

Panimalar Engineering College

Project Report Format

1. INTRODUCTION

a. Project Overview

Phishing is a form of fraud in which an attacker masquerades as a reputable entity or person in email or other forms of communication. Attackers will commonly use phishing emails to distribute malicious links or attachments that can perform a variety of functions. Some will extract login credentials or account information from victims.

Phishers can use public sources of information to gather background information about the victim's personal and work history, interests and activities. Typically through social networks like LinkedIn, Facebook and Twitter. These sources are normally used to uncover information such as names, job titles and email addresses of potential victims. This information can then be used to craft a believable email.

This Guided Project mainly focuses on applying a machine learning algorithm to detect Phishing websites.

b. Purpose

- Measure the degrees of corporate and employee vulnerability.
- Eliminate the cyber threat risk level.
- Increase user alertness to phishing risks.
- Install a cyber security culture and create cyber security heroes.
- Large organizations may get trapped in different kinds of scams.

2. LITERATURE SURVEY

a. Existing problem

Websites phishing is a cyber-attack that targets online users to steal their sensitive information including login credentials and banking details. Attackers fool the users by presenting the masked webpage as a legitimate or trustworthy to retrieve their essential data.

b. References

1 .Title: **Detection of Phishing Websites by Using Machine Learning-Based URL Analysis.**

Author: Mehmet Korkmaz, Ozgur KoraySahingoz, BanuDiri.

Year: 2020

Techniques Used: XGBOOST, RF , LR ,KNN,SVM,DTANN,NB .

Description: A machine learning-based phishing detection system by using eight different algorithms to analyze the URLs, and three different datasets to compare the results with other works. The experimental results depict that the proposed models have an outstanding performance with a success rate.

2 . Title: **A Deep Learning-Based Framework for Phishing Website Detection**

Author: Lizhen Tang , Qusay H. Mahmoud

Year: 2021

Techniques Used: RNN-GRU, web browser extension.

Description: The author briefed that they have implemented the framework as a browser plugin capable of determining whether there is a phishing risk in real-time when the user visits a web page and gives a warning messages .It combines the multiple strategies to improve accuracy, reduce false alarm rates, and reduce calculation time, including whitelist filtering, blacklist interception, and machine learning (ML) prediction.

3. Title: **Detection of Phishing Websites from URLs by using Classification Techniques on WEKA**

Author: BuketGeyik, Kubra Erensoy, EmreKocyigit

Year: 2021

Techniques Used: machine learning, classification algorithms, phishing detection, cybersecurity

Description: The anti-phishing method has been developed by detecting the attacks made with the technologies used. we combined the websites used by phishing attacks into a dataset, then we obtained some results using 4 classification algorithms with this dataset.

C. Problem Statement Definition

INPUT: A Web URL suspected to the phishing website.

OUTPUT: Yes \No

Yes means URL is an phishing URL

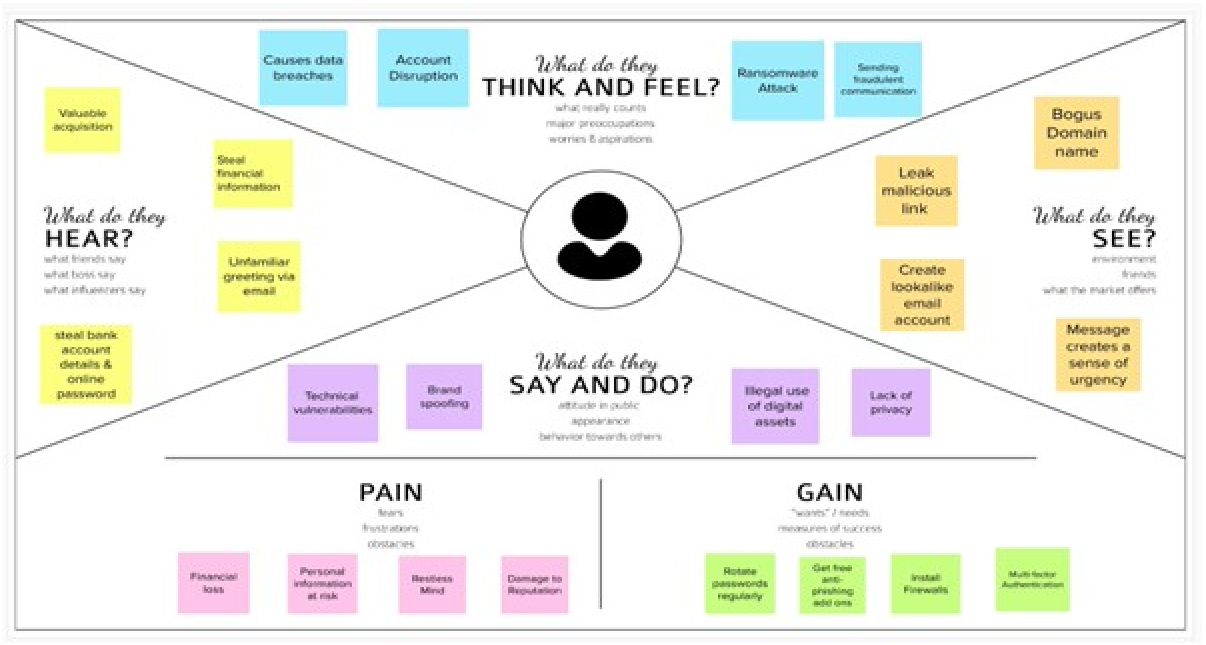
No means URL is an legitimate URL.

GOAL: our goal is to find the URL is an phishing or legitimate URL by using machine learning algorithms.

3. IDEATION & PROPOSED SOLUTION

a. Empathy Map Canvas

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.



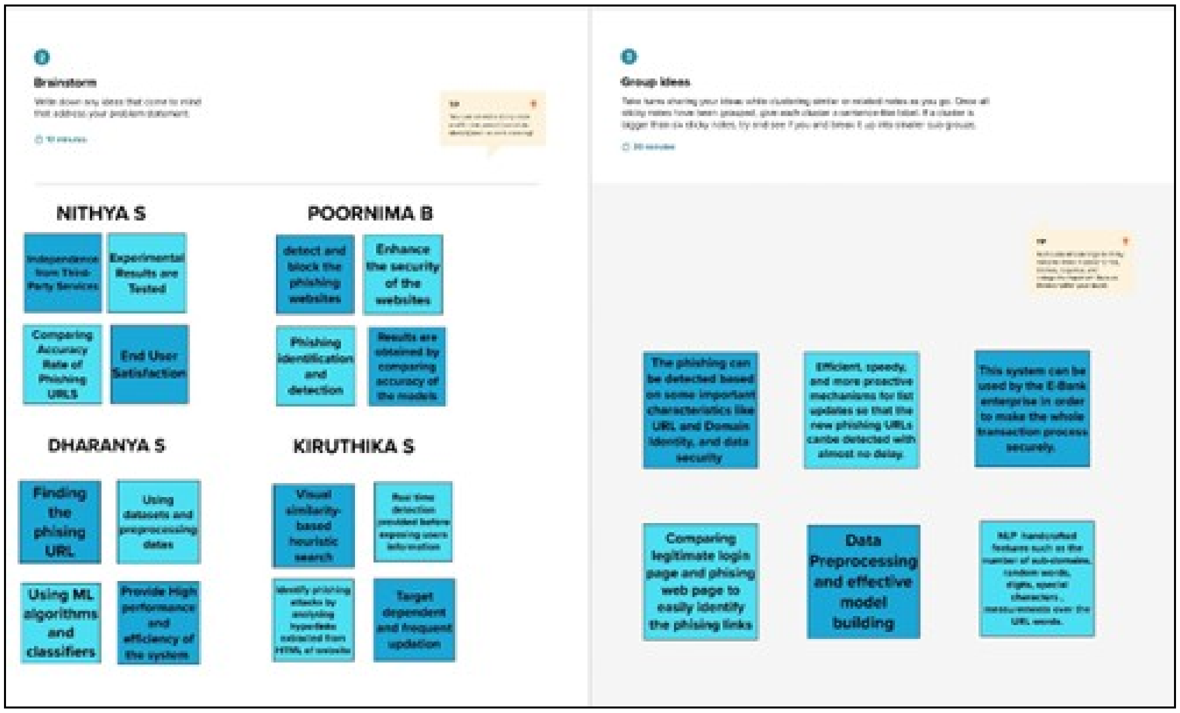
b. **Ideation & Brainstorming**

Brainstorm, Idea Listing and Grouping:

Brainstorm is nothing but to suggest idea for the project before starting the project. The process of brainstorming can assist the group focus its ideas and find solutions.

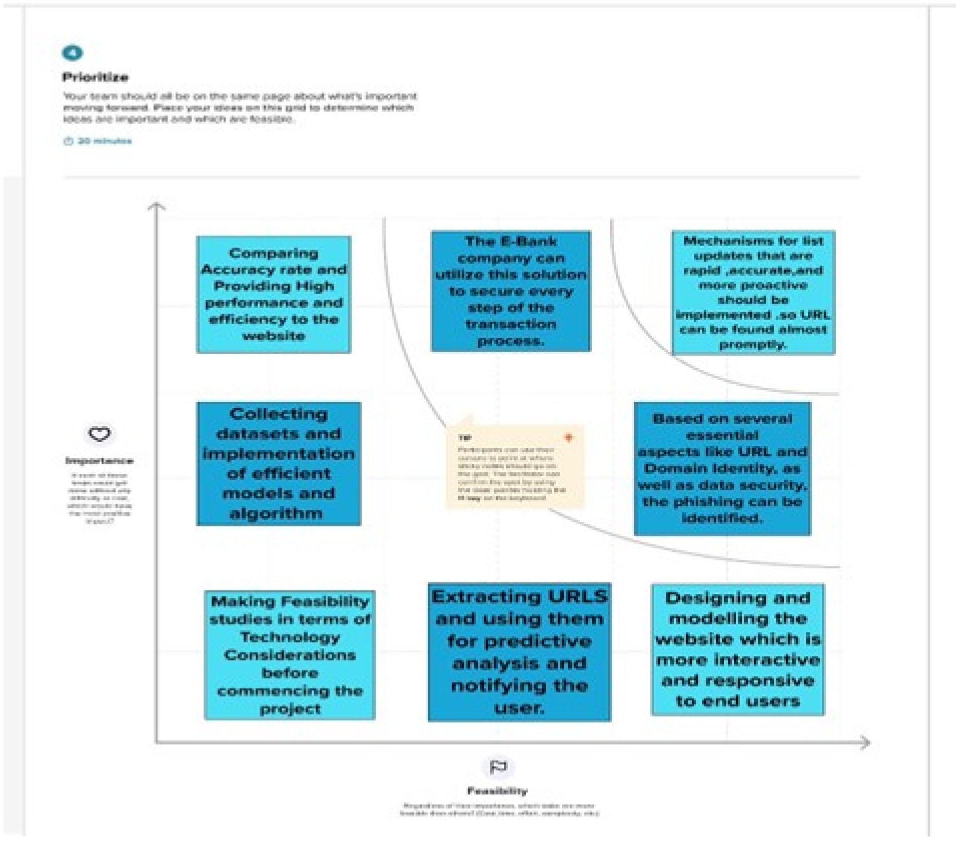
Project Ideas are where you begin documenting proposals for future research grant applications. During this stage, you are recording important project-related details as well as locating collaborators, potential funders, budget details, and project-related metadata. You can also make tasks and assign them to projectrelated people.

An administrative grouping of projects is known as a project group. Project groups make it possible for administrative operations to have an impact on several projects and users at once.


Brainstorm
Write down any ideas that come to mind that address your problem statement.
Group ideas
NITHYA S
Independence from Third-Party Services
Experimental Results are Tested
Comparing Accuracy Rate of Phishing URLS
End User Satisfaction
POORNIMA B
detect and block the phishing websites
Enhance the security of the websites
Phishing identification and detection
Results are obtained by comparing accuracy of the models
DHARANYA S
Finding the phising URL
Using datasets and preprocessing datas
Using ML algorithms and classifiers
Provide High performance and efficiency of the system
KIRUTHIKA S
Visual similarity-based heuristic search
Real time detection provided before exposing users information
Identify phishing attacks by analysing hyperlinks extracted from HTML of website
Target dependent and frequent updation
The phishing can be detected based on some important characteristics like URL and Domain Identity, and data security
Efficient, speedy, and more proactive mechanisms for list updates so that the new phishing URLs canbe detected with almost no delay
This system can be used by the E-Bank enterprise in order to make the whole transaction process securely.
Comparing legitimate login page and phising web page to easily identify the phising links
Data Preprocessing and effective model building
NLP handcrafted features such as the number of sub-domains, random words, digits, special characters , measurements over the URL words.

Idea prioritization:

Only a small portion of the idea management process involves idea prioritization. It takes time to develop an organized idea management strategy and a methodical approach to gathering, analyzing, and prioritizing new ideas.


Prioritize
Comparing Accuracy rate and Providing High performance and efficiency to the website
The E-Bank company can utilize this solution to secure every step of the transaction process.
Mechanisms for list updates that are rapid ,accurate,and more proactive should be implemented .so URL can be found almost promptly.
Importance
Collecting datasets and implementation of efficient models and algorithm
TIP
Based on several essential aspects like URL and Domain Identity, as well as data security, the phishing can be identified.
Making Feasibility studies in terms of Technology Considerations before commencing the project
Extracting URLS and using them for predictive analysis and notifying the user.
Designing and modelling the website which is more interactive and responsive to end users
Feasibility

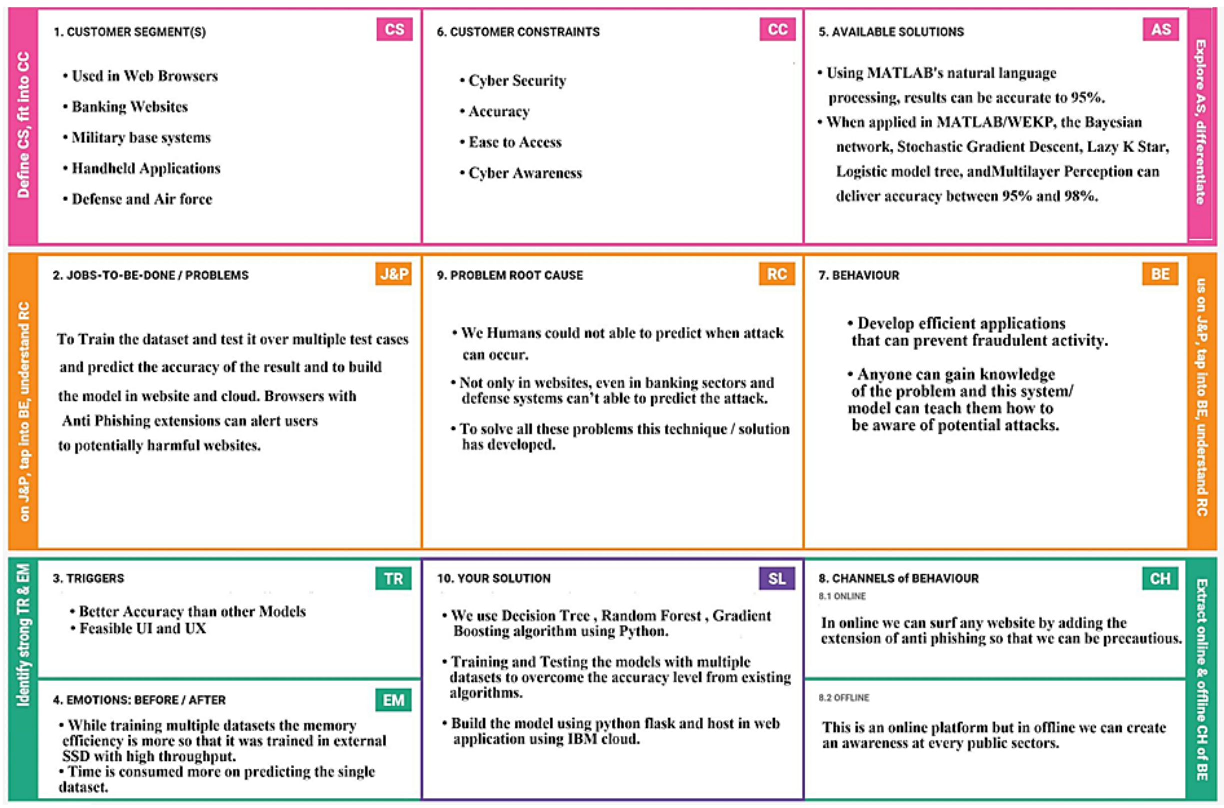
C. Proposed Solution

Proposed Solution refers to the technical response that the Implementation agency will offer in response to the Project's requirements and objectives .The proposed solution of our project is given below

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The fact that notable hackers are continuously coming up with new ways to overcome our barriers presents a uniquedifficulty in this field.The problem with the currenttechnologies is that they occasionally discover small falsepositive and false negative results. By adding a significantly improved feature to the machine learning algorithm, which wouldproduce significantly higheraccuracy, these drawbacks can be eliminated.
2.	Idea / Solution description	We concentrate on adding the project directly to the Browser extension so that if a user clicks on a specific URL and if that URL is a phishing site, a pop-up warning message appears.
3.	Novelty / Uniqueness	a. We created the Web application known as Web PhishingDetection using machine learning. 2. It determines whetheror not the website is harmful by looking at the URL and the number of visitors who have visitedthat specific webpage or website.
4.	Social Impact/ Customer Satisfaction	a. To increase publicawareness of variouscyberattacks, particularly this phishing attack. 2. This paradigm can be unlearned and relearned in numerous parts of data theft and cyber security.
5.	Business Model(Revenue Model)	a. This approach protects the banking and financial industries from data loss and data attack, resulting in no external financial loss. 2. In business organizations, companies can utilize this tool to preventcyberattacks and to put improvements in place forthe next time one occurs.
6.	Scalability of the Solution	a. We provide a good, functional UI/UX design fordetectingweb phishing. 2. In orderto achieve higheraccuracy than otheralgorithms, the model is testedand trained on a variety of datasets.

d. **Problem Solution fit**

Proposed solution fit is nothing but identify an existing problem and to solve it in with a solution that customers find useful and satisfying.



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- Used in Web Browsers
- Banking Websites
- Military base systems
- Handheld Applications
- Defense and Air force

6. CUSTOMER CONSTRAINTS CC

- Cyber Security
- Accuracy
- Ease to Access
- Cyber Awareness

5. AVAILABLE SOLUTIONS AS

- Using MATLAB's natural language processing, results can be accurate to 95%.
- When applied in MATLAB/WEKP, the Bayesian network, Stochastic Gradient Descent, Lazy K Star, Logistic model tree, andMultilayer Perception can deliver accuracy between 95% and 98%.

Explore AS, differentiate

on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

To Train the dataset and test it over multiple test cases and predict the accuracy of the result and to build the model in website and cloud. Browsers with Anti Phishing extensions can alert users to potentially harmful websites.

9. PROBLEM ROOT CAUSE RC

- We Humans could not able to predict when attack can occur.
- Not only in websites, even in banking sectors and defense systems can't able to predict the attack.
- To solve all these problems this technique / solution has developed.

7. BEHAVIOUR BE

- Develop efficient applications that can prevent fraudulent activity.
- Anyone can gain knowledge of the problem and this system/ model can teach them how to be aware of potential attacks.

us on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

- Better Accuracy than other Models
- Feasible UI and UX

4. EMOTIONS: BEFORE / AFTER EM

- While training multiple datasets the memory efficiency is more so that it was trained in external SSD with high throughput.
- Time is consumed more on predicting the single dataset.

10. YOUR SOLUTION SL

- We use Decision Tree , Random Forest , Gradient Boosting algorithm using Python.
- Training and Testing the models with multiple datasets to overcome the accuracy level from existing algorithms.
- Build the model using python flask and host in web application using IBM cloud.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

In online we can surf any website by adding the extension of anti phishing so that we can be precautious.

8.2 OFFLINE

This is an online platform but in offline we can create an awareness at every public sectors.

Extract online & offline CH of BE

4. REQUIREMENT ANALYSIS

a. Functional requirement

The desired operations of a program or system are referred to as functional requirements in software development and systems engineering. Product features or functions must be implemented by developers in order for users to complete their duties. For the development team as well as the stakeholders, it is crucial to make them apparent. Functional requirements often explain how a system will behave under particular circumstances.

FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	**User Input**	User inputs an URL in required fieldto check itsvalidation.
FR-2	**Website identification**	Model compares the websites using Blacklistand Whitelist approach.
FR-3	**Feature extraction**	After comparing, if none foundon comparison then itextracts feature usingheuristic and visualsimilarity approach.
FR-4	**Prediction**	Model predicts the URL usingMachine Learningalgorithms such as Regression, KNN
FR-5	**Classifier**	Model generates the final outcome by predicting all output fromthe classifier.
FR-6	**Results**	Before users submitany personal data into a website, a model predicts it and displays a pop-up.

b. **Non-Functional requirements**

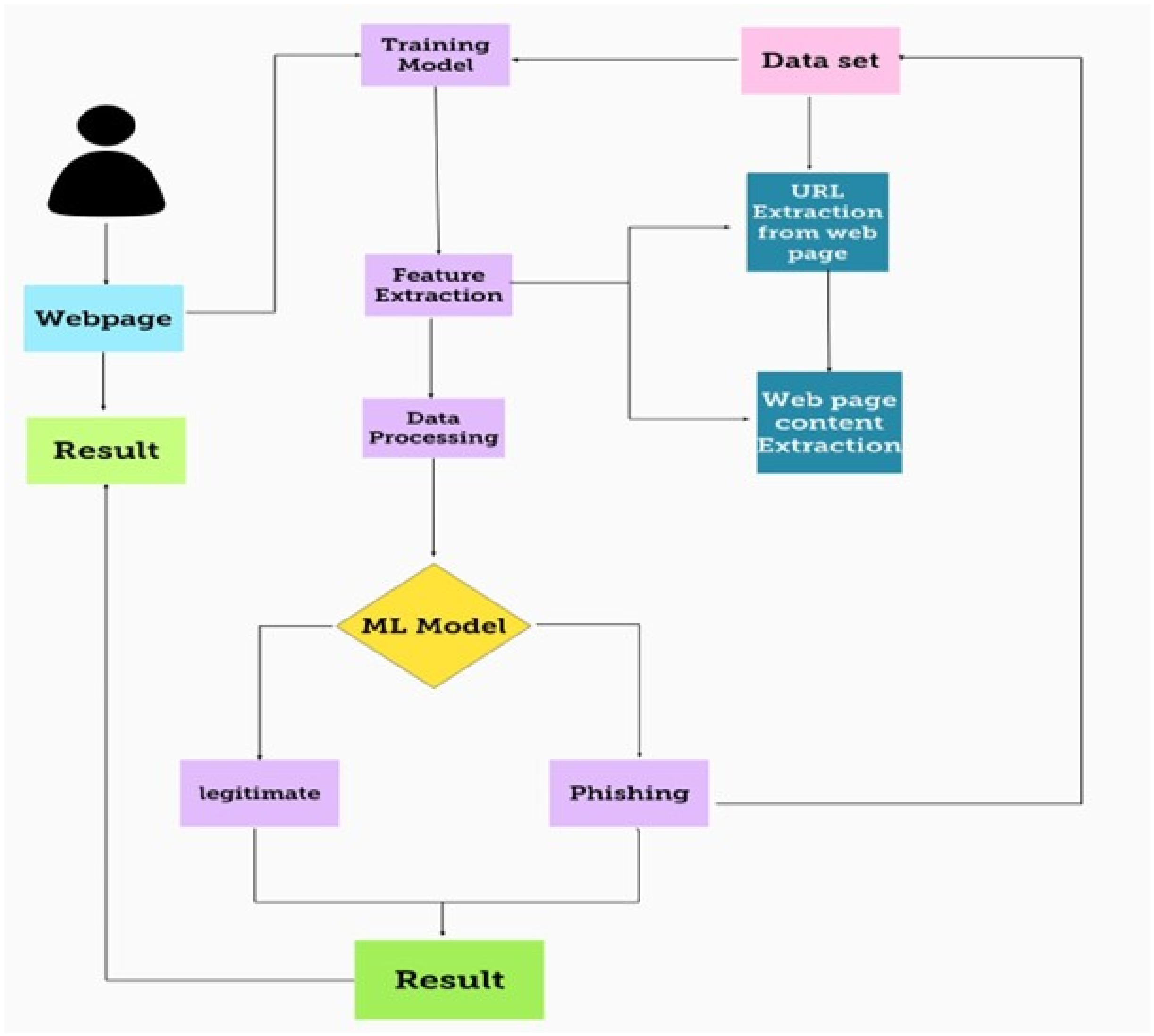
Non-functional requirements list a system's fundamental characteristics. They are sometimes referred to as qualities. It defines characteristics of the system such usability, scalability, maintainability, and performance. They act as limitations or restrictions on how the system is designed for the various backlogs

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	With web phishing detection, users can browse several websites without losing anyinformation.
NFR-2	**Security**	By receiving pop-upmessages, users may determine whether or notthe websites are secure.
NFR-3	**Reliability**	Any user accessing a website should feel confidentin it.
NFR-4	**Performance**	For the performance to be effective, it should be quicker and moreuser-friendly.
NFR-5	**Availability**	Users shouldbe able to access materials that arevalidand credible.
NFR-6	**Scalability**	The website's performance must be effective to handle the growing userand load withoutanyproblems.

5. PROJECT DESIGN

a. Data Flow Diagrams

Data Flow Diagrams It demonstrates the many types of data that will be input into and exported from the system, as well as where the data will be stored. A DFD is frequently an expansion of a context diagram to reveal more of the system's finer details that were initially depicted by the context diagram

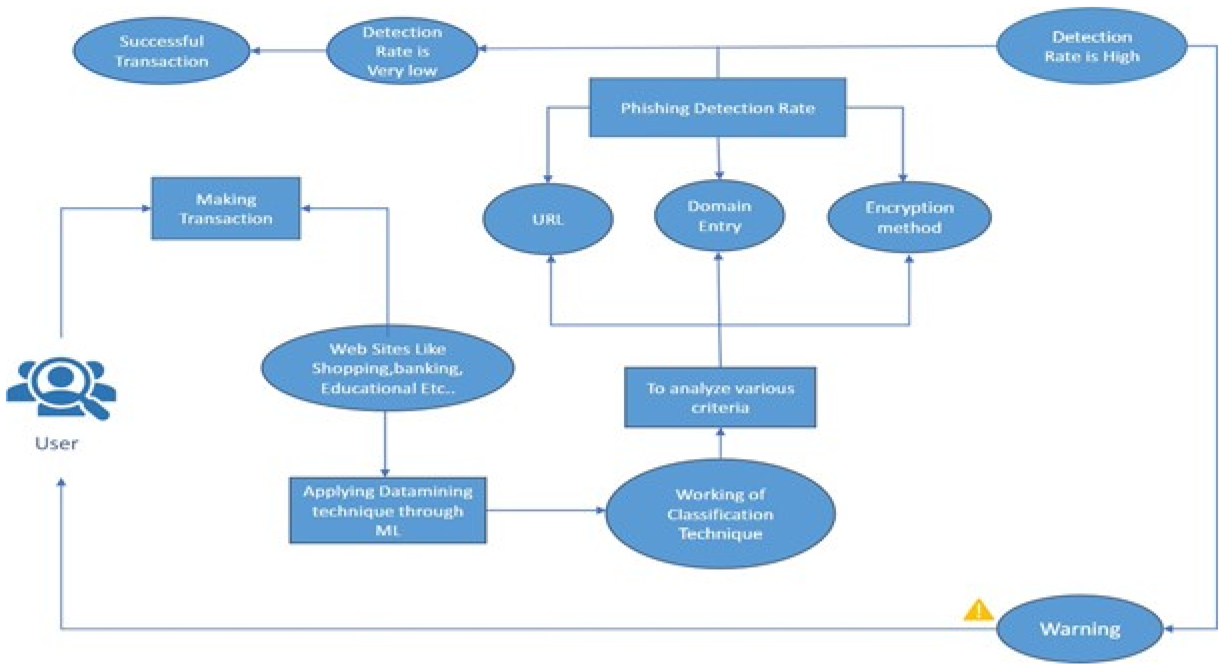


b. **Solution & Technical Architecture**

Solution Architecture

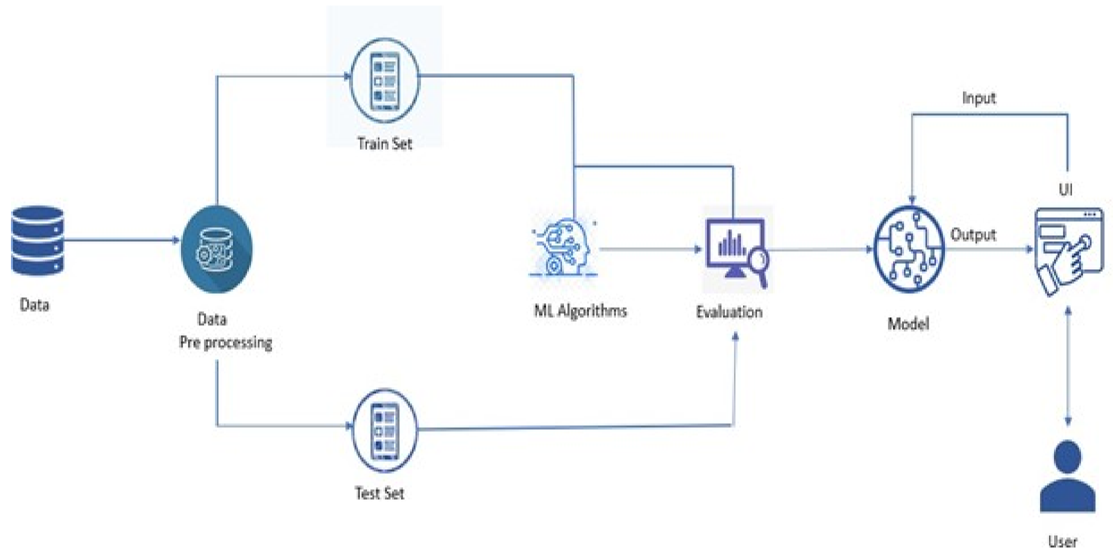
Solution architecture is a complex process – with many sub-processes – that bridgesthe gap betweenbusiness problems and technology solutions. Its goals are to:

- Find the best tech solutionto solve existingbusiness problems.
- Describe the structure, characteristics, behavior, and other aspects of thesoftware to project stakeholders.
- Define features, development phases, and solutionrequirements.
- Provide specifications according to which the solution is defined, managed,and delivered.


Successful Transaction
Detection Rate is Very low
Detection Rate is High
Phishing Detection Rate
Making Transaction
URL
Domain Entry
Encryption method
Web Sites Like Shopping,banking, Educational Etc..
To analyze various criteria
User
Applying Datamining technique through ML
Working of Classification Technique
Warning

Technical Architecture

The main system components, their connections, and the agreements that specify how the components interact are all included in the technical architecture. The objective of technical architects is to fulfil all business requirements with an application that is both performance- and securityoptimized. creating the framework for technological systems. controlling the execution of programmes. collaborating with the software development group to make sure the system functions properly.



c. User Stories

A user story is a casual, all-inclusive description of a software feature written from the viewpoint of the client or end user. A user story's objective is to describe how a piece of work will provide the customer with a specific value. The fact that user stories, unlike requirements or use cases, are not intended to stand alone may be the most significant advantage of using user stories in agile product development. Every user story is instead a standing placeholder for a future discussion with the development team.

Sprint	Functional Requirement (Epic)	User StoryNumber	User Story / Task	Story Points	Priority	Team Members
Sprint-1	User Input	USN-1	User inputsan URL in the required field to checkits validation	2	High	NITHYA S
Sprint-1	Website Comparison	USN-2	Model compares the websites usingBlacklist and Whitelist approach.	1	High	KIRUTHIKA S
Sprint-2	Feature Extraction	USN-3	After comparison, if none found on comparison then it extracts feature using heuristic and visual similarity.	2	Low	POORNIMA B
Sprint-2	Prediction	USN-4	Model predicts the URL usingMachine learning algorithms suchas logistic Regression, KNN.	2	Medium	DHARANYA S
Sprint-3	Classifier	USN-5	Model thendisplays whether the website is legalsiteor a phishing site	1	High	NITHYA S
Sprint-3	Announcement	USN-6	Model thendisplays whether thewebsite is legal site or a phishing site	1	High	DHARANYA S
Sprint-4	Events	USN-7	This model needs the capability of retrieving anddisplaying accurate resultfor a website.	1	High	POORNIMA B

6. PROJECT PLANNING & SCHEDULING

a. Sprint Planning & Estimation

Sprint planning is an event in scrum that kicks off the sprint. The purpose of sprint planning is to define what can be delivered in the sprint and how that work will be achieved. Sprint planning is done in collaboration with the whole scrum team.

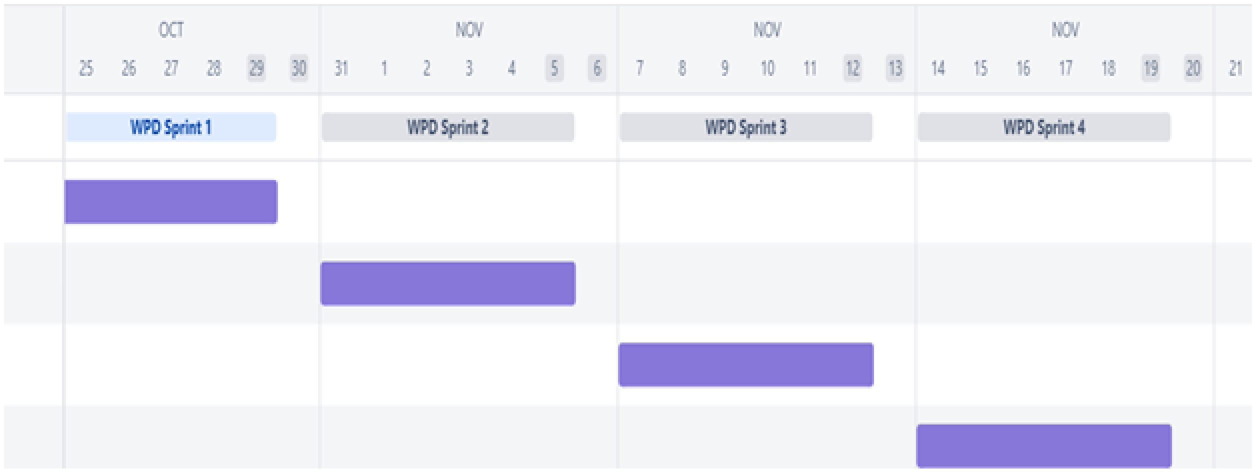
In Scrum Projects, Estimation is done by the entire team during Sprint Planning Meeting. The objective of the Estimation would be to consider the User Stories for the Sprint by Priority and by the Ability of the team to deliver during the Time Box of the Sprint.

b. Sprint Delivery Schedule

A sprint schedule is a written description of the entire sprint planning process. It's one of the initial steps in the agile sprint planning process, and it calls for sufficient investigation, preparation, and coordination. It center on a product backlog, which is a list of open requests for development and iteration. A burndown chart, which displays how rapidly a team is progressing through a customer's user stories, is a project management chart. This agile tool records the description of a feature from the viewpoint of the end user and compares the overall effort to the quantity of work for each agile sprint.

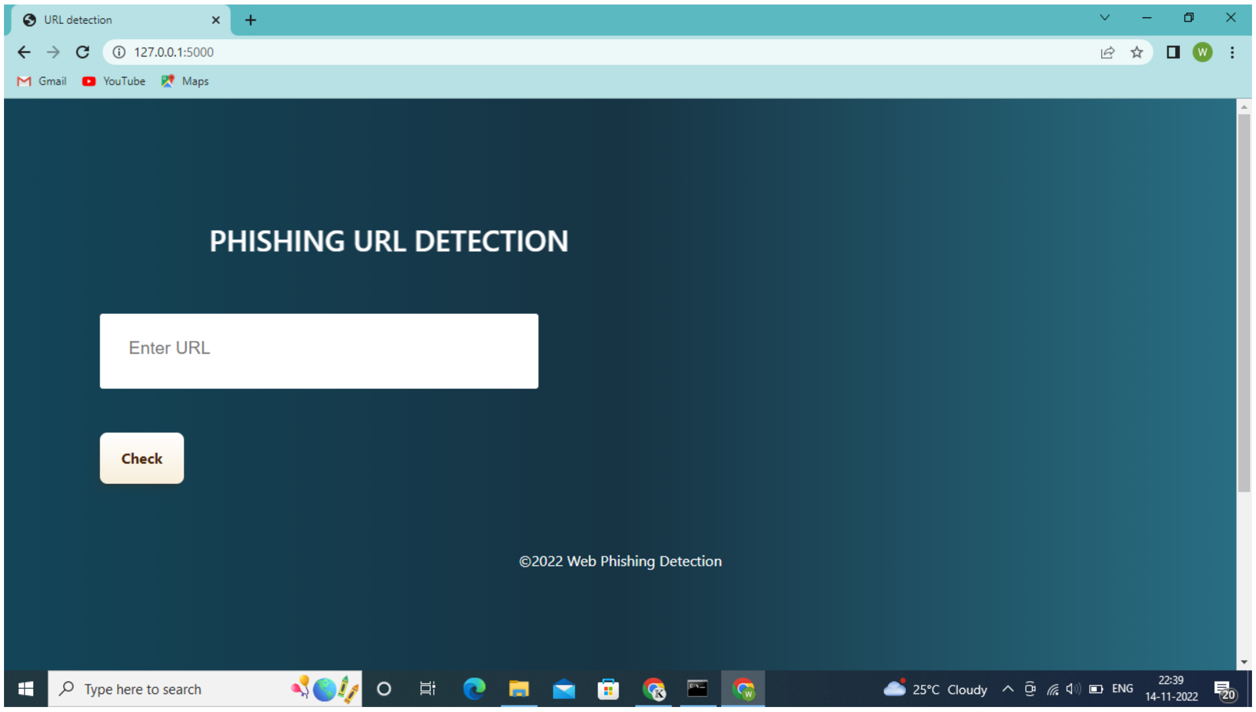
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned EndDate)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

c. **Reports from JIRA**

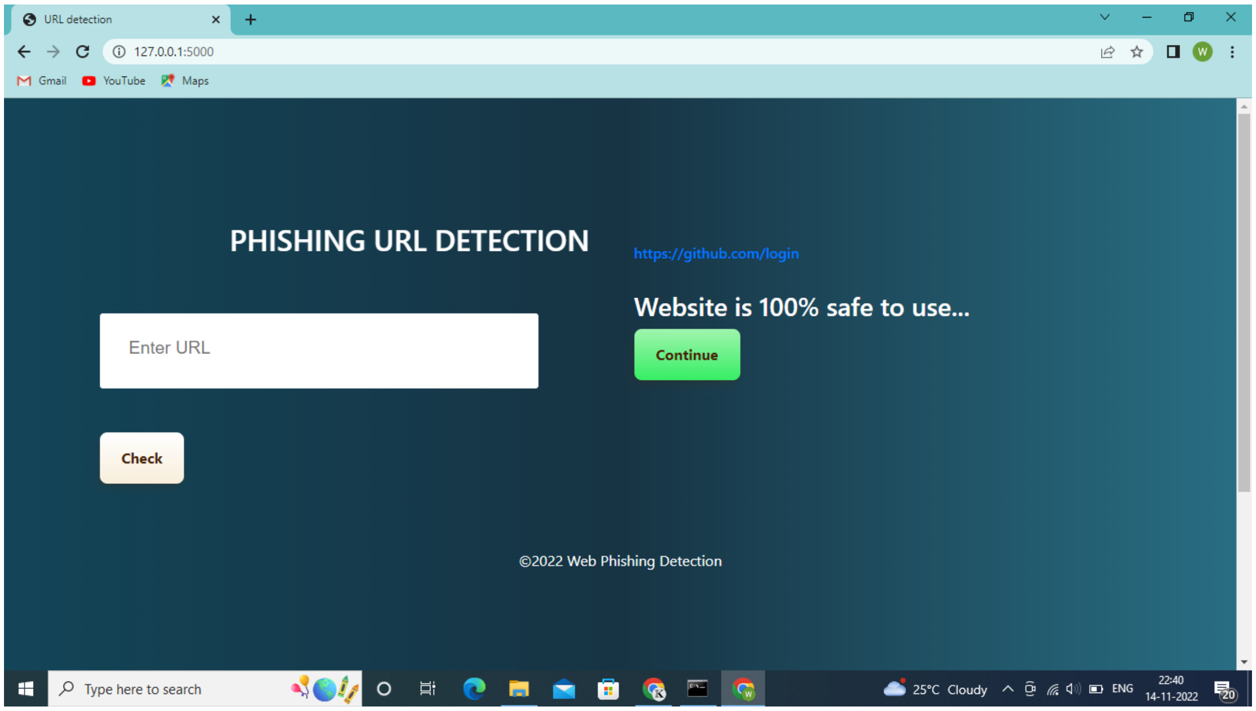


7. CODING & SOLUTIONING (Explain the features added in the project along with code)

a. Feature 1



b. Feature 2



PHISHING URL DETECTION

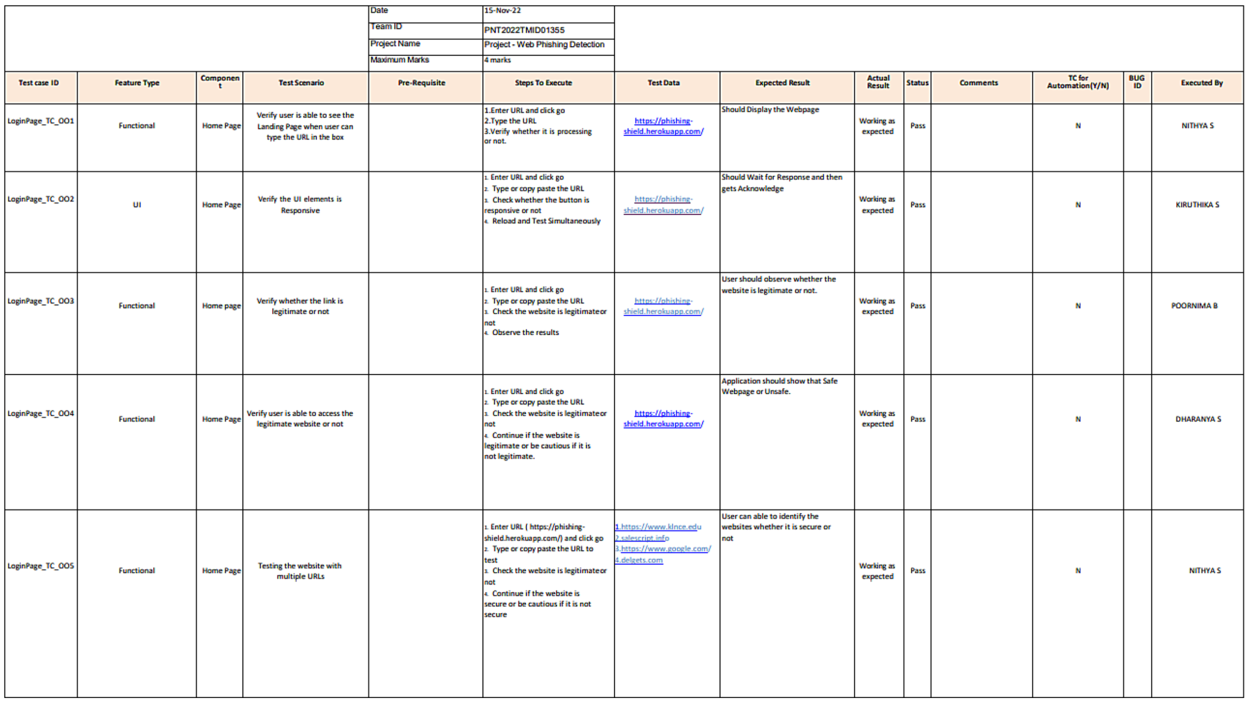
https://github.com/login

Website is 100% safe to use...

©2022 Web Phishing Detection

8. TESTING

a. Test Cases



Date	15-Nov-22
Team ID	PNT2022TMID01355
Project Name	Project - Web Phishing Detection
Maximum Marks	4 marks

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Landing Page when user can type the URL in the box		1.Enter URL and click go 2.Type the URL 3.Verify whether it is processing or not.	https://phishing-shield.herokuapp.com/	Should Display the Webpage	Working as expected	Pass		N		NITHYA S
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements is Responsive		1. Enter URL and click go 2. Type or copy paste the URL 3. Check whether the button is responsive or not 4. Reload and Test Simultaneously	https://phishing-shield.herokuapp.com/	Should Wait for Response and then gets Acknowledge	Working as expected	Pass		N		KIRUTHIKA S
LoginPage_TC_OO3	Functional	Home page	Verify whether the link is legitimate or not		1. Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimateor not 4. Observe the results	https://phishing-shield.herokuapp.com/	User should observe whether the website is legitimate or not.	Working as expected	Pass		N		POORNIMA B
LoginPage_TC_OO4	Functional	Home Page	Verify user is able to access the legitimate website or not		1. Enter URL and click go 2. Type or copy paste the URL 3. Check the website is legitimateor not 4. Continue if the website is legitimate or be cautious if it is not legitimate.	https://phishing-shield.herokuapp.com/	Application should show that Safe Webpage or Unsafe.	Working as expected	Pass		N		DHARANYA S
LoginPage_TC_OO5	Functional	Home Page	Testing the website with multiple URLs		1. Enter URL (https://phishing-shield.herokuapp.com/) and click go 2. Type or copy paste the URL to test 3. Check the website is legitimateor not 4. Continue if the website is secure or be cautious if it is not secure	1.https://www.klnce.edu 2.salescript.info 3.https://www.google.com/ 4.delgets.com	User can able to identify the websites whether it is secure or not	Working as expected	Pass		N		NITHYA S

b. User Acceptance Testing

Defect Analysis

This reportshows the numberof resolved or closedbugs at each severitylevel, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	10	2	4	20	36
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	0	2	1	3
Totals	23	9	12	25	70

Test Case Analysis

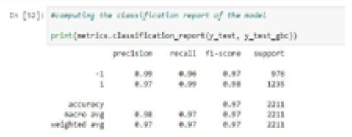
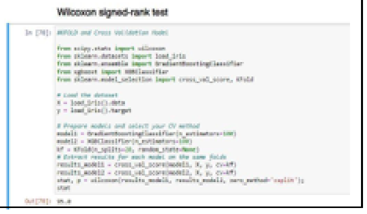
This report shows the number of test cases that have passed, failed,and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	50	0	0	50
Security	5	0	0	4
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	9
Final Report Output	10	0	0	10
Version Control	4	0	0	4

9. RESULTS

a. Performance Metrics

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Classification Model:** **Gradient Boosting Classification** Accuray Score- 97.4%	
2.	Tune the Model	Hyperparameter Tuning - 97% Validation Method – KFOLD & Cross Validation Method	

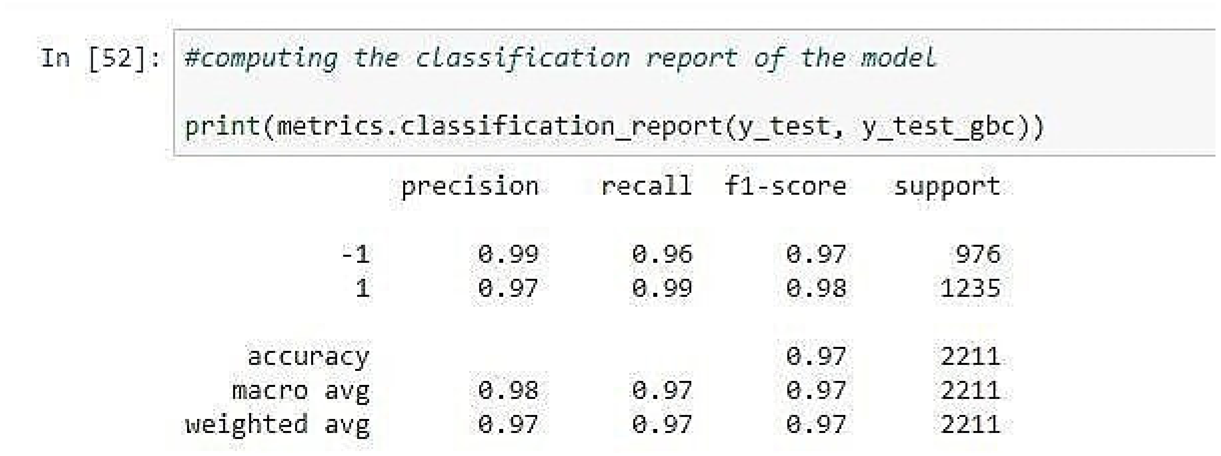
```
In [52]: #computing the classification report of the model

         print(metrics.classification_report(y_test, y_test_gbc))
```

Wilcoxon signed-rank test

```
In [78]: #KFOLD and Cross Validation Model

         from scipy.stats import wilcoxon
         from sklearn.datasets import load_iris
         from sklearn.ensemble import GradientBoostingClassifier
         from xgboost import XGBClassifier
         from sklearn.model_selection import cross_val_score, KFold

         # Load the dataset
         X = load_iris().data
         y = load_iris().target

         # Prepare models and select your CV method
         model1 = GradientBoostingClassifier(n_estimators=100)
         model2 = XGBClassifier(n_estimators=100)
         kf = KFold(n_splits=10, random_state=None)
         # Extract results for each model on the same folds
         results_model1 = cross_val_score(model1, X, y, cv=kf)
         results_model2 = cross_val_score(model2, X, y, cv=kf)
         stat, p = wilcoxon(results_model1, results_model2, zero_method='zsplit');
         stat

Out[78]: 95.0
```

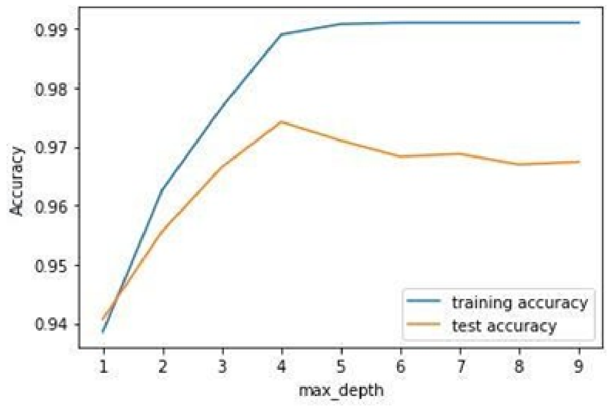
1.METRICS:

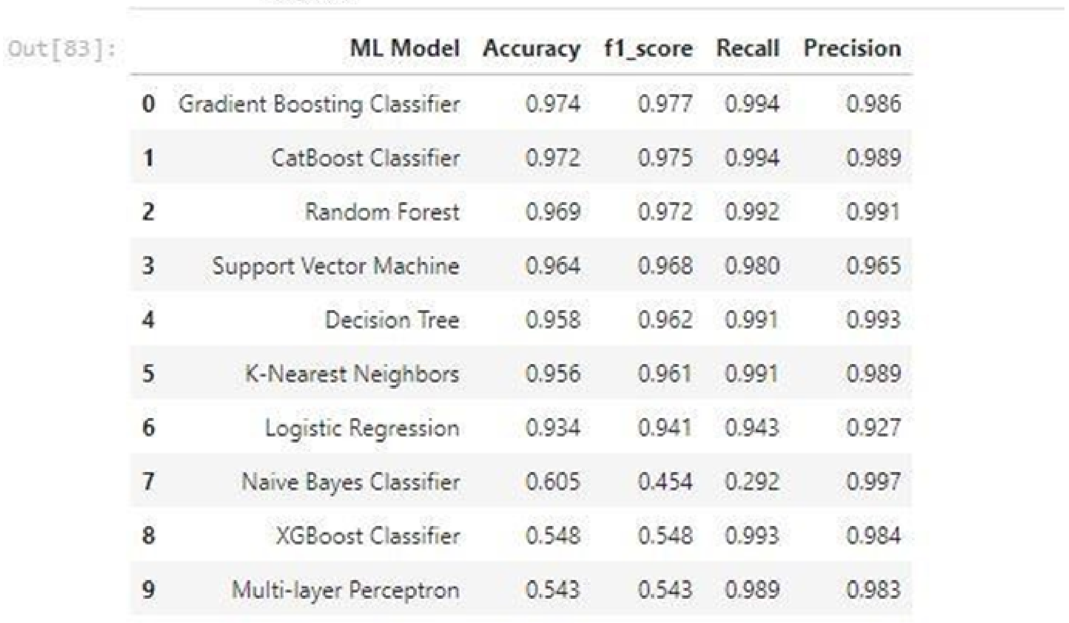
CLASSIFICATION REPORT:



```
In [52]: #computing the classification report of the model

         print(metrics.classification_report(y_test, y_test_gbc))

              precision    recall  f1-score   support

          -1       0.99      0.96      0.97       976
           1       0.97      0.99      0.98      1235

    accuracy                           0.97      2211
   macro avg       0.98      0.97      0.97      2211
weighted avg       0.97      0.97      0.97      2211
```

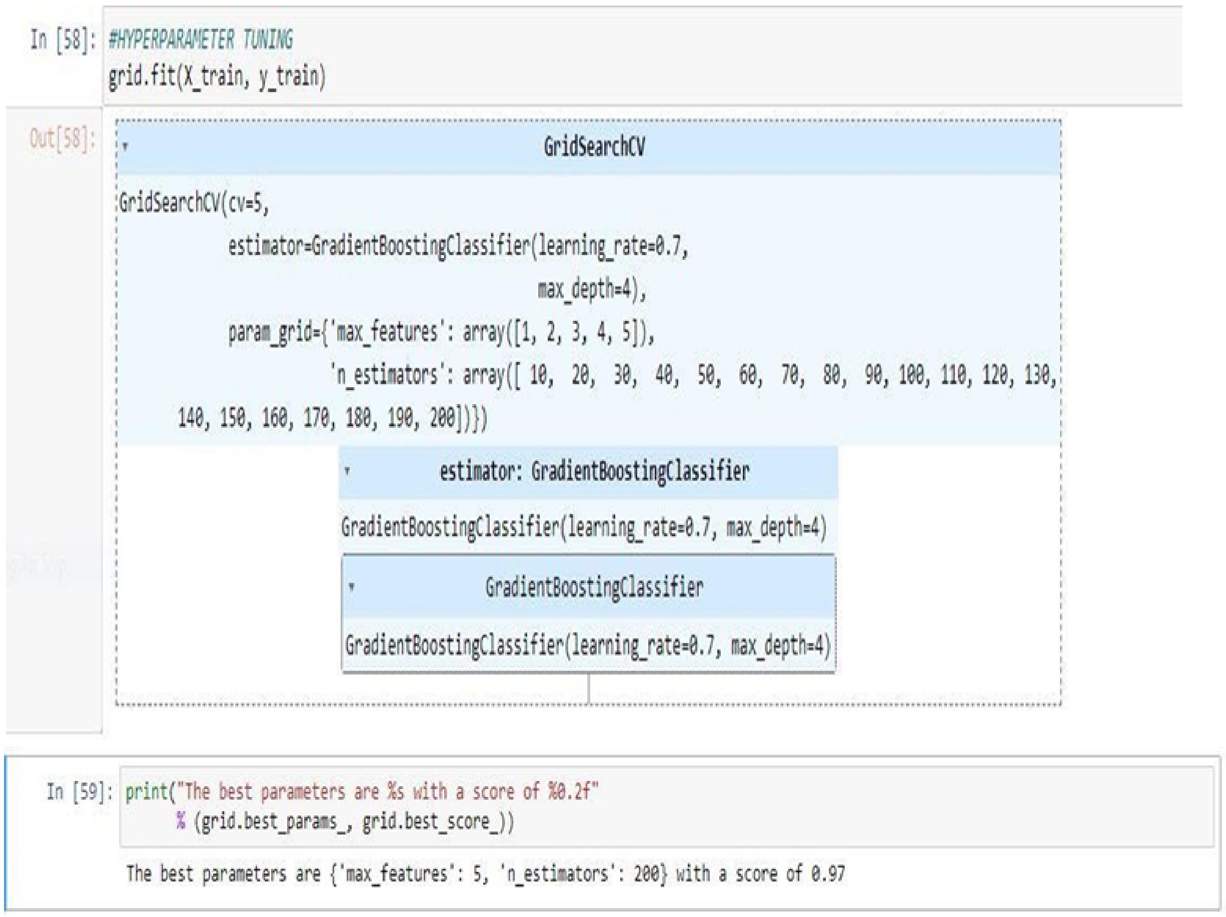
PERFORMANCE:




Out[83]:

	ML Model	Accuracy	f1_score	Recall	Precision
0	Gradient Boosting Classifier	0.974	0.977	0.994	0.986
1	CatBoost Classifier	0.972	0.975	0.994	0.989
2	Random Forest	0.969	0.972	0.992	0.991
3	Support Vector Machine	0.964	0.968	0.980	0.965
4	Decision Tree	0.958	0.962	0.991	0.993
5	K-Nearest Neighbors	0.956	0.961	0.991	0.989
6	Logistic Regression	0.934	0.941	0.943	0.927
7	Naive Bayes Classifier	0.605	0.454	0.292	0.997
8	XGBoost Classifier	0.548	0.548	0.993	0.984
9	Multi-layer Perceptron	0.543	0.543	0.989	0.983

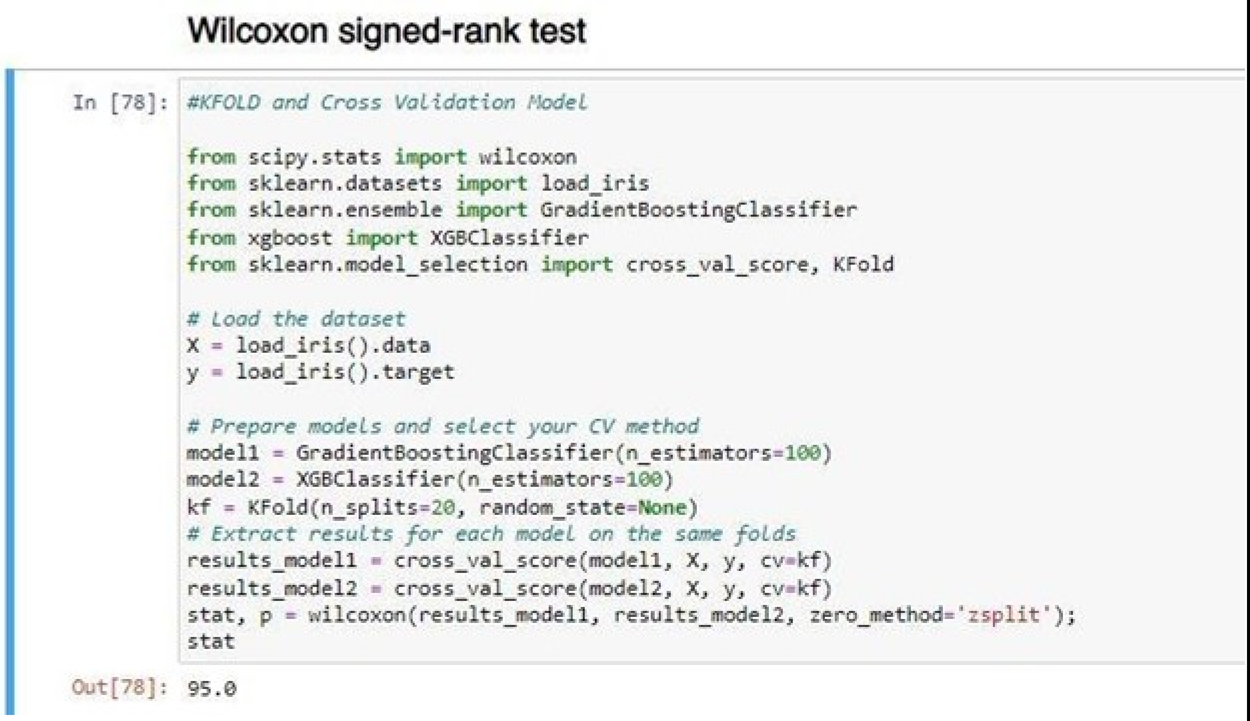
2.TUNE THE MODEL – HYPERPARAMETER TUNING



```
In [58]: #HYPERPARAMETER TUNING
         grid.fit(X_train, y_train)

Out[58]:
GridSearchCV

GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(learning_rate=0.7,
                                                  max_depth=4),
             param_grid={'max_features': array([1, 2, 3, 4, 5]),
                         'n_estimators': array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120, 130,
       140, 150, 160, 170, 180, 190, 200])})

estimator: GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)

GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

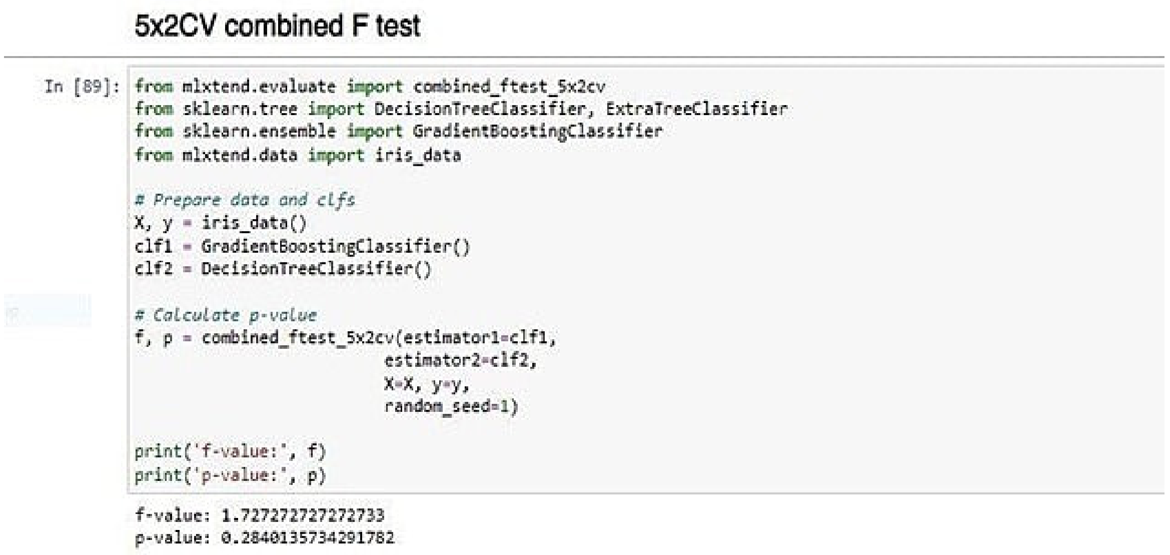
```
In [59]: print("The best parameters are %s with a score of %0.2f"
               % (grid.best_params_, grid.best_score_))

The best parameters are {'max_features': 5, 'n_estimators': 200} with a score of 0.97
```

VALIDATION METHODS: KFOLD & Cross Folding



Wilcoxon signed-rank test

```
In [78]: #KFOLD and Cross Validation Model

         from scipy.stats import wilcoxon
         from sklearn.datasets import load_iris
         from sklearn.ensemble import GradientBoostingClassifier
         from xgboost import XGBClassifier
         from sklearn.model_selection import cross_val_score, KFold

         # Load the dataset
         X = load_iris().data
         y = load_iris().target

         # Prepare models and select your CV method
         model1 = GradientBoostingClassifier(n_estimators=100)
         model2 = XGBClassifier(n_estimators=100)
         kf = KFold(n_splits=20, random_state=None)
         # Extract results for each model on the same folds
         results_model1 = cross_val_score(model1, X, y, cv=kf)
         results_model2 = cross_val_score(model2, X, y, cv=kf)
         stat, p = wilcoxon(results_model1, results_model2, zero_method='zsplit');
         stat

Out[78]: 95.0
```



5x2CV combined F test

```
In [89]: from mlxtend.evaluate import combined_ftest_5x2cv
         from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
         from sklearn.ensemble import GradientBoostingClassifier
         from mlxtend.data import iris_data

         # Prepare data and clfs
         X, y = iris_data()
         clf1 = GradientBoostingClassifier()
         clf2 = DecisionTreeClassifier()

         # Calculate p-value
         f, p = combined_ftest_5x2cv(estimator1=clf1,
                                     estimator2=clf2,
                                     X=X, y=y,
                                     random_seed=1)

         print('f-value:', f)
         print('p-value:', p)

         f-value: 1.727272727272733
         p-value: 0.2840135734291782
```

10. ADVANTAGES & DISADVANTAGES

Advantages

This model has the capability of retrieving and displaying accurate result for a website
.

Out of the many features considered, the most important one was HTTPS with SSL i.e. whether a website uses HTTPS or not.

The main advantage of our approach is the low false-positive rate when classifying this type of URL.

This model executes the framework with high efficiency , exactness and cost effectively and this task is actualized utilizing machine learning classification models.

Disadvantages

Finally, our approach's training time is relatively long due to the high dimensional vector generated by textual content features.

The main disadvantage of these systems is that the small change in the URL prevents matching in the list.

11. CONCLUSION

This Phishing URL Detection project aims to enhance detection method to detect phishing websites using machine learning technology.

In this model it displays whether the website is legal site or a phishing site. The machine learning models such as Logistic Regression , k-Nearest Neighbors , Support Vector Classifier , Naive Bayes , Decision Tree , Random Forest , Gradient Boosting , Catboost , Xgboost , Multilayer Perceptrons which perform Exploratory Data Analysis on phishing dataset and understanding their features.

Also the classifiers gives the better performance when we used more data as training data. Creating the notebook helped to learn a lot about the features affecting the models to detect whether URL is safe or not, also we came to know how to tuned model and how they affect the model performance.

Gradient Boosting Classifier currently classify URL upto 97.4% respective classes and hence reduces the chance of malicious attachments

12. FUTURE SCOPE

The future direction of this project is to develop an unsupervised deep learning method to generate insight from a URL.

In the future, we would like to extend our project by creating an extension to block the detected phishing website whenever the user clicks on their link.

In addition, the study can be extended in order to generate an outcome for a larger network and protect the privacy of an individuals.

13. **APPENDIX**

Source Code

App.py

```
#importing required libraries

from flask import Flask, request, render_template
import numpy as np
import pandas as pd
from sklearn import metrics
import warnings
import pickle
warnings.filterwarnings('ignore')
from feature import FeatureExtraction

file = open("urlmodel.pkl", "rb")
gbc = pickle.load(file)
file.close()



app = Flask(__name__)

@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":

        url = request.form["url"]
        obj = FeatureExtraction(url)
        x = np.array(obj.getFeaturesList()).reshape(1,30)

        y_pred =gbc.predict(x)[0]
        #1 is safe
        #-1 is unsafe
        y_pro_phishing = gbc.predict_proba(x)[0,0]
        y_pro_non_phishing = gbc.predict_proba(x)[0,1]
        # if(y_pred ==1 ):
        pred = "It is {0:.2f} % safe to go ".format(y_pro_phishing*100)
        return render_template('index.html',xx =round(y_pro_non_phishing,2),url=url )
    return render_template("index.html", xx =-1)

if __name__ == "__main__":
    app.run(debug=True,port=2002)
```

feature.py

```
import ipaddress
import re
from urllib import response
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import date, datetime
import time
from dateutil.parser import parse as date_parse
from urllib.parse import urlparse

class FeatureExtraction:
    features = []
    def __init__(self,url):
        self.features = []
        self.url = url
        self.domain = ""
        self.whois_response = ""
        self.urlparse = ""
        self.response = ""
        self.soup = ""

        try:
            self.response = requests.get(url)
            self.soup = BeautifulSoup(response.text, 'html.parser')
        except:
            pass

        try:
            self.urlparse = urlparse(url)
            self.domain = self.urlparse.netloc
        except:
            pass
```

```
try:
    self.whois_response = whois.whois(self.domain)
except:
    pass




self.features.append(self.UsingIp())
self.features.append(self.longUrl())
self.features.append(self.shortUrl())
self.features.append(self.symbol())
self.features.append(self.redirecting())
self.features.append(self.prefixSuffix())
self.features.append(self.SubDomains())
self.features.append(self.Hppts())
self.features.append(self.DomainRegLen())
self.features.append(self.Favicon())


self.features.append(self.NonStdPort())
self.features.append(self.HTTPSDomainURL())
self.features.append(self.RequestURL())
self.features.append(self.AnchorURL())
self.features.append(self.LinksInScriptTags())
self.features.append(self.ServerFormHandler())
self.features.append(self.InfoEmail())
self.features.append(self.AbnormalURL())
self.features.append(self.WebsiteForwarding())
self.features.append(self.StatusBarCust())

self.features.append(self.DisableRightClick())
self.features.append(self.UsingPopupWindow())
self.features.append(self.IframeRedirection())
self.features.append(self.AgeofDomain())
self.features.append(self.DNSRecording())
self.features.append(self.WebsiteTraffic())
self.features.append(self.PageRank())
self.features.append(self.GoogleIndex())
self.features.append(self.LinksPointingToPage())
self.features.append(self.StatsReport())
```

```
 # 1.UsingIp
def UsingIp(self):
    try:
        ipaddress.ip_address(self.url)
        return -1
    except:
        return 1

# 2.longUrl
def longUrl(self):
    if len(self.url) < 54:
        return 1
    if len(self.url) >= 54 and len(self.url) <= 75:
        return 0
    return -1

# 3.shortUrl
def shortUrl(self):
                                                                                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\.i
m|link\.zip\.net', self.url)
    if match:
        return -1
    return 1

# 4.Symbol@
def symbol(self):
    if re.findall("@",self.url):
        return -1
    return 1
```

```
# 5.Redirecting//
def redirecting(self):
    if self.url.rfind('//')>6:
        return -1
    return 1

# 6.prefixSuffix
def prefixSuffix(self):
    try:
        match = re.findall('\-', self.domain)
        if match:
            return -1
        return 1
    except:
        return -1

# 7.SubDomains
def SubDomains(self):
    dot_count = len(re.findall("\.", self.url))
    if dot_count == 1:
        return 1
    elif dot_count == 2:
        return 0
    return -1

# 8.HTTPS
def Hppts(self):
    try:
        https = self.urlparse.scheme
        if 'https' in https:
            return 1
        return -1
    except:
        return 1

# 9.DomainRegLen
def DomainRegLen(self):
    try:
        expiration_date = self.whois_response.expiration_date
        creation_date = self.whois_response.creation_date
        try:
            if(len(expiration_date)):
                expiration_date = expiration_date[0]
            except:
                pass
            try:
                if(len(creation_date)):
                    creation_date = creation_date[0]
            except:
                pass

                    age = (expiration_date.year-creation_date.year)*12+ (expiration_date.month-
creation_date.month)
            if age >=12:
                return 1
            return -1
        except:
            return -1

    # 10. Favicon
    def Favicon(self):
        try:
            for head in self.soup.find_all('head'):
                for head.link in self.soup.find_all('link', href=True):
                    dots = [x.start(0) for x in re.finditer('\.', head.link['href'])]
                    if self.url in head.link['href'] or len(dots) == 1 or domain in head.link['href']:
                        return 1
            return -1
        except:
            return -1

    # 11. NonStdPort
    def NonStdPort(self):
        try:
            port = self.domain.split(":")
            if len(port)>1:
                return -1
            return 1
        except:
            return -1

    # 12. HTTPSDomainURL
    def HTTPSDomainURL(self):
        try:
            if 'https' in self.domain:
                return -1
            return 1
        except:
            return -1

    # 13. RequestURL
    def RequestURL(self):
        try:
            for img in self.soup.find_all('img', src=True):
                dots = [x.start(0) for x in re.finditer('\.', img['src'])]
                if self.url in img['src'] or self.domain in img['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for audio in self.soup.find_all('audio', src=True):
                dots = [x.start(0) for x in re.finditer('\.', audio['src'])]
                if self.url in audio['src'] or self.domain in audio['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for embed in self.soup.find_all('embed', src=True):
                dots = [x.start(0) for x in re.finditer('\.', embed['src'])]
                if self.url in embed['src'] or self.domain in embed['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for iframe in self.soup.find_all('iframe', src=True):
                dots = [x.start(0) for x in re.finditer('\.', iframe['src'])]
                if self.url in iframe['src'] or self.domain in iframe['src'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            try:
                percentage = success/float(i) * 100
                if percentage < 22.0:
                    return 1
                elif((percentage >= 22.0) and (percentage < 61.0)):
                    return 0
                else:
                    return -1
            except:
                return 0
        except:
            return -1

    # 14. AnchorURL
    def AnchorURL(self):
        try:
            i,unsafe = 0,0
            for a in self.soup.find_all('a', href=True):
                if "#" in a['href'] or "javascript" in a['href'].lower() or "mailto" in a['href'].lower() or
not (url in a['href'] or self.domain in a['href']):
                    unsafe = unsafe + 1
                i = i + 1

            try:
                percentage = unsafe / float(i) * 100
                if percentage < 31.0:
                    return 1
                elif ((percentage >= 31.0) and (percentage < 67.0)):
                    return 0
                else:
                    return -1
            except:
                return -1

        except:
            return -1

    # 15. LinksInScriptTags
    def LinksInScriptTags(self):
        try:
            i,success = 0,0

            for link in self.soup.find_all('link', href=True):
                dots = [x.start(0) for x in re.finditer('\.', link['href'])]
                if self.url in link['href'] or self.domain in link['href'] or len(dots) == 1:
                    success = success + 1
                i = i+1

            for script in self.soup.find_all('script', src=True):
                dots = [x.start(0) for x in re.finditer('\.', script['src'])]
                if self.url in script['src'] or self.domain in script['src'] or len(dots) == 1:
```

```
                success = success + 1
            i = i+1

        try:
            percentage = success / float(i) * 100
            if percentage < 17.0:
                return 1
            elif((percentage >= 17.0) and (percentage < 81.0)):
                return 0
            else:
                return -1
        except:
            return 0
    except:
        return -1

# 16. ServerFormHandler
def ServerFormHandler(self):
    try:
        if len(self.soup.find_all('form', action=True))==0:
            return 1
        else :
            for form in self.soup.find_all('form', action=True):
                if form['action'] == "" or form['action'] == "about:blank":
                    return -1
                elif self.url not in form['action'] and self.domain not in form['action']:
                    return 0
                else:
                    return 1
    except:
        return -1

# 17. InfoEmail
def InfoEmail(self):
    try:
        if re.findall(r"[mail\(\)|mailto:?]", self.soap):
            return -1
        else:
            return 1
    except:
        return -1
```

```
# 18. AbnormalURL
def AbnormalURL(self):
    try:
        if self.response.text == self.whois_response:
            return 1
        else:
            return -1
    except:
        return -1

# 19. WebsiteForwarding
def WebsiteForwarding(self):
    try:
        if len(self.response.history) <= 1:
            return 1
        elif len(self.response.history) <= 4:
            return 0
        else:
            return -1
    except:
         return -1

# 20. StatusBarCust
def StatusBarCust(self):
    try:
        if re.findall("<script>.+onmouseover.+</script>", self.response.text):
            return 1
        else:
            return -1
    except:
         return -1

# 21. DisableRightClick
def DisableRightClick(self):
    try:
        if re.findall(r"event.button ?== ?2", self.response.text):
            return 1
        else:
            return -1
    except:
         return -1
```

```
# 22. UsingPopupWindow
def UsingPopupWindow(self):
    try:
        if re.findall(r"alert\(", self.response.text):
            return 1
        else:
            return -1
    except:
         return -1

# 23. IframeRedirection
def IframeRedirection(self):
    try:
        if re.findall(r"[<iframe>|<frameBorder>]", self.response.text):
            return 1
        else:
            return -1
    except:
         return -1

# 24. AgeofDomain
def AgeofDomain(self):
    try:
        creation_date = self.whois_response.creation_date
        try:
            if(len(creation_date)):
                creation_date = creation_date[0]
        except:
            pass

        today  = date.today()
        age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
        if age >=6:
            return 1
        return -1
    except:
        return -1

# 25. DNSRecording
def DNSRecording(self):
    try:
        creation_date = self.whois_response.creation_date
```

```
                try:
                    if(len(creation_date)):
                        creation_date = creation_date[0]
                except:
                    pass

                today  = date.today()
                age = (today.year-creation_date.year)*12+(today.month-creation_date.month)
                if age >=6:
                    return 1
                return -1
            except:
                return -1

        # 26. WebsiteTraffic
        def WebsiteTraffic(self):
            try:
                                                                                rank          =
    BeautifulSoup(urllib.request.urlopen("http://data.alexa.com/data?cli=10&dat=s&url="            +
    url).read(), "xml").find("REACH")['RANK']
                if (int(rank) < 100000):
                    return 1
                return 0
            except :
                return -1

        # 27. PageRank
        def PageRank(self):
            try:
                 prank_checker_response = requests.post("https://www.checkpagerank.net/index.php",
    {"name": self.domain})

                global_rank = int(re.findall(r"Global Rank: ([0-9]+)", rank_checker_response.text)[0])
                if global_rank > 0 and global_rank < 100000:
                    return 1
                return -1
            except:
                return -1


        # 28. GoogleIndex
        def GoogleIndex(self):
```

```
        try:
            site = search(self.url, 5)
            if site:
                return 1
            else:
                return -1
        except:
            return 1

    # 29. LinksPointingToPage
    def LinksPointingToPage(self):
        try:
            number_of_links = len(re.findall(r"<a href=", self.response.text))
            if number_of_links == 0:
                return 1
            elif number_of_links <= 2:
                return 0
            else:
                return -1
        except:
            return -1

    # 30. StatsReport
    def StatsReport(self):
        try:
            url_match = re.search(

'at\.ua|usa\.cc|baltazarpresentes\.com\.br|pe\.hu|esy\.es|hol\.es|sweddy\.com|myjino\.ru|96\.lt|ow\.l
y', url)
            ip_address = socket.gethostbyname(self.domain)
                                                                        ip_match        =
re.search('146\.112\.61\.108|213\.174\.157\.151|121\.50\.168\.88|192\.185\.217\.116|78\.46\.211\.1
58|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\.1
51\.148\.109|119\.28\.52\.61|54\.83\.43\.69|52\.69\.166\.231|216\.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\.2
24\.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\.1
9|37\.157\.192\.102|204\.11\.56\.48|110\.34\.231\.42', ip_address)
              if url_match:
                 return -1
              elif ip_match:
                 return -1
              return 1
           except:
              return 1

        def getFeaturesList(self):
           return self.features
```

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <meta name="description" content="This website is develop for identify the safety of url.">
   <meta name="keywords" content="phishing url,phishing,cyber security,machine
learning,classifier,python">
   <meta name="author" content="IBM">

   <!-- BootStrap -->
   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
      integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">

   <link href="static/styles.css" rel="stylesheet">
   <title>URL detection</title>

</head>

<body>

<div class=" container">
   <div class="row">
      <div class="form col-md" id="form1">
        <marquee direction="left"> <h2>PHISHING URL DETECTION</h2></marquee>

         <br><br><br>
         <form action="/" method ="post">
            <input type="text" class="form__input" name ='url' id="url" placeholder="Enter URL" required=""
/>
            <label for="url" class="form__label">URL</label>
            <button class="button" role="button" >Check</button>
         </form>
```

```
    </div>

    <div class="col-md" id="form2">

        <br>
        <h6 class = "right "><a href= {{ url }} target="_blank">{{ url }}</a></h6>

        <br>
        <h3 id="prediction"></h3>
        <button class="button2" id="button2" role="button" onclick="window.open('{{url}}')" target="_blank"
>Still want to Continue</button>
        <button class="button1" id="button1" role="button"  onclick="window.open('{{url}}')"
target="_blank">Continue</button>
    </div>
</div>
<br><br><br>
<center><p>©2022 Web Phishing Detection</p></center>
</div>

    <!-- JavaScript -->
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"
        integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
        crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.0/dist/umd/popper.min.js"
        integrity="sha384-Q6E9RHvbIyZFJoft+2mJbHaEWldlvI9IOYy5n3zV9zzTtmI3UksdQRVvoxMfooAo"
        crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/js/bootstrap.min.js"
        integrity="sha384-OgVRvuATP1z7JjHLkuOU7Xw704+h835Lr+6QL9UvYjZE3Ipu6Tp75j7Bh/kR0JKI"
        crossorigin="anonymous"></script>


    <script>

            let x = '{{xx}}';
            let num = x*100;
            if (0<=x && x<0.50){
                num = 100-num;
            }
            let txtx = num.toString();
            if(x<=1 && x>=0.50){
               var label = "Website is "+txtx +"% safe to use...";
               document.getElementById("prediction").innerHTML = label;
               document.getElementById("button1").style.display="block";
            }
            else if (0<=x && x<0.50){
               var label = "Website is "+txtx +"% unsafe to use..."
               document.getElementById("prediction").innerHTML = label ;
               document.getElementById("button2").style.display="block";
            }

      </script>

   </body>
   -
   </html>
```

<u>Phishing URL detection</u>

```
# Phishing URL Detection
```

The Internet has become an indispensable part of

our life, However, It also has provided opportunities to anonymously perform malicious activities like Phishing. Phishers try to deceive their victims by social engineering or creating mockup websites to steal information such as account ID, username, password from individuals and organizations. Although many methods have been proposed to detect phishing websites, Phishers have evolved their methods to escape from these detection methods. One of the most successful methods for detecting these malicious activities is Machine Learning. This is because most Phishing attacks have some common characteristics which can be identified by machine learning methods.

The steps demonstrated in this notebook are:

1. Loading the data
2. Familiarizing with data & EDA
3. Visualizing the data
4. Splitting the data
5. Training the data
6. Comparision of Model
7. Conclusion

```
#importing required libraries


import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

from sklearn import metrics

import warnings

warnings.filterwarnings('ignore')
```

1. Loading Data:

The dataset is borrowed from Kaggle, https://www.kaggle.com/eswarchandt/phishing-website-detector .

A collection of website URLs for 11000+ websites. Each sample has 30 website parameters and a class label identifying it as a phishing website or not (1 or -1).

The overview of this dataset is, it has 11054 samples with 32 features. Download the dataset from the link provided.

```
#Loading data into dataframe

data = pd.read_csv("phishing.csv")

data.head()
```

2. Familiarizing with Data & EDA:

In this step, few dataframe methods are used to look into the data and its features.

```
#Shape of dataframe

data.shape

#Listing the features of the dataset

data.columns

#Information about the dataset

data.info()

# nunique value in columns
```

data.nunique()

#droping index column

data = data.drop(['Index'],axis = 1)

#description of dataset

data.describe().T

data_set.append(9 OBSERVATIONS:

1. There are 11054 instances and 31 fearures in dataset.

2. Out of which 30 are independent features where as 1 is dependent feature.

3. Each feature is in int datatype, so there is no need to use LabelEncoder.

4. There is no outlier present in dataset.

5. There is no missing value in dataset.

<u>3. Visualizing the data:</u>

Few plots and graphs are displayed to find how the data is distributed and the how features are related to each other.

#Correlation heatmap

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

sns.heatmap(data.corr(), annot=True)

plt.show()

#pairplot for particular features

df = data[['PrefixSuffix-', 'SubDomains', 'HTTPS','AnchorURL','WebsiteTraffic','class']]

sns.pairplot(data = df,hue="class",corner=True);

```
# Phishing Count in pie chart


data['class'].value_counts().plot(kind='pie',autopct='%1.2f%%')

plt.title("Phishing Count")

plt.show()
```

4. Splitting the Data:

The data is split into train & test sets, 80-20 split.

```
# Splitting the dataset into dependant and independant fetature


X = data.drop(["class"],axis =1)

y = data["class"]

# Splitting the dataset into train and test sets: 80-20 split


from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)

X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

5. Model Building & Training:

Supervised machine learning is one of the most commonly used and successful types of machine learning. Supervised learning is used whenever we want to predict a certain outcome/label from a given set of features, and we have examples of features-label pairs. We build a machine learning model from these features-label pairs, which comprise our training set. Our goal is to make accurate predictions for new, never-before-seen data.

There are two major types of supervised machine learning problems, called classification and regression. Our data set comes under regression problem, as the prediction of suicide rate is a continuous number, or a floating-point number in programming terms. The supervised machine learning models (regression) considered to train the dataset in this notebook are:

1. Logistic Regression
2. k-Nearest Neighbors
3. Support Vector Clasifier
4. Naive Bayes
5. Decision Tree
6. Random Forest
7. Gradient Boosting
8. Catboost
9. Xgboost
10. Multilayer Perceptrons

The metrics considered to evaluate the model performance are Accuracy & F1 score.

```
# Creating holders to store the model performance results

ML_Model = []

accuracy = []

f1_score = []

recall = []

precision = []


#function to call for storing the results

def storeResults(model, a,b,c,d):

  ML_Model.append(model)
  accuracy.append(round(a, 3))

  f1_score.append(round(b, 3))

  recall.append(round(c, 3))

  precision.append(round(d, 3))
```

## **5.1. Logistic Regression**

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

```
# Linear regression model

from sklearn.linear_model import LogisticRegression

#from sklearn.pipeline import Pipeline



# instantiate the model

log = LogisticRegression()



# fit the model

log.fit(X_train,y_train)

#predicting the target value from the model for the samples



y_train_log = log.predict(X_train)

y_test_log = log.predict(X_test)

#computing the accuracy, f1_score, Recall, precision of the model performance



acc_train_log = metrics.accuracy_score(y_train,y_train_log)

acc_test_log = metrics.accuracy_score(y_test,y_test_log)
```

```
print("Logistic Regression : Accuracy on training Data: {:.3f}".format(acc_train_log))

print("Logistic Regression : Accuracy on test Data: {:.3f}".format(acc_test_log))

print()



f1_score_train_log = metrics.f1_score(y_train,y_train_log)

f1_score_test_log = metrics.f1_score(y_test,y_test_log)

print("Logistic Regression : f1_score on training Data: {:.3f}".format(f1_score_train_log))

print("Logistic Regression : f1_score on test Data: {:.3f}".format(f1_score_test_log))

print()



recall_score_train_log = metrics.recall_score(y_train,y_train_log)

recall_score_test_log = metrics.recall_score(y_test,y_test_log)

print("Logistic Regression : Recall on training Data: {:.3f}".format(recall_score_train_log))

print("Logistic Regression : Recall on test Data: {:.3f}".format(recall_score_test_log))

print()



precision_score_train_log = metrics.precision_score(y_train,y_train_log)

precision_score_test_log = metrics.precision_score(y_test,y_test_log)

print("Logistic Regression : precision on training Data: {:.3f}".format(precision_score_train_log))

print("Logistic Regression : precision on test Data: {:.3f}".format(precision_score_test_log))

#computing the classification report of the model



print(metrics.classification_report(y_test, y_test_log))

#storing the results. The below mentioned order of parameter passing is important.
```

```
storeResults('Logistic Regression',acc_test_log,f1_score_test_log,
          recall_score_train_log,precision_score_train_log)
```

## 5.2. K-Nearest Neighbors : Classifier

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

```
# K-Nearest Neighbors Classifier model

from sklearn.neighbors import KNeighborsClassifier



# instantiate the model

knn = KNeighborsClassifier(n_neighbors=1)



# fit the model

knn.fit(X_train,y_train)

#predicting the target value from the model for the samples

y_train_knn = knn.predict(X_train)

y_test_knn = knn.predict(X_test)

#computing the accuracy,f1_score,Recall,precision of the model performance



acc_train_knn = metrics.accuracy_score(y_train,y_train_knn)

acc_test_knn = metrics.accuracy_score(y_test,y_test_knn)

print("K-Nearest Neighbors : Accuracy on training Data: {:.3f}".format(acc_train_knn))

print("K-Nearest Neighbors : Accuracy on test Data: {:.3f}".format(acc_test_knn))

print()
```

```
f1_score_train_knn = metrics.f1_score(y_train,y_train_knn)

f1_score_test_knn = metrics.f1_score(y_test,y_test_knn)

print("K-Nearest Neighbors : f1_score on training Data: {:.3f}".format(f1_score_train_knn))

print("K-Nearest Neighbors : f1_score on test Data: {:.3f}".format(f1_score_test_knn))

print()


recall_score_train_knn = metrics.recall_score(y_train,y_train_knn)

recall_score_test_knn = metrics.recall_score(y_test,y_test_knn)

print("K-Nearest Neighborsn : Recall on training Data: {:.3f}".format(recall_score_train_knn))

print("Logistic Regression : Recall on test Data: {:.3f}".format(recall_score_test_knn))

print()


precision_score_train_knn = metrics.precision_score(y_train,y_train_knn)

precision_score_test_knn = metrics.precision_score(y_test,y_test_knn)

print("K-Nearest Neighbors : precision on training Data: {:.3f}".format(precision_score_train_knn))

print("K-Nearest Neighbors : precision on test Data: {:.3f}".format(precision_score_test_knn))

#computing the classification report of the model


print(metrics.classification_report(y_test, y_test_knn))

training_accuracy = []

test_accuracy = []

# try max_depth from 1 to 20

depth = range(1,20)

for n in depth:

    knn = KNeighborsClassifier(n_neighbors=n)
    knn.fit(X_train, y_train)

    # record training set accuracy

    training_accuracy.append(knn.score(X_train, y_train))

    # record generalization accuracy

    test_accuracy.append(knn.score(X_test, y_test))




#plotting the training & testing accuracy for n_estimators from 1 to 20

plt.plot(depth, training_accuracy, label="training accuracy")

plt.plot(depth, test_accuracy, label="test accuracy")

plt.ylabel("Accuracy")

plt.xlabel("n_neighbors")

plt.legend();

#storing the results. The below mentioned order of parameter passing is important.


storeResults('K-Nearest Neighbors',acc_test_knn,f1_score_test_knn,

         recall_score_train_knn,precision_score_train_knn)
```

## 5.3. Support Vector Machine : Classifier

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.

```
# Support Vector Classifier model

from sklearn.svm import SVC

from sklearn.model_selection import GridSearchCV
```

```
# defining parameter range

param_grid = {'gamma': [0.1],'kernel': ['rbf','linear']}


svc = GridSearchCV(SVC(), param_grid)


# fitting the model for grid search

svc.fit(X_train, y_train)


#predicting the target value from the model for the samples

y_train_svc = svc.predict(X_train)

y_test_svc = svc.predict(X_test)

#computing the accuracy, f1_score, Recall, precision of the model performance


acc_train_svc = metrics.accuracy_score(y_train,y_train_svc)

acc_test_svc = metrics.accuracy_score(y_test,y_test_svc)

print("Support Vector Machine : Accuracy on training Data: {:.3f}".format(acc_train_svc))

print("Support Vector Machine : Accuracy on test Data: {:.3f}".format(acc_test_svc))

print()


f1_score_train_svc = metrics.f1_score(y_train,y_train_svc)

f1_score_test_svc = metrics.f1_score(y_test,y_test_svc)

print("Support Vector Machine : f1_score on training Data: {:.3f}".format(f1_score_train_svc))

print("Support Vector Machine : f1_score on test Data: {:.3f}".format(f1_score_test_svc))

print()
```

```
recall_score_train_svc = metrics.recall_score(y_train,y_train_svc)

recall_score_test_svc = metrics.recall_score(y_test,y_test_svc)

print("Support Vector Machine : Recall on training Data: {:.3f}".format(recall_score_train_svc))

print("Support Vector Machine : Recall on test Data: {:.3f}".format(recall_score_test_svc))

print()



precision_score_train_svc = metrics.precision_score(y_train,y_train_svc)

precision_score_test_svc = metrics.precision_score(y_test,y_test_svc)

print("Support Vector Machine : precision on training Data: {:.3f}".format(precision_score_train_svc))

print("Support Vector Machine : precision on test Data: {:.3f}".format(precision_score_test_svc))

#computing the classification report of the model



print(metrics.classification_report(y_test, y_test_svc))

#storing the results. The below mentioned order of parameter passing is important.



storeResults('Support Vector Machine',acc_test_svc,f1_score_test_svc,

        recall_score_train_svc,precision_score_train_svc)
```

## 5.4. Naive Bayes : Classifier

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.It is mainly used in text, image classification that includes a high-dimensional training dataset. Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

```
# Naive Bayes Classifier Model

from sklearn.naive_bayes import GaussianNB

from sklearn.pipeline import Pipeline
```

```
# instantiate the model

nb=  GaussianNB()


# fit the model

nb.fit(X_train,y_train)

#predicting the target value from the model for the samples

y_train_nb = nb.predict(X_train)

y_test_nb = nb.predict(X_test)

#computing the accuracy, f1_score, Recall, precision of the model performance


acc_train_nb = metrics.accuracy_score(y_train,y_train_nb)

acc_test_nb = metrics.accuracy_score(y_test,y_test_nb)

print("Naive Bayes Classifier : Accuracy on training Data: {:.3f}".format(acc_train_nb))

print("Naive Bayes Classifier : Accuracy on test Data: {:.3f}".format(acc_test_nb))

print()


f1_score_train_nb = metrics.f1_score(y_train,y_train_nb)

f1_score_test_nb = metrics.f1_score(y_test,y_test_nb)

print("Naive Bayes Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_nb))

print("Naive Bayes Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_nb))

print()


recall_score_train_nb = metrics.recall_score(y_train,y_train_nb)

recall_score_test_nb = metrics.recall_score(y_test,y_test_nb)
```

```
print("Naive Bayes Classifier : Recall on training Data: {:.3f}".format(recall_score_train_nb))

print("Naive Bayes Classifier : Recall on test Data: {:.3f}".format(recall_score_test_nb))

print()



precision_score_train_nb = metrics.precision_score(y_train,y_train_nb)

precision_score_test_nb = metrics.precision_score(y_test,y_test_nb)

print("Naive Bayes Classifier : precision on training Data: {:.3f}".format(precision_score_train_nb))

print("Naive Bayes Classifier : precision on test Data: {:.3f}".format(precision_score_test_nb))

#computing the classification report of the model



print(metrics.classification_report(y_test, y_test_svc))

#storing the results. The below mentioned order of parameter passing is important.



storeResults('Naive Bayes Classifier',acc_test_nb,f1_score_test_nb,

          recall_score_train_nb,precision_score_train_nb)
```

## 5.5. Decision Trees : Classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

```
# Decision Tree Classifier model

from sklearn.tree import DecisionTreeClassifier



# instantiate the model

tree = DecisionTreeClassifier(max_depth=30)
```

```
# fit the model

tree.fit(X_train, y_train)

#predicting the target value from the model for the samples


y_train_tree = tree.predict(X_train)

y_test_tree = tree.predict(X_test)

#computing the accuracy, f1_score, Recall, precision of the model performance


acc_train_tree = metrics.accuracy_score(y_train,y_train_tree)

acc_test_tree = metrics.accuracy_score(y_test,y_test_tree)

print("Decision Tree : Accuracy on training Data: {:.3f}".format(acc_train_tree))

print("Decision Tree : Accuracy on test Data: {:.3f}".format(acc_test_tree))

print()


f1_score_train_tree = metrics.f1_score(y_train,y_train_tree)

f1_score_test_tree = metrics.f1_score(y_test,y_test_tree)

print("Decision Tree : f1_score on training Data: {:.3f}".format(f1_score_train_tree))

print("Decision Tree : f1_score on test Data: {:.3f}".format(f1_score_test_tree))

print()


recall_score_train_tree = metrics.recall_score(y_train,y_train_tree)

recall_score_test_tree = metrics.recall_score(y_test,y_test_tree)

print("Decision Tree : Recall on training Data: {:.3f}".format(recall_score_train_tree))

print("Decision Tree : Recall on test Data: {:.3f}".format(recall_score_test_tree))

print()
```

```
precision_score_train_tree = metrics.precision_score(y_train,y_train_tree)

precision_score_test_tree = metrics.precision_score(y_test,y_test_tree)

print("Decision Tree : precision on training Data: {:.3f}".format(precision_score_train_tree))

print("Decision Tree : precision on test Data: {:.3f}".format(precision_score_test_tree))

#computing the classification report of the model


print(metrics.classification_report(y_test, y_test_tree))

training_accuracy = []

test_accuracy = []

# try max_depth from 1 to 30

depth = range(1,30)

for n in depth:

    tree_test = DecisionTreeClassifier(max_depth=n)


    tree_test.fit(X_train, y_train)

    # record training set accuracy

    training_accuracy.append(tree_test.score(X_train, y_train))

    # record generalization accuracy

    test_accuracy.append(tree_test.score(X_test, y_test))



#plotting the training & testing accuracy for max_depth from 1 to 30

plt.plot(depth, training_accuracy, label="training accuracy")

plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")

plt.xlabel("max_depth")

plt.legend();

#storing the results. The below mentioned order of parameter passing is important.



storeResults('Decision Tree',acc_test_tree,f1_score_test_tree,

        recall_score_train_tree,precision_score_train_tree)
```

## 5.6. Random Forest : Classifier

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

```
# Random Forest Classifier Model

from sklearn.ensemble import RandomForestClassifier



# instantiate the model

forest = RandomForestClassifier(n_estimators=10)



# fit the model

forest.fit(X_train,y_train)

#predicting the target value from the model for the samples

y_train_forest = forest.predict(X_train)

y_test_forest = forest.predict(X_test)

#computing the accuracy, f1_score, Recall, precision of the model performance



acc_train_forest = metrics.accuracy_score(y_train,y_train_forest)
```

```
acc_test_forest = metrics.accuracy_score(y_test,y_test_forest)

print("Random Forest : Accuracy on training Data: {:.3f}".format(acc_train_forest))

print("Random Forest : Accuracy on test Data: {:.3f}".format(acc_test_forest))

print()



f1_score_train_forest = metrics.f1_score(y_train,y_train_forest)

f1_score_test_forest = metrics.f1_score(y_test,y_test_forest)

print("Random Forest : f1_score on training Data: {:.3f}".format(f1_score_train_forest))

print("Random Forest : f1_score on test Data: {:.3f}".format(f1_score_test_forest))

print()



recall_score_train_forest = metrics.recall_score(y_train,y_train_forest)

recall_score_test_forest = metrics.recall_score(y_test,y_test_forest)

print("Random Forest : Recall on training Data: {:.3f}".format(recall_score_train_forest))

print("Random Forest : Recall on test Data: {:.3f}".format(recall_score_test_forest))

print()



precision_score_train_forest = metrics.precision_score(y_train,y_train_forest)

precision_score_test_forest = metrics.precision_score(y_test,y_test_tree)

print("Random Forest : precision on training Data: {:.3f}".format(precision_score_train_forest))

print("Random Forest : precision on test Data: {:.3f}".format(precision_score_test_forest))

#computing the classification report of the model



print(metrics.classification_report(y_test, y_test_forest))

training_accuracy = []
```

```
test_accuracy = []
# try max_depth from 1 to 20
depth = range(1,20)
for n in depth:
    forest_test =  RandomForestClassifier(n_estimators=n)


    forest_test.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy
    test_accuracy.append(forest_test.score(X_test, y_test))



#plotting the training & testing accuracy for n_estimators from 1 to 20
plt.figure(figsize=None)
plt.plot(depth, training_accuracy, label="training accuracy")
plt.plot(depth, test_accuracy, label="test accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_estimators")
plt.legend();
#storing the results. The below mentioned order of parameter passing is important.

storeResults('Random Forest',acc_test_forest,f1_score_test_forest,
          recall_score_train_forest,precision_score_train_forest)
```

## 5.7.Gradient Boosting Classifier

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting. Boosting algorithms play a crucial role in dealing with bias variance trade-off. Unlike bagging algorithms, which only controls for high variance in a model, boosting controls both the aspects (bias & variance), and is considered to be more effective.

```
# Gradient Boosting Classifier Model

from sklearn.ensemble import GradientBoostingClassifier



# instantiate the model

gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)



# fit the model

gbc.fit(X_train,y_train)

#predicting the target value from the model for the samples

y_train_gbc = gbc.predict(X_train)

y_test_gbc = gbc.predict(X_test)

#computing the accuracy, f1_score, Recall, precision of the model performance



acc_train_gbc = metrics.accuracy_score(y_train,y_train_gbc)

acc_test_gbc = metrics.accuracy_score(y_test,y_test_gbc)

print("Gradient Boosting Classifier : Accuracy on training Data: {:.3f}".format(acc_train_gbc))

print("Gradient Boosting Classifier : Accuracy on test Data: {:.3f}".format(acc_test_gbc))

print()



f1_score_train_gbc = metrics.f1_score(y_train,y_train_gbc)

f1_score_test_gbc = metrics.f1_score(y_test,y_test_gbc)

print("Gradient Boosting Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_gbc))

print("Gradient Boosting Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_gbc))
print()


recall_score_train_gbc = metrics.recall_score(y_train,y_train_gbc)

recall_score_test_gbc =  metrics.recall_score(y_test,y_test_gbc)

print("Gradient Boosting Classifier : Recall on training Data: {:.3f}".format(recall_score_train_gbc))

print("Gradient Boosting Classifier : Recall on test Data: {:.3f}".format(recall_score_test_gbc))

print()


precision_score_train_gbc = metrics.precision_score(y_train,y_train_gbc)

precision_score_test_gbc = metrics.precision_score(y_test,y_test_gbc)

print("Gradient Boosting Classifier : precision on training Data: {:.3f}".format(precision_score_train_gbc))

print("Gradient Boosting Classifier : precision on test Data: {:.3f}".format(precision_score_test_gbc))

#computing the classification report of the model


print(metrics.classification_report(y_test, y_test_gbc))

training_accuracy = []

test_accuracy = []

# try learning_rate from 0.1 to 0.9

depth = range(1,10)

for n in depth:

    forest_test =  GradientBoostingClassifier(learning_rate = n*0.1)


    forest_test.fit(X_train, y_train)

    # record training set accuracy

    training_accuracy.append(forest_test.score(X_train, y_train))
    # record generalization accuracy

    test_accuracy.append(forest_test.score(X_test, y_test))




#plotting the training & testing accuracy for n_estimators from 1 to 50

plt.figure(figsize=None)

plt.plot(depth, training_accuracy, label="training accuracy")

plt.plot(depth, test_accuracy, label="test accuracy")

plt.ylabel("Accuracy")

plt.xlabel("learning_rate")

plt.legend();

training_accuracy = []

test_accuracy = []

# try learning_rate from 0.1 to 0.9

depth = range(1,10,1)

for n in depth:

    forest_test =  GradientBoostingClassifier(max_depth=n,learning_rate = 0.7)


    forest_test.fit(X_train, y_train)

    # record training set accuracy

    training_accuracy.append(forest_test.score(X_train, y_train))

    # record generalization accuracy

    test_accuracy.append(forest_test.score(X_test, y_test))
```

```
#plotting the training & testing accuracy for n_estimators from 1 to 50

plt.figure(figsize=None)

plt.plot(depth, training_accuracy, label="training accuracy")

plt.plot(depth, test_accuracy, label="test accuracy")

plt.ylabel("Accuracy")

plt.xlabel("max_depth")

plt.legend();

#storing the results. The below mentioned order of parameter passing is important.


storeResults('Gradient Boosting Classifier',acc_test_gbc,f1_score_test_gbc,

        recall_score_train_gbc,precision_score_train_gbc)
```

## 5.8. CatBoost Classifier

CatBoost is a recently open-sourced machine learning algorithm from Yandex. It can easily integrate with deep learning frameworks like Google's TensorFlow and Apple's Core ML. It can work with diverse data types to help solve a wide range of problems that businesses face today.

```
#  catboost Classifier Model

from catboost import CatBoostClassifier


# instantiate the model

cat = CatBoostClassifier(learning_rate  = 0.1)


# fit the model

cat.fit(X_train,y_train)

#predicting the target value from the model for the samples

y_train_cat = cat.predict(X_train)
```

```
y_test_cat = cat.predict(X_test)


#computing the accuracy, f1_score, Recall, precision of the model performance



acc_train_cat  = metrics.accuracy_score(y_train,y_train_cat)

acc_test_cat = metrics.accuracy_score(y_test,y_test_cat)

print("CatBoost Classifier : Accuracy on training Data: {:.3f}".format(acc_train_cat))

print("CatBoost Classifier : Accuracy on test Data: {:.3f}".format(acc_test_cat))

print()


f1_score_train_cat = metrics.f1_score(y_train,y_train_cat)

f1_score_test_cat = metrics.f1_score(y_test,y_test_cat)

print("CatBoost Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_cat))

print("CatBoost Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_cat))

print()


recall_score_train_cat = metrics.recall_score(y_train,y_train_cat)

recall_score_test_cat = metrics.recall_score(y_test,y_test_cat)

print("CatBoost Classifier : Recall on training Data: {:.3f}".format(recall_score_train_cat))

print("CatBoost Classifier : Recall on test Data: {:.3f}".format(recall_score_test_cat))

print()


precision_score_train_cat = metrics.precision_score(y_train,y_train_cat)

precision_score_test_cat = metrics.precision_score(y_test,y_test_cat)

print("CatBoost Classifier : precision on training Data: {:.3f}".format(precision_score_train_cat))
print("CatBoost Classifier : precision on test Data: {:.3f}".format(precision_score_test_cat))

#computing the classification report of the model


print(metrics.classification_report(y_test, y_test_cat))

training_accuracy = []

test_accuracy = []

# try learning_rate from 0.1 to 0.9

depth = range(1,10)

for n in depth:

    forest_test =  CatBoostClassifier(learning_rate = n*0.1)


    forest_test.fit(X_train, y_train)

    # record training set accuracy

    training_accuracy.append(forest_test.score(X_train, y_train))

    # record generalization accuracy

    test_accuracy.append(forest_test.score(X_test, y_test))



#plotting the training & testing accuracy for n_estimators from 1 to 50

plt.figure(figsize=None)

plt.plot(depth, training_accuracy, label="training accuracy")

plt.plot(depth, test_accuracy, label="test accuracy")

plt.ylabel("Accuracy")

plt.xlabel("learning_rate")

plt.legend();
```

```
#storing the results. The below mentioned order of parameter passing is important.


storeResults('CatBoost Classifier',acc_test_cat,f1_score_test_cat,

        recall_score_train_cat,precision_score_train_cat)
```

## 5.9. XGBoost Classifier

XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning. In this post you will discover how you can install and create your first XGBoost model in Python

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

y_train = le.fit_transform(y_train)

#  XGBoost Classifier Model

from xgboost import XGBClassifier


# instantiate the model

xgb = XGBClassifier()


# fit the model

xgb.fit(X_train,y_train)

#predicting the target value from the model for the samples

y_train_xgb = xgb.predict(X_train)

y_test_xgb = xgb.predict(X_test)

#computing the accuracy, f1_score, Recall, precision of the model performance
```

```
acc_train_xgb = metrics.accuracy_score(y_train,y_train_xgb)

acc_test_xgb = metrics.accuracy_score(y_test,y_test_xgb)

print("XGBoost Classifier : Accuracy on training Data: {:.3f}".format(acc_train_xgb))

print("XGBoost Classifier : Accuracy on test Data: {:.3f}".format(acc_test_xgb))

print()


f1_score_train_xgb = metrics.f1_score(y_train,y_train_xgb)

f1_score_test_xgb = metrics.f1_score(y_test,y_test_xgb,average="micro")

print("XGBoost Classifier : f1_score on training Data: {:.3f}".format(f1_score_train_xgb))

print("XGBoost Classifier : f1_score on test Data: {:.3f}".format(f1_score_test_xgb))

print()


recall_score_train_xgb = metrics.recall_score(y_train,y_train_xgb)

recall_score_test_xgb = metrics.recall_score(y_test,y_test_xgb,average="micro")

print("XGBoost Classifier : Recall on training Data: {:.3f}".format(recall_score_train_xgb))

print("XGBoost Classifier : Recall on test Data: {:.3f}".format(recall_score_train_xgb))

print()


precision_score_train_xgb = metrics.precision_score(y_train,y_train_xgb)

precision_score_test_xgb = metrics.precision_score(y_test,y_test_xgb,average="micro")

print("XGBoost Classifier : precision on training Data: {:.3f}".format(precision_score_train_xgb))

print("XGBoost Classifier : precision on test Data: {:.3f}".format(precision_score_train_xgb))

#storing the results. The below mentioned order of parameter passing is important.


storeResults('XGBoost Classifier',acc_test_xgb,f1_score_test_xgb,
        recall_score_train_xgb,precision_score_train_xgb)
```

## 5.10. Multi-layer Perceptron classifier

MLPClassifier stands for Multi-layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLPClassifier relies on an underlying Neural Network to perform the task of classification.

```
# Multi-layer Perceptron Classifier Model

from sklearn.neural_network import MLPClassifier



# instantiate the model

mlp = MLPClassifier()

#mlp = GridSearchCV(mlpc, parameter_space)



# fit the model

mlp.fit(X_train,y_train)

#predicting the target value from the model for the samples

y_train_mlp = mlp.predict(X_train)

y_test_mlp = mlp.predict(X_test)

#computing the accuracy, f1_score, Recall, precision of the model performance



acc_train_mlp  = metrics.accuracy_score(y_train,y_train_mlp)

acc_test_mlp = metrics.accuracy_score(y_test,y_test_mlp)

print("Multi-layer Perceptron : Accuracy on training Data: {:.3f}".format(acc_train_mlp))

print("Multi-layer Perceptron : Accuracy on test Data: {:.3f}".format(acc_test_mlp))

print()
```

```
f1_score_train_mlp = metrics.f1_score(y_train,y_train_mlp)

f1_score_test_mlp = metrics.f1_score(y_test,y_test_mlp,average="micro")

print("Multi-layer Perceptron : f1_score on training Data: {:.3f}".format(f1_score_train_mlp))

print("Multi-layer Perceptron : f1_score on test Data: {:.3f}".format(f1_score_train_mlp))

print()


recall_score_train_mlp = metrics.recall_score(y_train,y_train_mlp)

recall_score_test_mlp = metrics.recall_score(y_test,y_test_mlp,average="micro")

print("Multi-layer Perceptron : Recall on training Data: {:.3f}".format(recall_score_train_mlp))

print("Multi-layer Perceptron : Recall on test Data: {:.3f}".format(recall_score_test_mlp))

print()


precision_score_train_mlp = metrics.precision_score(y_train,y_train_mlp)

precision_score_test_mlp = metrics.precision_score(y_test,y_test_mlp,average="micro")

print("Multi-layer Perceptron : precision on training Data: {:.3f}".format(precision_score_train_mlp))

print("Multi-layer Perceptron : precision on test Data: {:.3f}".format(precision_score_test_mlp))

#storing the results. The below mentioned order of parameter passing is important.


storeResults('Multi-layer Perceptron',acc_test_mlp,f1_score_test_mlp,

        recall_score_train_mlp,precision_score_train_mlp)
```

## 6. Comparision of Models

To compare the models performance, a dataframe is created. The columns of this dataframe are the lists created to store the results of the model.

```
#creating dataframe

result = pd.DataFrame({ 'ML Model' : ML_Model,
                    'Accuracy' : accuracy,
                    'f1_score' : f1_score,
                    'Recall'   : recall,
                    'Precision': precision,
                  })
# dispalying total result
result
#Sorting the datafram on accuracy
sorted_result=result.sort_values(by=['Accuracy', 'f1_score'],ascending=False).reset_index(drop=True)
# dispalying total result
sorted_result
## Storing Best Model
#  XGBoost Classifier Model
from xgboost import XGBClassifier

# instantiate the model
gbc = GradientBoostingClassifier(max_depth=4,learning_rate=0.7)

# fit the model
gbc.fit(X_train,y_train)
import pickle

# dump information to that file
pickle.dump(gbc, open('model.pkl', 'wb'))
#checking the feature improtance in the model
```

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

n_features = X_train.shape[1]

plt.barh(range(n_features), gbc.feature_importances_, align='center')

plt.yticks(np.arange(n_features), X_train.columns)

plt.title("Feature importances using permutation on full model")

plt.xlabel("Feature importance")

plt.ylabel("Feature")

plt.show()
```

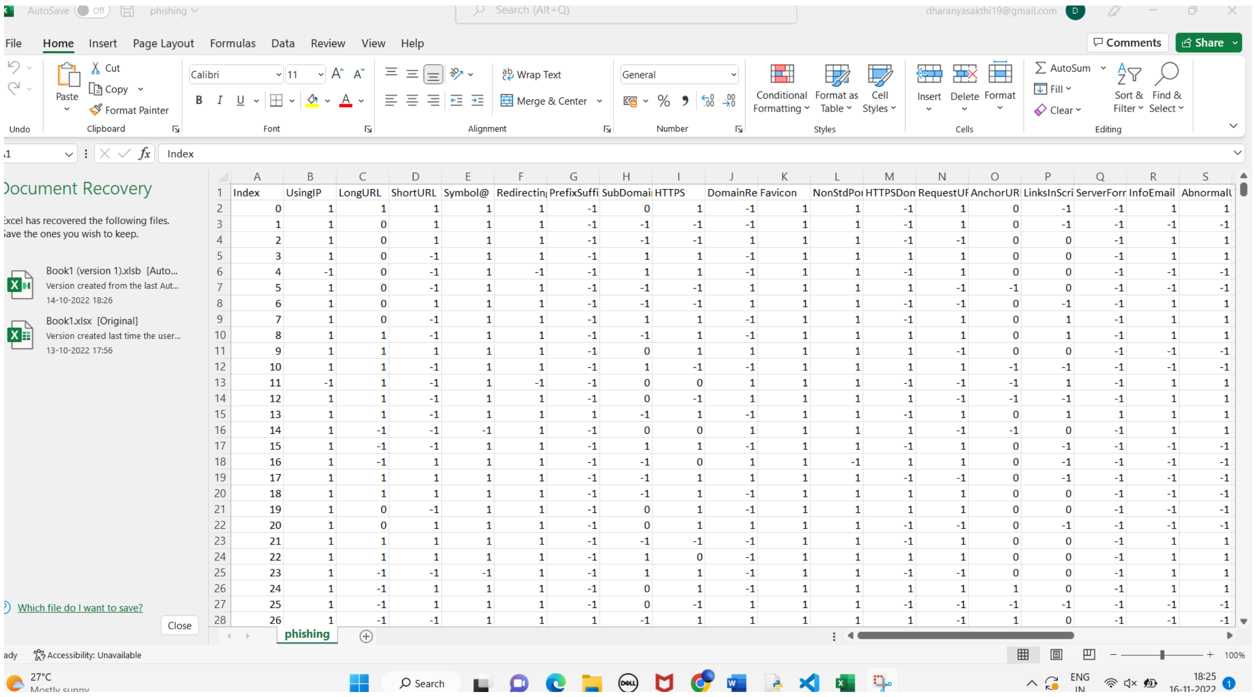
## **7. Conclusion**

1. The final take away form this project is to explore various machine learning models, perform Exploratory Data Analysis on phishing dataset and understanding their features.

2. Creating this notebook helped me to learn a lot about the features affecting the models to detect whether URL is safe or not, also I came to know how to tuned model and how they affect the model performance.

3. The final conclusion on the Phishing dataset is that the some feature like "HTTTPS", "AnchorURL", "WebsiteTraffic" have more importance to classify URL is phishing URL or not.

4. Gradient Boosting Classifier currectly classify URL upto 97.4% respective classes and hence reduces the chance of malicious attachments.



Document Recovery

Excel has recovered the following files. Save the ones you wish to keep.

Book1 (version 1).xlsb [Auto...
Version created from the last Aut...
14-10-2022 18:26

Book1.xlsx [Original]
Version created last time the user...
13-10-2022 17:56

Which file do I want to save?

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Index	UsingIP	LongURL	ShortURL	Symbol@	Redirecting	PrefixSuffi	SubDomai	HTTPS	DomainRe	Favicon	NonStdPoı	HTTPSDom	RequestUF	AnchorURI	LinksInScri	ServerForr	InfoEmail	AbnormalU
2	0	1	1	1	1	1	-1	0	1	-1	1	1	-1	1	0	-1	-1	1	1
3	1	1	0	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	-1	-1	-1	-1
4	2	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	0	-1	1	1
5	3	1	0	-1	1	1	-1	1	1	-1	1	1	1	1	0	0	-1	1	1
6	4	-1	0	-1	1	-1	-1	1	1	-1	1	1	-1	1	0	0	-1	-1	-1
7	5	1	0	-1	1	1	-1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1
8	6	1	0	1	1	1	-1	-1	-1	1	1	1	-1	-1	0	-1	-1	1	1
9	7	1	0	-1	1	1	-1	1	1	-1	1	1	-1	1	0	1	-1	1	1
10	8	1	1	-1	1	1	-1	-1	1	-1	1	1	1	1	0	1	-1	1	1
11	9	1	1	1	1	1	-1	0	1	1	1	1	1	-1	0	0	-1	-1	-1
12	10	1	1	-1	1	1	-1	1	-1	-1	1	1	1	1	-1	-1	-1	-1	-1
13	11	-1	1	-1	1	-1	-1	0	0	1	1	1	-1	-1	-1	1	-1	1	1
14	12	1	1	-1	1	1	-1	0	-1	1	1	1	1	-1	-1	-1	-1	1	1
15	13	1	1	-1	1	1	1	-1	1	-1	1	1	-1	1	0	1	1	1	1
16	14	1	-1	-1	-1	1	-1	0	0	1	1	1	1	-1	-1	0	-1	1	1
17	15	1	-1	-1	1	1	-1	1	1	-1	1	1	-1	1	0	-1	-1	-1	-1
18	16	1	-1	1	1	1	-1	-1	0	1	1	-1	1	1	0	-1	-1	-1	-1
19	17	1	1	1	1	1	-1	-1	1	1	1	1	-1	-1	0	-1	-1	-1	-1
20	18	1	1	1	1	1	-1	-1	1	-1	1	1	1	1	0	0	-1	-1	-1
21	19	1	0	-1	1	1	-1	0	1	-1	1	1	1	1	0	0	-1	-1	-1
22	20	1	0	1	1	1	-1	0	1	1	1	1	-1	-1	0	-1	-1	-1	-1
23	21	1	1	1	1	1	-1	-1	-1	-1	1	1	-1	1	0	0	-1	1	1
24	22	1	1	1	1	1	-1	1	0	-1	1	1	1	1	0	0	-1	1	1
25	23	1	-1	-1	-1	1	-1	1	1	-1	1	1	-1	-1	0	0	-1	1	1
26	24	1	-1	1	1	1	-1	0	1	-1	1	1	1	1	1	0	-1	1	1
27	25	1	-1	1	1	1	-1	0	-1	1	1	1	-1	-1	-1	-1	-1	-1	-1
28	26	1	-1	-1	1	1	1	-1	1	1	1	1	1	-1	1	0	-1	-1	-1

README.me

-

# Phishing URL Detection
![image](https://user-images.githubusercontent.com/79131292/144742825-23367f0f-9e67-4c99-ba1f-b86a187675c9.png)
![image](https://user-images.githubusercontent.com/79131292/144742785-d183f50a-52d6-4296-a43a-90a1ee3502d8.png)

## Table of Content
* [Introduction](#introduction)
* [Installation](#installation)
* [Directory Tree](#directory-tree)
* [Result](#result)
* [Conclusion](#conclusion)

## Introduction

The Internet has become an indispensable part of our life, However, It also has provided opportunities to anonymously perform malicious activities like Phishing. Phishers try to deceive their victims by social engineering or creating mockup websites to steal information such as account ID, username, password from individuals and organizations. Although many methods have been proposed to detect phishing websites, Phishers have evolved their methods to escape from these detection methods. One of the most successful methods for detecting these malicious activities is Machine Learning. This is because most Phishing attacks have some common characteristics which can be identified by machine learning methods. To see project click [here]("/").

## Installation
The Code is written in Python 3.6.10. If you don't have Python installed you can find it [here](https://www.python.org/downloads/). If you are using a lower version of Python you can upgrade using the pip package, ensuring you have the latest version of pip. To install the required packages and libraries, run this command in the project directory after [cloning](https://www.howtogeek.com/451360/how-to-clone-a-github-repository/) the repository:
```bash
pip install -r requirements.txt
```

## Directory Tree
```
├── pickle
│   ├── model.pkl
├── static
```

~~~~
│   ├── styles.css
├── templates
│   ├── index.html
├── Phishing URL Detection.ipynb
├── Procfile
├── README.md
├── app.py
├── feature.py
├── phishing.csv
├── requirements.txt

```

## Technologies Used

![](https://forthebadge.com/images/badges/made-with-python.svg)

[<img target="_blank" src="https://upload.wikimedia.org/wikipedia/commons/3/31/NumPy_logo_2020.svg" width=200>](https://numpy.org/doc/) [<img target="_blank" src="https://upload.wikimedia.org/wikipedia/commons/e/ed/Pandas_logo.svg" width=200>](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html)
[<img target="_blank" src="https://upload.wikimedia.org/wikipedia/commons/8/84/Matplotlib_icon.svg" width=100>](https://matplotlib.org/)
[<img target="_blank" src="https://scikit-learn.org/stable/_static/scikit-learn-logo-small.png" width=200>](https://scikit-learn.org/stable/)
[<img target="_blank" src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcScq-xocLctL07Jy0tpR_p9w0Q42_rK1aAkNfW6sm3ucjFKWML39aaJPgdhadyCnEiK7vw&usqp=CAU" width=200>](https://flask.palletsprojects.com/en/2.0.x/)

## Result

Accuracy of various model used for URL detection
<br>

<br>

||ML Model| Accuracy|   f1_score| Recall| Precision|
|---|---|---|---|---|---|---|
0|  Gradient Boosting Classifier| 0.974|  0.977|  0.994|  0.986|
1|  CatBoost Classifier|         0.972|  0.975|  0.994|  0.989|
~~~~

2	XGBoost Classifier	0.969	0.973	0.993	0.984
3	Multi-layer Perceptron	0.969	0.973	0.995	0.981
4	Random Forest	0.967	0.971	0.993	0.990
5	Support Vector Machine	0.964	0.968	0.980	0.965
6	Decision Tree	0.960	0.964	0.991	0.993
7	K-Nearest Neighbors	0.956	0.961	0.991	0.989
8	Logistic Regression	0.934	0.941	0.943	0.927
9	Naive Bayes Classifier	0.605	0.454	0.292	0.997

Feature importance for Phishing URL Detection
<br><br>
![image](https://user-images.githubusercontent.com/79131292/144603941-19044aae-7d7b-4e9a-88a8-6adfd8626f77.png)

7.Conclusion

1. The final take away form this project is to explore various machine learning models, perform Exploratory Data Analysis on phishing dataset and understanding their features.
2. Creating this notebook helped me to learn a lot about the features affecting the models to detect whether URL is safe or not, also I came to know how to tuned model and how they affect the model performance.
3. The final conclusion on the Phishing dataset is that the some feature like "HTTTPS", "AnchorURL", "WebsiteTraffic" have more importance to classify URL is phishing URL or not.
4. Gradient Boosting Classifier currectly classify URL upto 97.4% respective classes and hence reduces the chance of malicious attachments.

Requirements.txt

```
beautifulsoup4==4.9.3
Flask==2.0.2
googlesearch_python==1.0.1
numpy==1.21.4
pandas==1.3.4
python_dateutil==2.8.2
requests==2.25.1
scikit_learn==1.0.1
whois==0.9.13
gunicorn==20.1.0
```

styles.css

```
-
*,
*::after,
*::before {
  margin: 0;
  padding: 0;
  box-sizing: inherit;
  font-size: 62,5%;
}

body {
  padding: 10% 5%;
  background: #14658a;
  background: linear-gradient(to right,#134558, #173444, #296e83);
  justify-content: center;
  align-items: center;
  height: 100vh;
  color: #fff;
}

.form__label {
  font-family: 'Roboto', sans-serif;
  font-size: 1.2rem;
  margin-left: 2rem;
  margin-top: 0.7rem;
  display: block;
  transition: all 0.3s;
  transform: translateY(0rem);
}

.form__input {
  top: -24px;
  font-family: 'Roboto', sans-serif;
  color: #333;
  font-size: 1.2rem;
  padding: 1.5rem 2rem;
  border-radius: 0.2rem;
  background-color: rgb(255, 255, 255);
  border: none;
  width: 75%;
  display: block;
  border-bottom: 0.3rem solid transparent;
  transition: all 0.3s;
}

.form__input:placeholder-shown + .form__label {
  opacity: 0;
  visibility: hidden;
  -webkit-transform: translateY(+4rem);
  transform: translateY(+4rem);
}
```

```
.button {
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, #fff, #f8eedb);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color
Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
}

.button:active {
  background-color: #f3f4f6;
  box-shadow: -1px 2px 5px rgba(81,41,10,0.15),0px 1px 1px rgba(81,41,10,0.15);
  transform: translateY(0.125rem);
}

.button:focus {
  box-shadow: rgba(72, 35, 7, .46) 0 0 0 4px, -6px 8px 10px rgba(81,41,10,0.1), 0px 2px 2px
rgba(81,41,10,0.2);
}
```

```css
.main-body{
  display: flex;
  flex-direction: row;
  width: 75%;
  justify-content:space-around;
}

.button1{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(160, 245, 174), #37ee65);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color
Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}

.button2{
  appearance: button;
  background-color: transparent;
  background-image: linear-gradient(to bottom, rgb(252, 162, 162), #ee3737);
  border: 0 solid #e5e7eb;
  border-radius: .5rem;
  box-sizing: border-box;
  color: #482307;
  column-gap: 1rem;
  cursor: pointer;
  display: flex;
  font-family: ui-sans-serif,system-ui,-apple-system,system-ui,"Segoe UI",Roboto,"Helvetica
Neue",Arial,"Noto Sans",sans-serif,"Apple Color Emoji","Segoe UI Emoji","Segoe UI Symbol","Noto Color
Emoji";
  font-size: 100%;
  font-weight: 700;
  line-height: 24px;
  margin: 0;
  outline: 2px solid transparent;
  padding: 1rem 1.5rem;
  text-align: center;
  text-transform: none;
  transition: all .1s cubic-bezier(.4, 0, .2, 1);
  user-select: none;
  -webkit-user-select: none;
  touch-action: manipulation;
  box-shadow: -6px 8px 10px rgba(81,41,10,0.1),0px 2px 2px rgba(81,41,10,0.2);
  display: none;
}

.right {
  right: 0px;
  width: 300px;
}

@media (max-width: 576px) {
  .form {
    width: 100%;
  }
 }
.abc{
  width: 50%;
}
```

GitHub

https://github.com/IBM-EPBL/IBM-Project-5208-1658751568

Project Demo Link

https://drive.google.com/drive/folders/1d4CCH0CfDO9d86b_MumISEm5274RUEmM?usp=share_link