**Machine Learning Model Deployment with IBMCloud Watson Studio**

**Innovations and Enhancements**

**A PROJECT REPORT**

**Submitted by**

HALISH RICHARD J (**TL**)

MOHAMED ATHIL M

ARUSAMY A

RAHUL DRAVID A

of

COMPUTER SCIENCE AND ENGINEERING

Parisutham Institute of Technology and Science

**CHAPTER NO TITLE PAGE NO**

1 INTRODUCTION

1.1 Project Overview 3

1.2 Purpose 3

2. LITERATURE SURVEY

2.1 Existing problem 3

2.2 Problem Definition 4

3. REQUIREMENT ANALYSIS

3.1 Project Requirement 4

3.2 Functional requirement 4

4 ARCHITECTURE

4.1Technical Architecture 4

5 SETUP WATSON

5.1 Create a new Cloud Pak for Data project 6

5.2 Import require Data Set as Train 8

5.3 Build a model with AutoAI 9

5.4 Deploy the model with WML 10

6 CODING & SOLUTIONING

6.2 Copy private end point link and API 15

6.3 Develop flask program 15

6.5 Develop web site for UI 17

7 TESTING & RESULT

7.1 Output 23

7.2 Conclusion 24

8 APPENDIX

8.1 GitHub & Project Demo Link 24

8.2 REFERANCE 25

**CHAPTER 1**

**INTRODUCTION**

**1.1 Project Overview:**

Our solution is to build an efficient and intelligent system to detect phishing sites by applying a machine learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy by carefully analysing and identifying various factors that could be used to detect a phishing site. These factors fall under the categories of address bar-based features, domain-based features, HTML & JavaScript based features. Using these features, we can identify a phishing site with high accuracy

**1.2 Purpose:**

Web phishing is a threat in various aspects of security on the internet, which might involve scams and private information disclosure. Some of the common threats of web phishing are:

* Attempt to fraudulently solicit personal information from an individual or organization.
* Attempt to deliver malicious software by posing as a trustworthy organization or entity.
* Installing those malwares infects the data that cause a data breach or even nature’s forces that takes down your company’s data headquarters, disrupting access.

For this purpose, the objective of our project involves building an efficient and intelligent system to detect such websites by applying a machine-learning algorithm which implements classification algorithms and techniques to extract the phishing datasets criteria to classify their legitimacy and as a result of which whenever a user makes a transaction online and makes payment through an ebanking website our system will use a data mining algorithm to detect whether the e-banking website is a phishing website or not.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 Existing problem:**

There are phishing detection sites out in the web. But they charge users after a limit of usage. Most of them are built on a clean set of features. We have carefully analysed and identified several factors that could be used to detect a phishing site. These factors fall under the categories of address barbased features, domain-based features, HTML & JavaScript based features. Using these features, we build an intelligent system which can identify a phishing site with high accuracy and efficiency. It is also an open-source website which will be easily accessible to all users.

**2.2 Problem Definition**

The objective of this project is to leverage IBM Cloud Watson Studio to train a machine learning model and deploy it as a web service. The project's primary goal is to gain proficiency in predictive analytics by developing a model capable of making real-time predictions. This encompasses defining a predictive use case, selecting an appropriate dataset, training the machine learning model, deploying it as a web service, and integrating it into applications.

**CHAPTER 3**

**REQUIREMENT ANALYSIS**

**3.1 Project Requirement**

* An IBM Cloud Account.
* An account on IBM Cloud Pak for Data.

**3.2 Functional requirement**

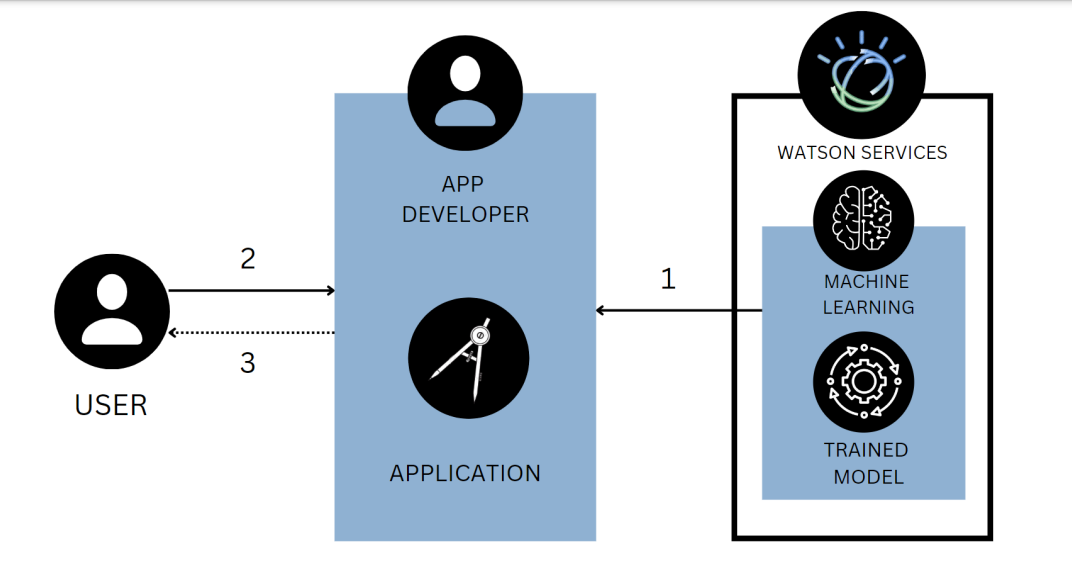
|  |  |
| --- | --- |
| **Functional Requirement** | **Description** |
| User Input | User inputs an URL in the form to check whether it is a malicious website. |
| Website comparison | The model compares the given URL with the list of phishing URLs present in the database. |
| Feature Extraction | If it is found none on the comparison it extracts the HTML and domain-based features from the URL. |
| Prediction | The model predicts the URL using machine Learning algorithms such as Random Forest technique. |
| Classifier | Model then sends the output to the classifier and produces the result. |

**CHAPTER 4**

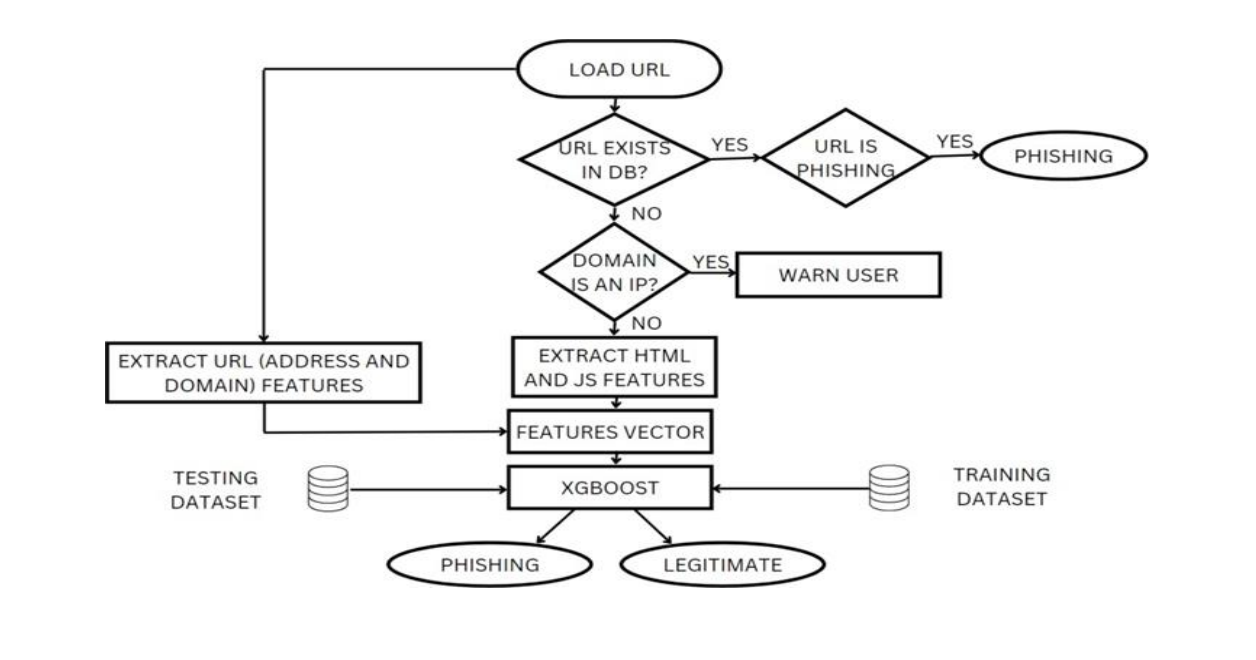
**ARCHITECTURE**

**4.1Technical Architecture:**

Technical architecture which is also often referred to as application architecture includes the major components of the system, their relationships, and the contracts that define the interactions between the components. The goal of technical architects is to achieve all the business needs with an application that is optimized for both performance and security.

****

**Fig 1**

****

**Fig 2**

1. The application developer builds a Python-based app and deploys it.

2. The user enters the URL of a website in the application to check for its genuineness.

3. The user submits the URL through the web-based application and gets back the result.

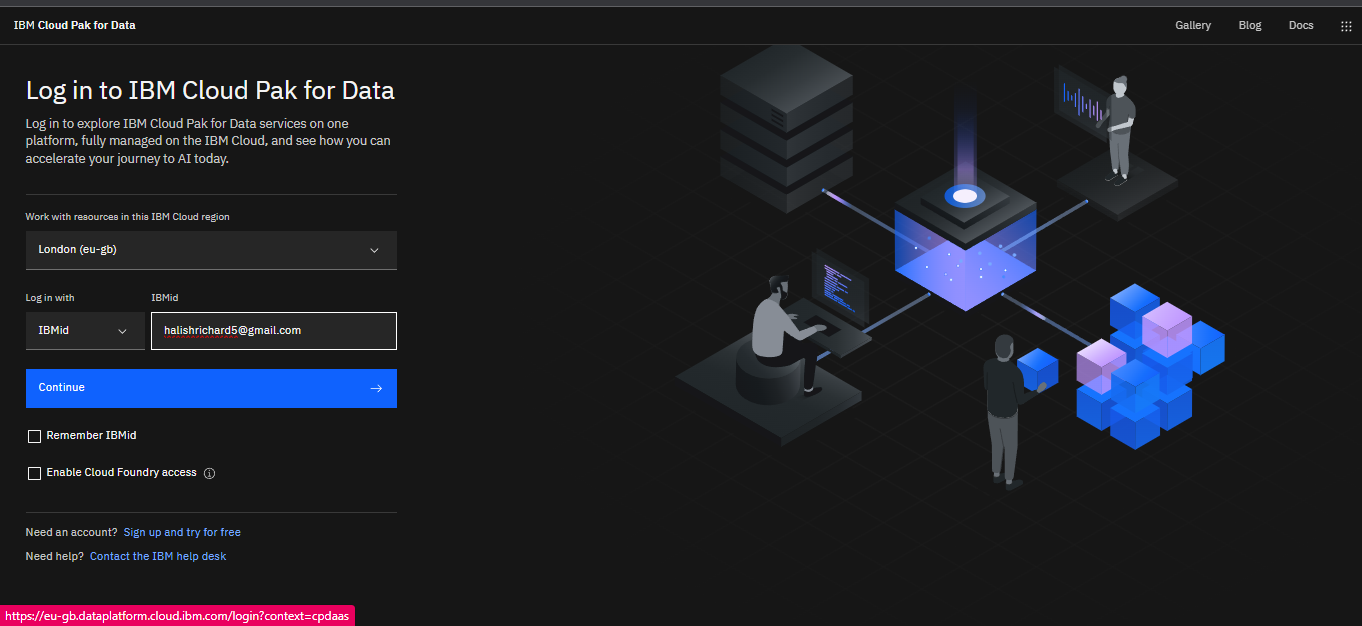
4. The user makes a decision whether to proceed surfing in that website or move to another one.

**CHAPTER 5**

**SETUP WATSON**

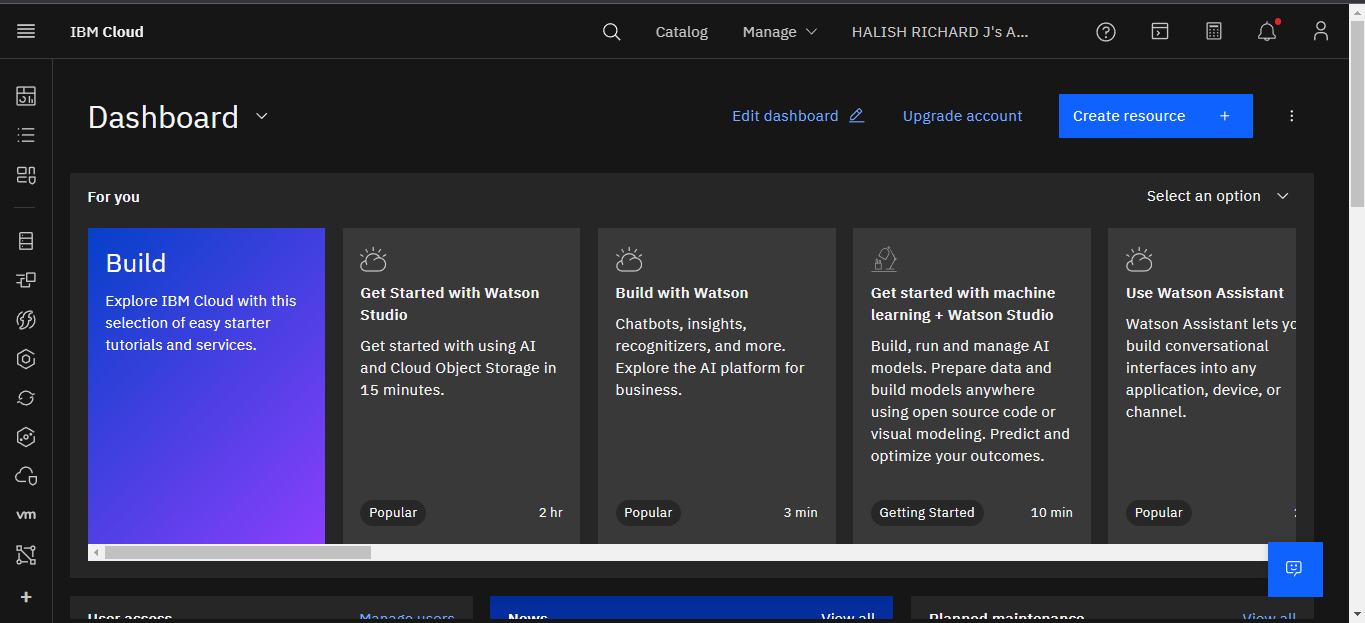
**5.1Create a new Cloud Pak for Data project**

* Log into IBM's [Cloud Pak for Data](https://dataplatform.cloud.ibm.com/) service.

****

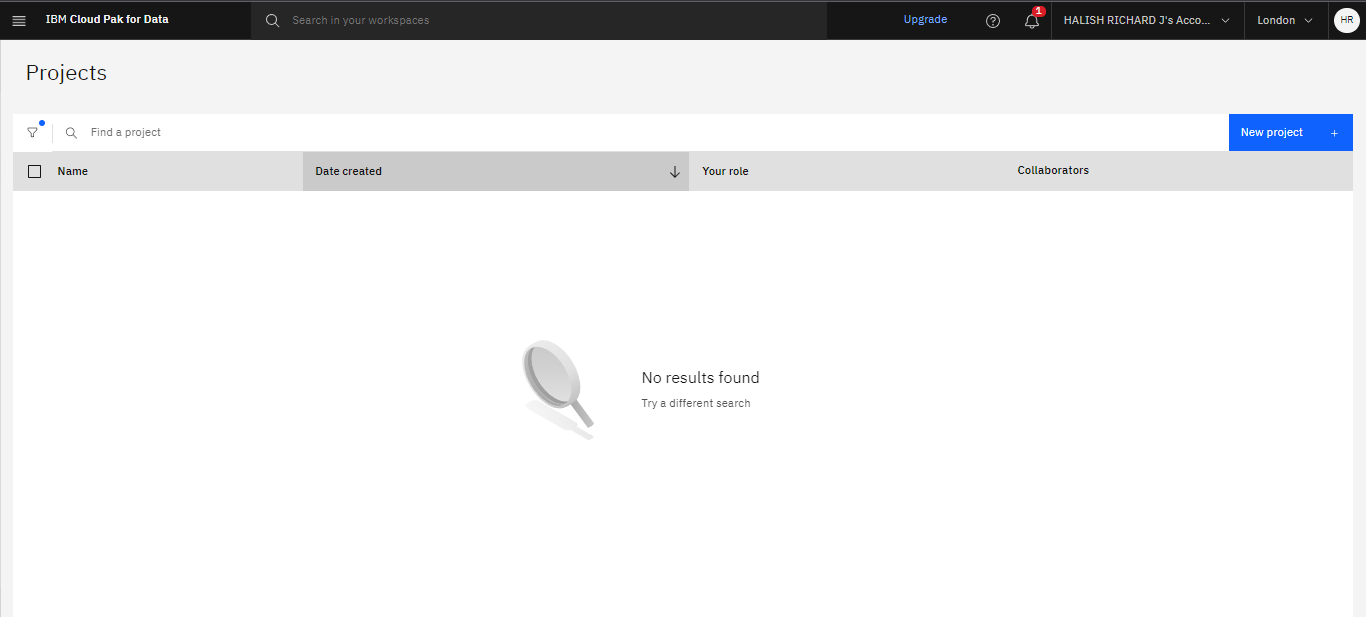
**Fig-3**

Once in, you'll land on the dashboard.



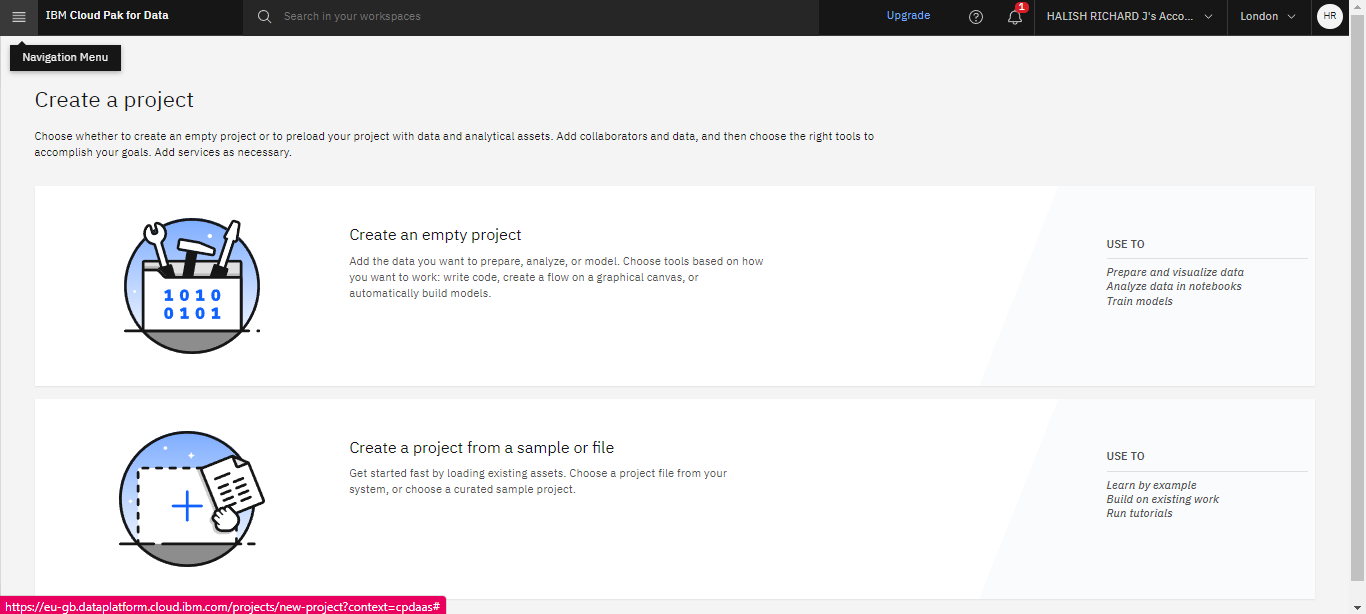
**Fig-4**

* Create a new project by clicking Create a project.



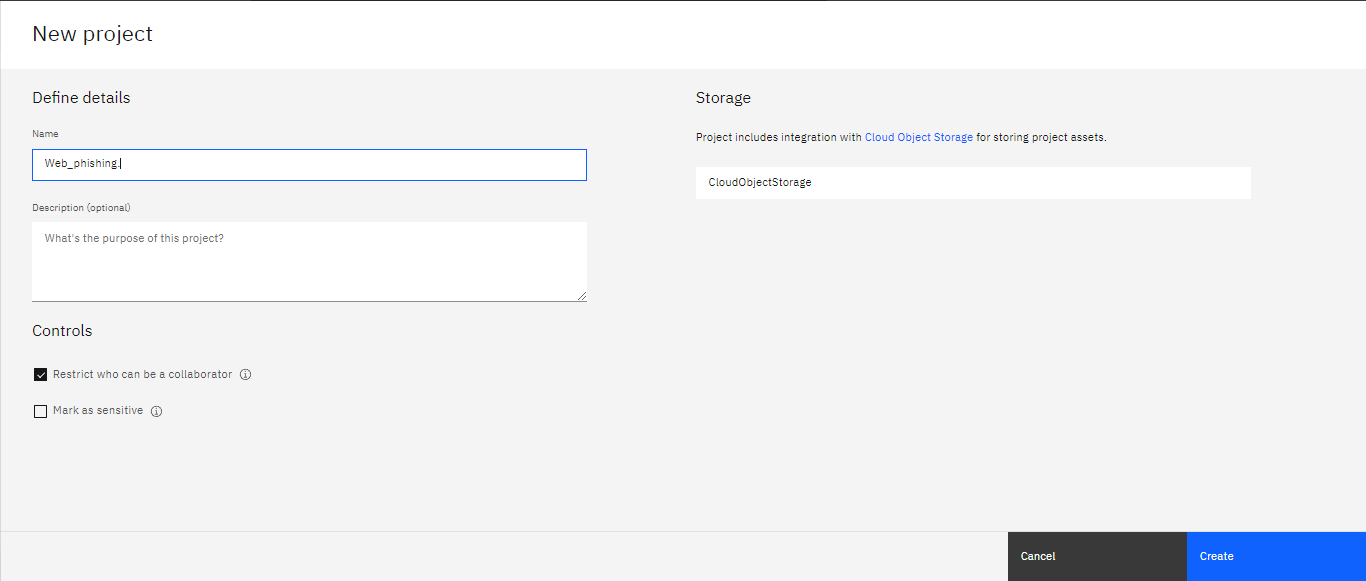
**Fig-5**

* Choose an Empty project.



**Fig-6**

* Enter a  Name and associate the project with a Cloud Object Storage service.

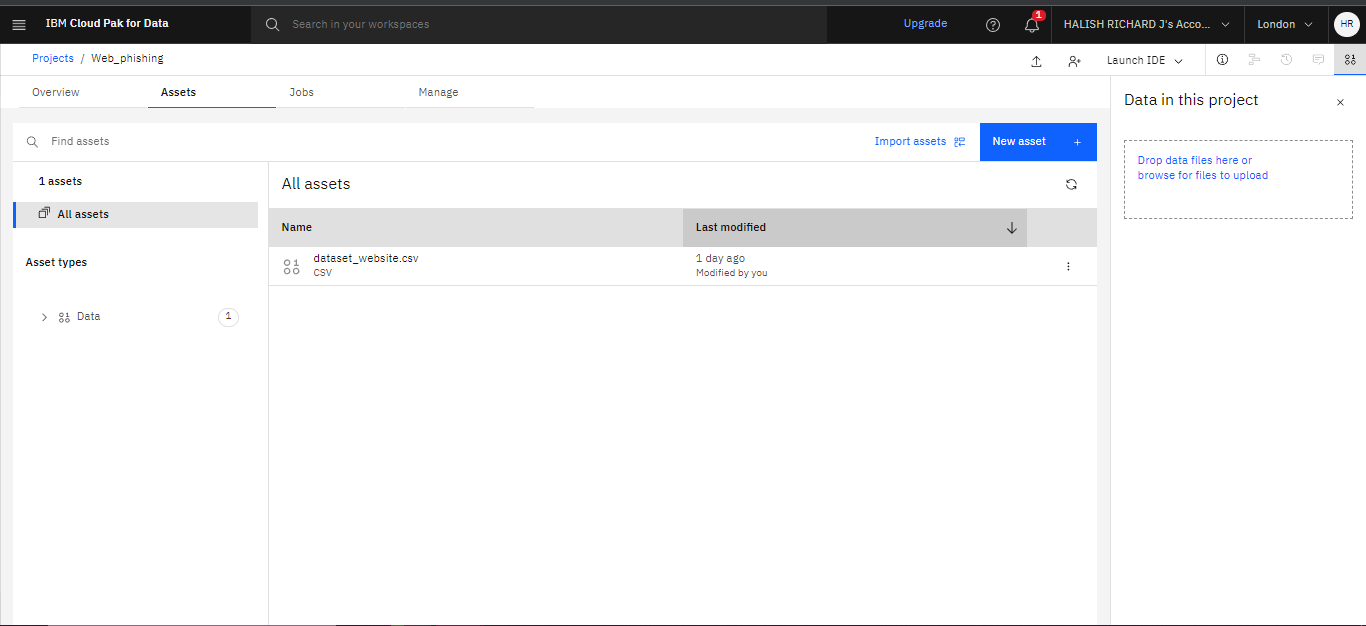


**Fig-7**

**5.2 Import require Data Set as Train**

* At the project dashboard click on the Assets tab and upload the data set associated.

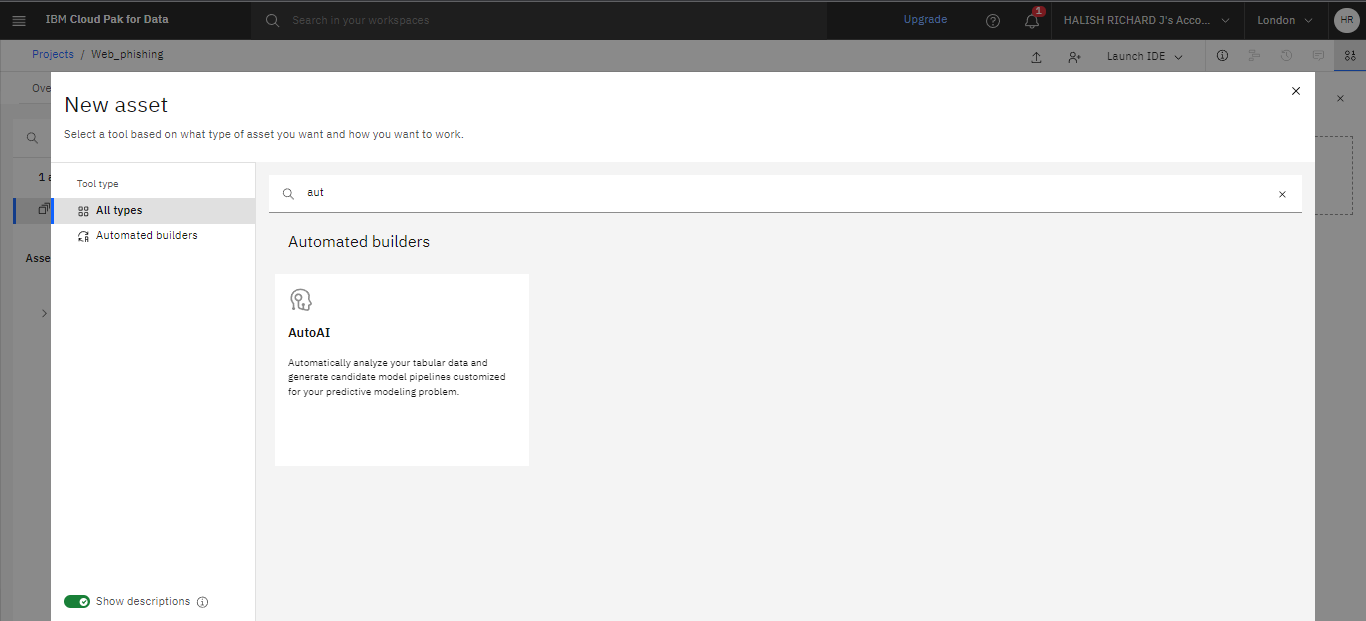
[dataset\_website.csv](https://eu-gb.dataplatform.cloud.ibm.com/projects/6c0c326d-4042-4619-b2dc-fca15946fc78/data-assets/ada65449-8456-4b6f-8923-bf60f16ab9f6?context=cpdaas&walkme_guided_tutorial=false)



**Fig-8**

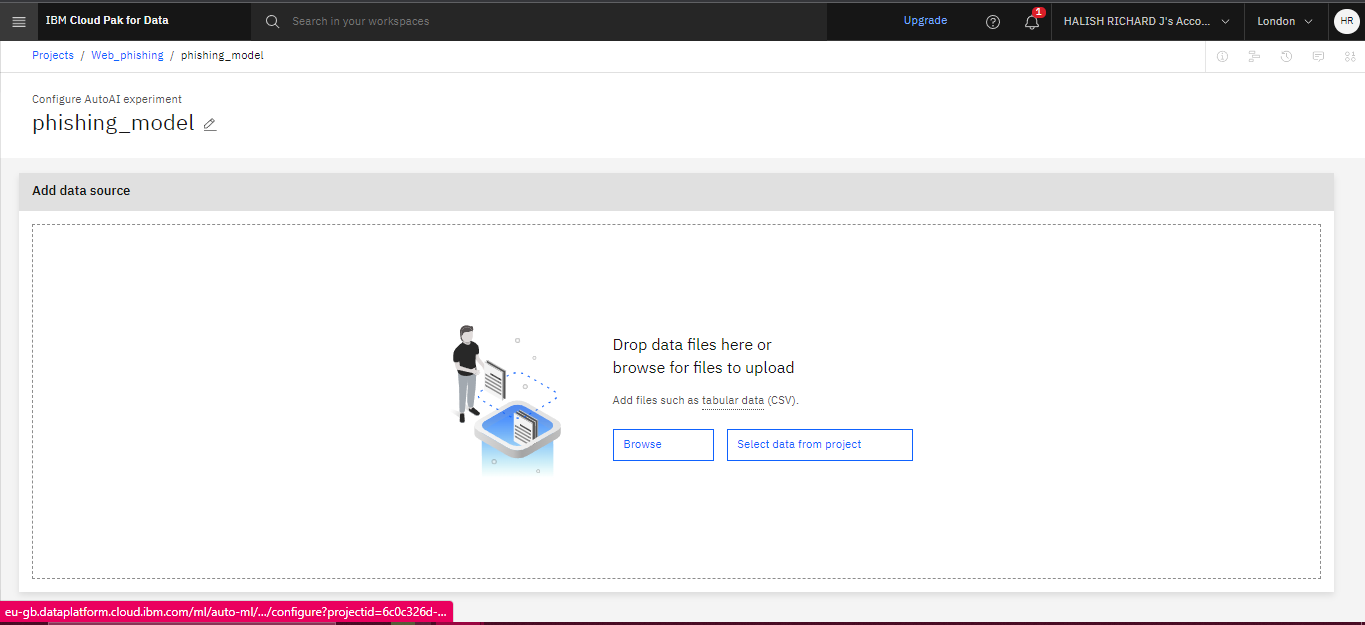
**5.3 Build a model with AutoAI**

* Now we're going to build a model from the data using IBM's AutoAI. A tool that will automatically create multiple models and test them, giving us the best result. Data science made easy.
* Start by clicking on Add to project and choosing AutoAI experiment.



**Fig-9**

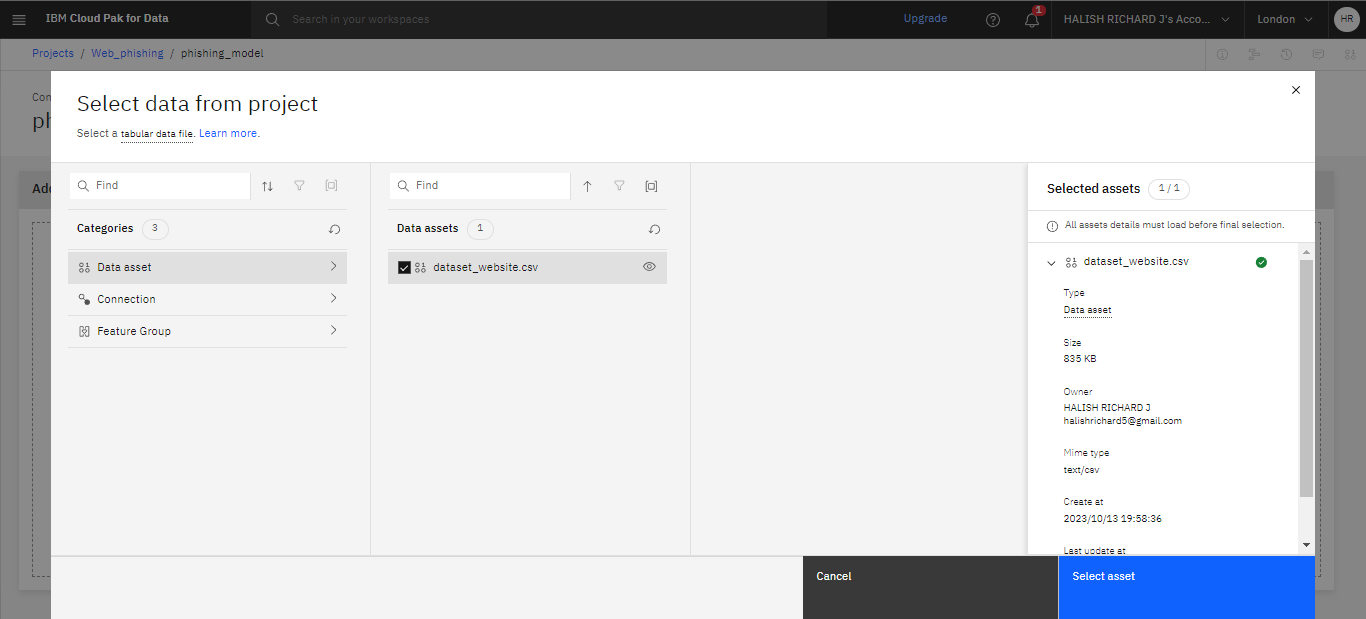
* Choose to use data from your project.



**Fig-10**

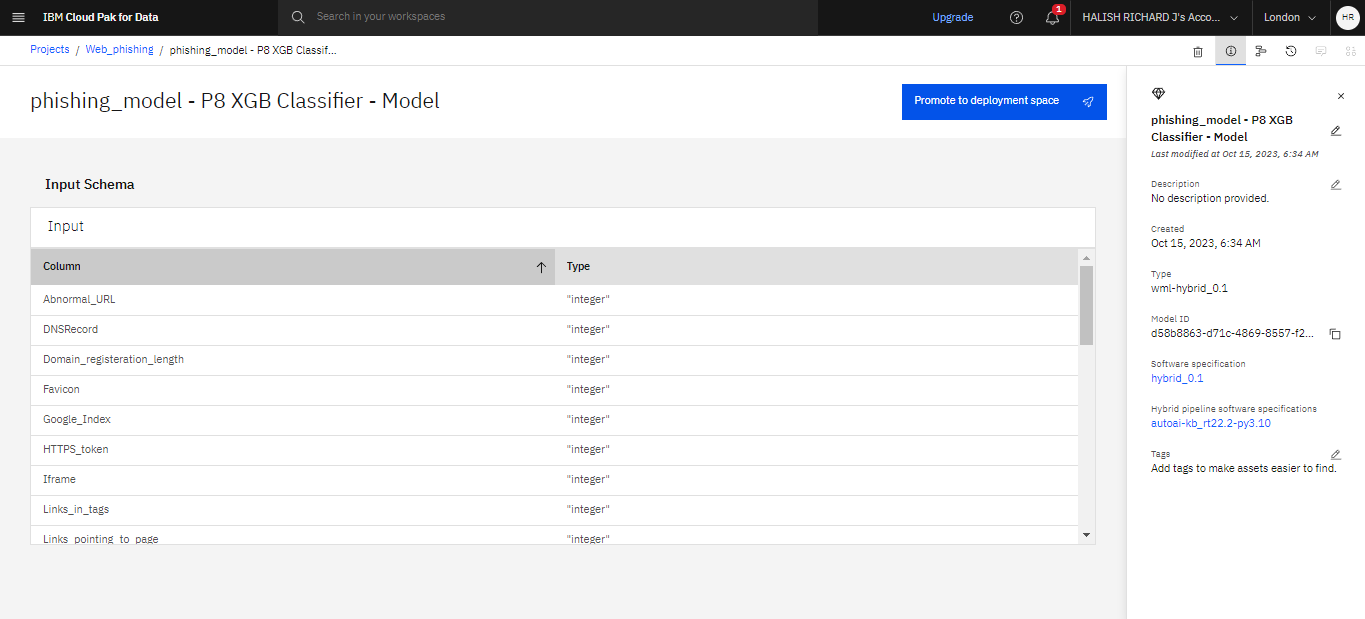
**5.4 Deploy the model with WML**

* Choose the [dataset\_website.csv](https://eu-gb.dataplatform.cloud.ibm.com/projects/6c0c326d-4042-4619-b2dc-fca15946fc78/data-assets/ada65449-8456-4b6f-8923-bf60f16ab9f6?context=cpdaas&walkme_guided_tutorial=false) option.



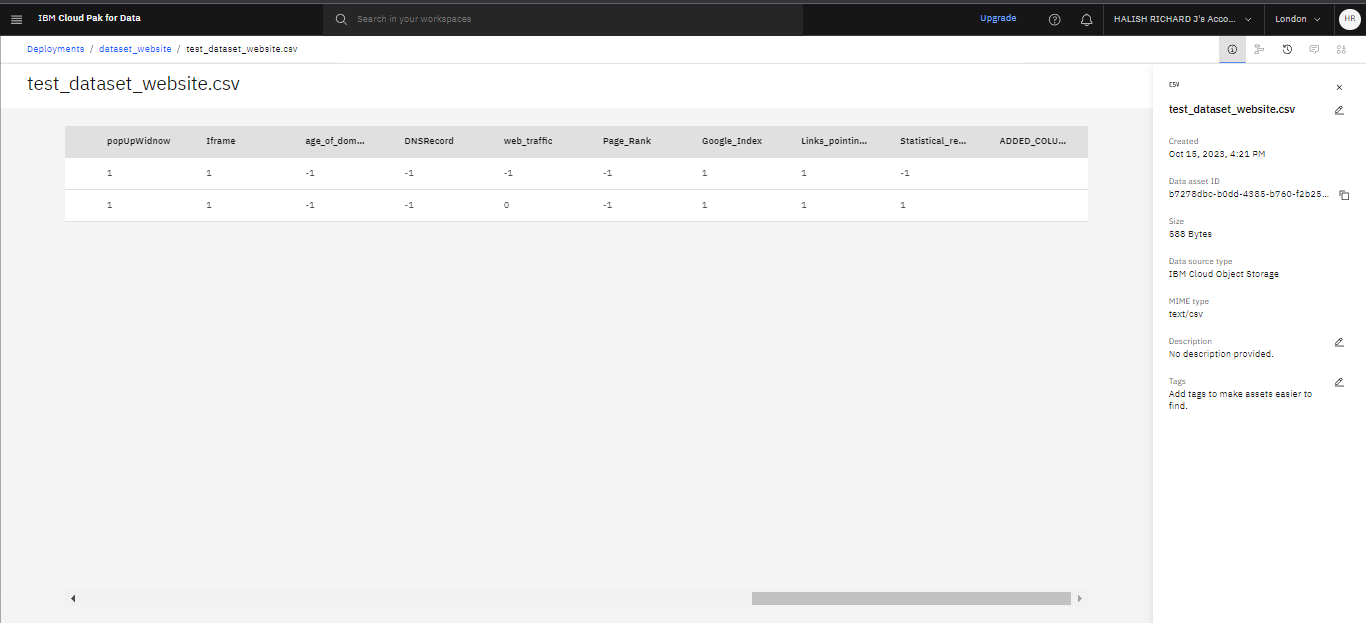
**Fig-11**

* Once you're at the model overview choose the button “Promote to deployment space”.



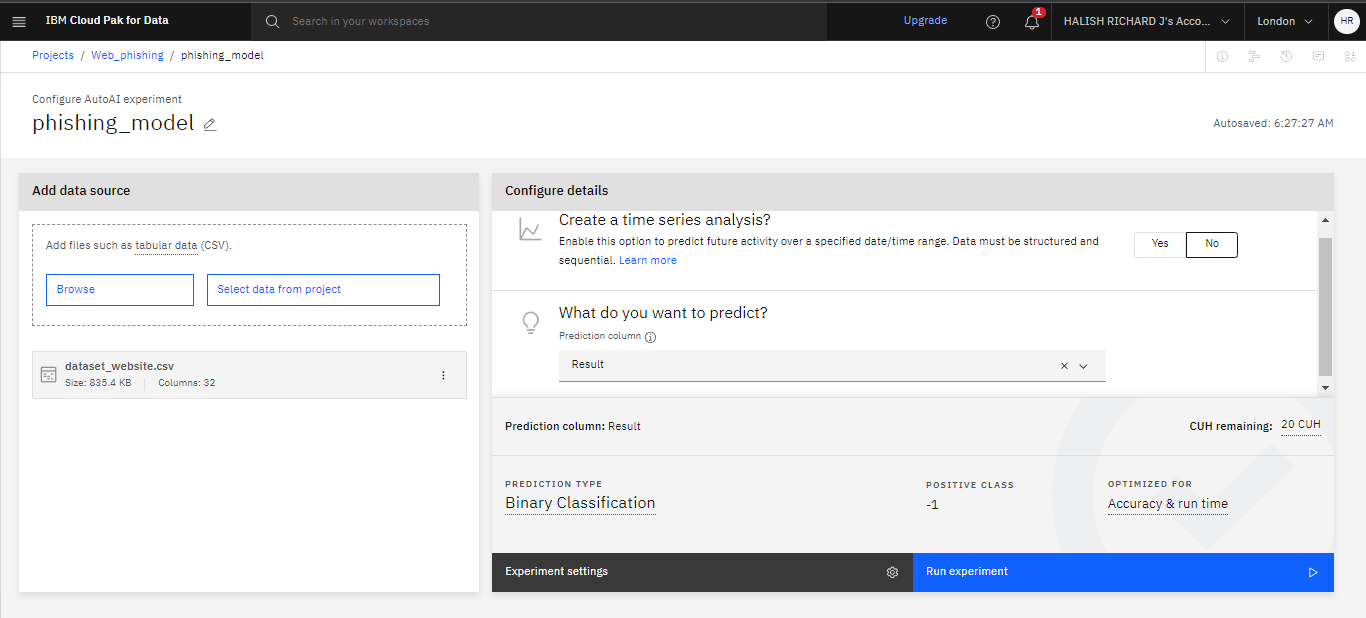
**Fig-12**

test-dataset\_website.csv



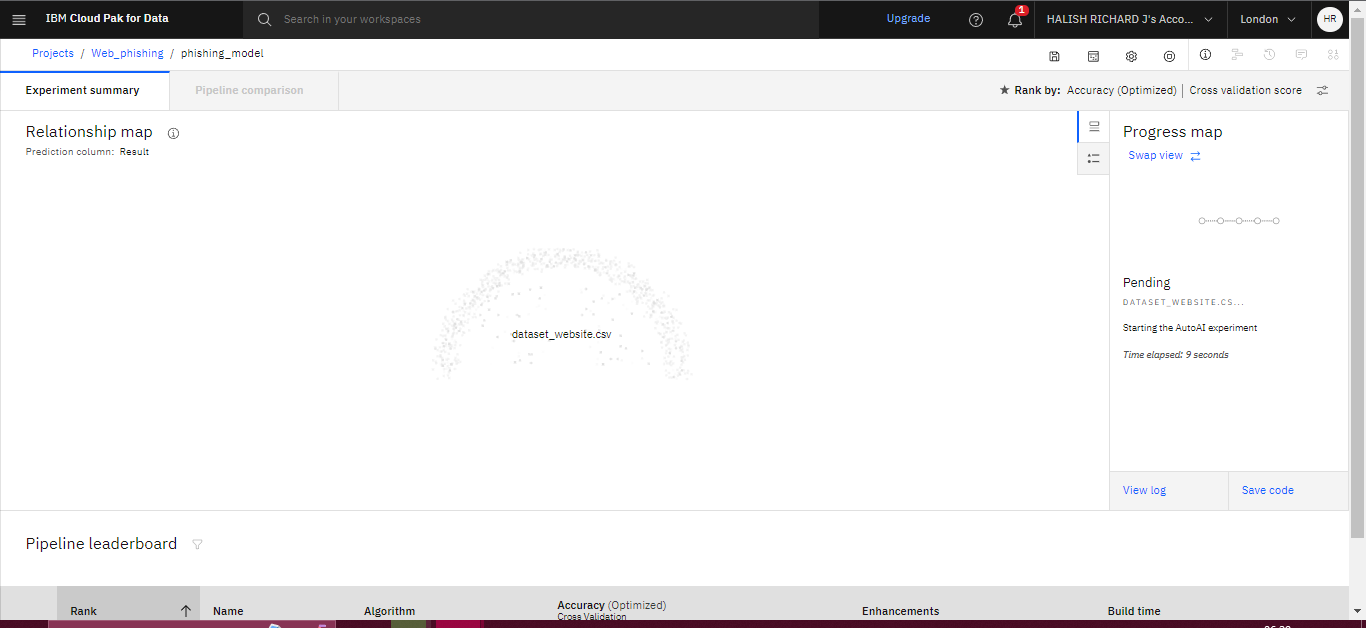
**Fig-13**

* Click “Run Experiment” for create.
* Set prediction column as “Result”.



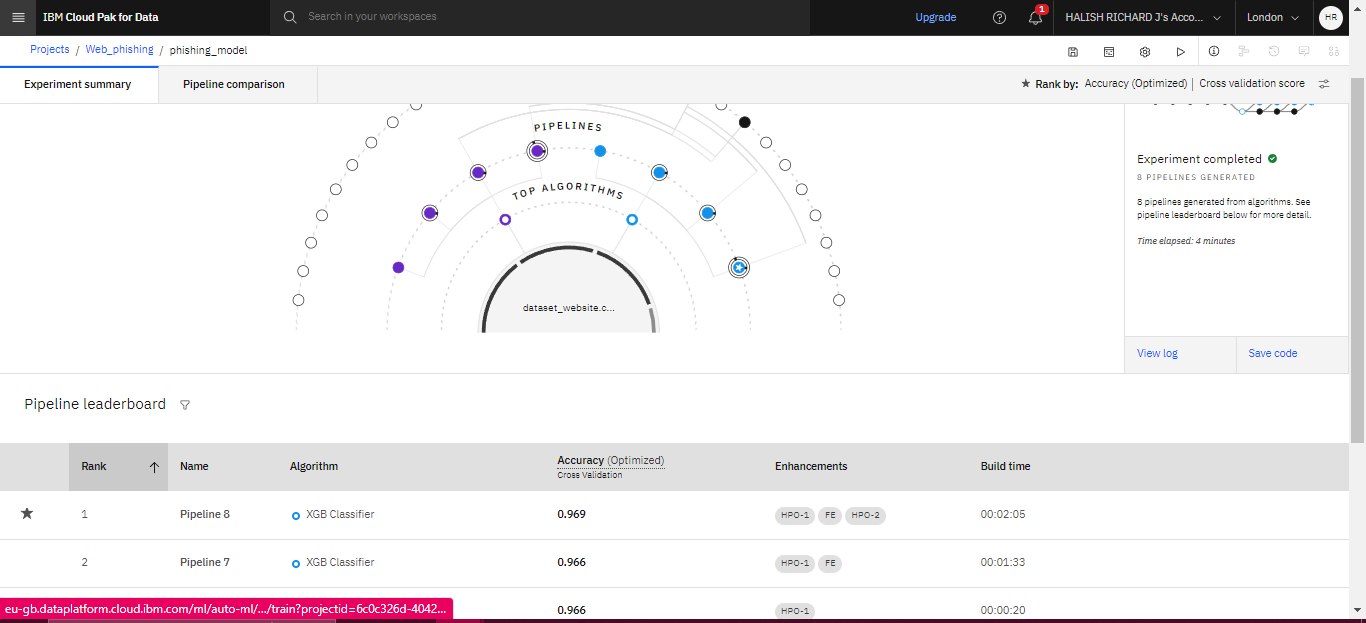
**Fig-14**

* The experiment will take a few minutes to run. Once completed hover over the top option to make the Save as button appear.
* Click it.



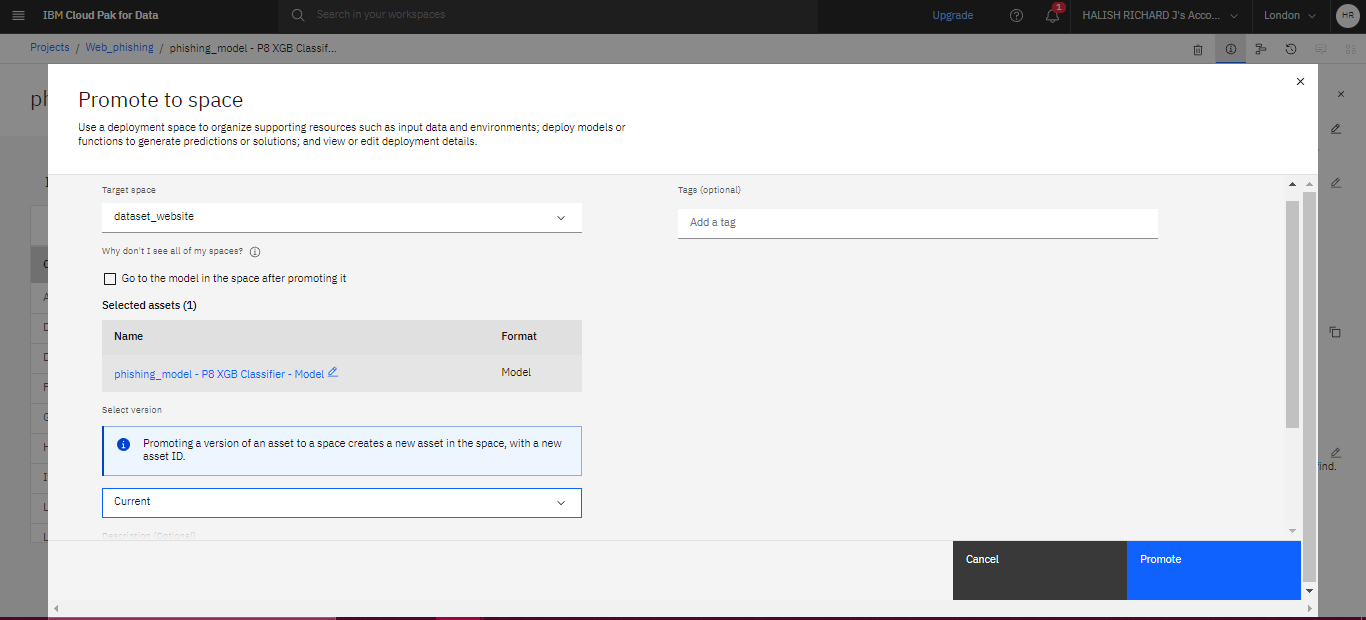
**Fig-15**

* Choose to save the experiment as a Model. You can optionally download a generated Jupyter Notebook that can be used to re-create the steps that were taken to create the model.
* AutoAI also prefer XGBoost Classifier as 0.969 accuracy.



**Fig-16**

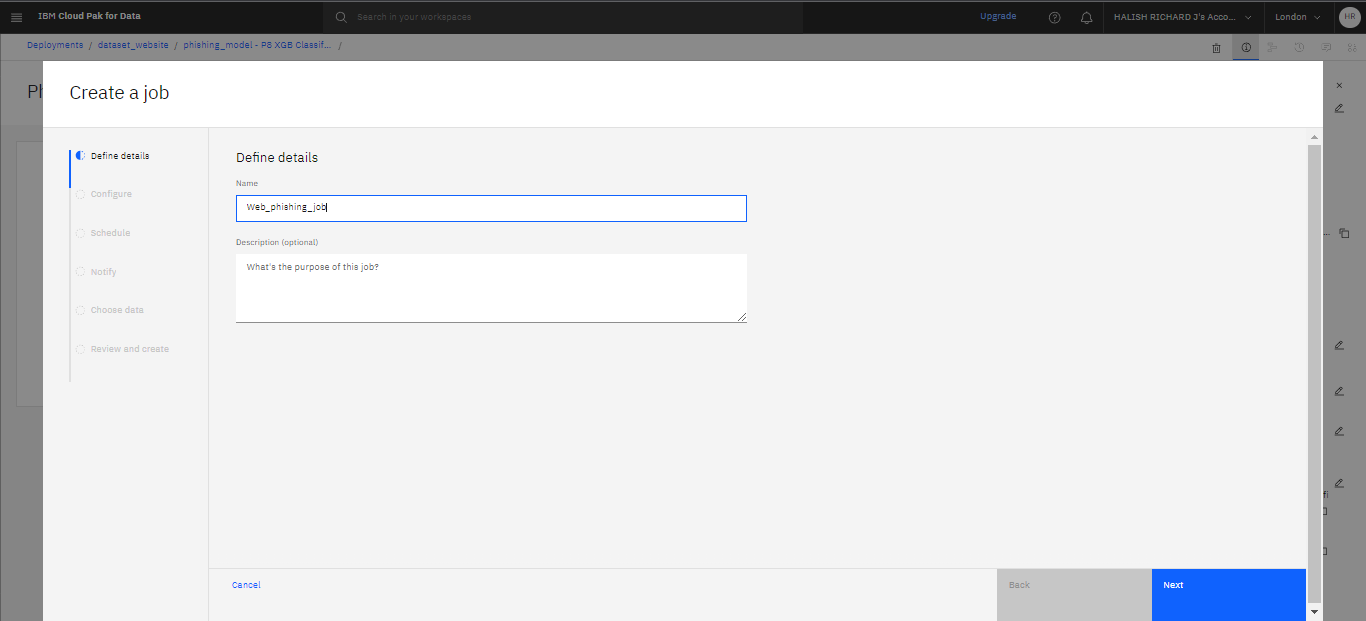
* To promote the model to deployment you must specify a deployment space.
* choose the New space + option to create one. This action will associate the model with the space.
* Select Data Model as “phishing\_model-PBXGB Classifier-Model”.
* Click Promote.



**Fig-17**

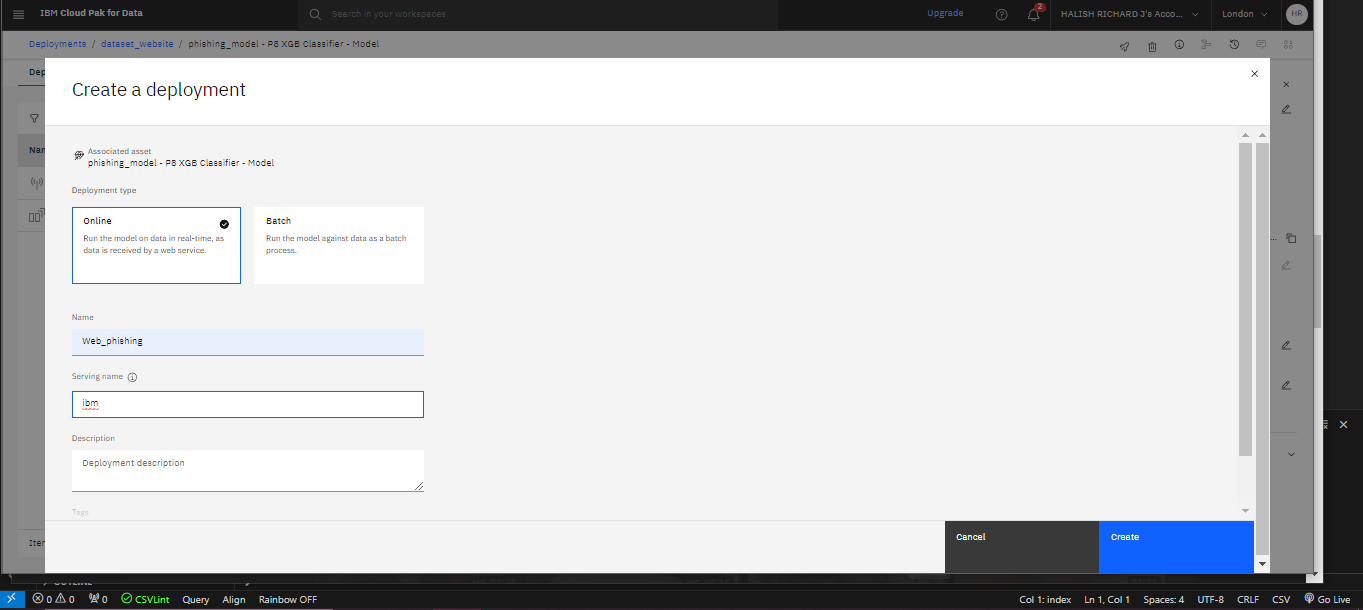
**5.5 Run the Job**

* Enter job Name as “Web\_phishing\_job”.
* Click Next and Next.
* In choose data, Set dataset for train and test.

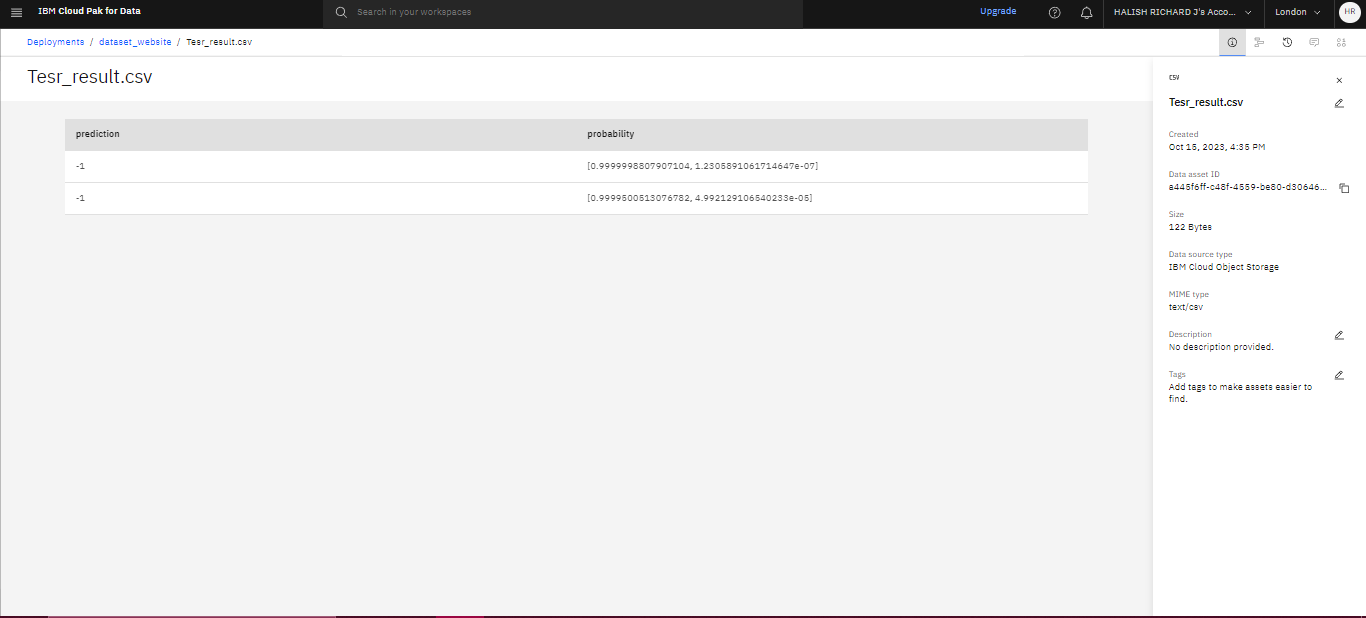


**Fig-18**

* Creating deployment as online type by using Xgboost-Model.
* That can be available for Continue building.

**Fig-19**

* In deployments space occur tesr\_result.csv for output That can be predication value.
* We just give two rows dataset in train dataset so the prediction will be two output.



**Fig-20**

If Prediction value positive that is not suspicious or negative that is suspicious.

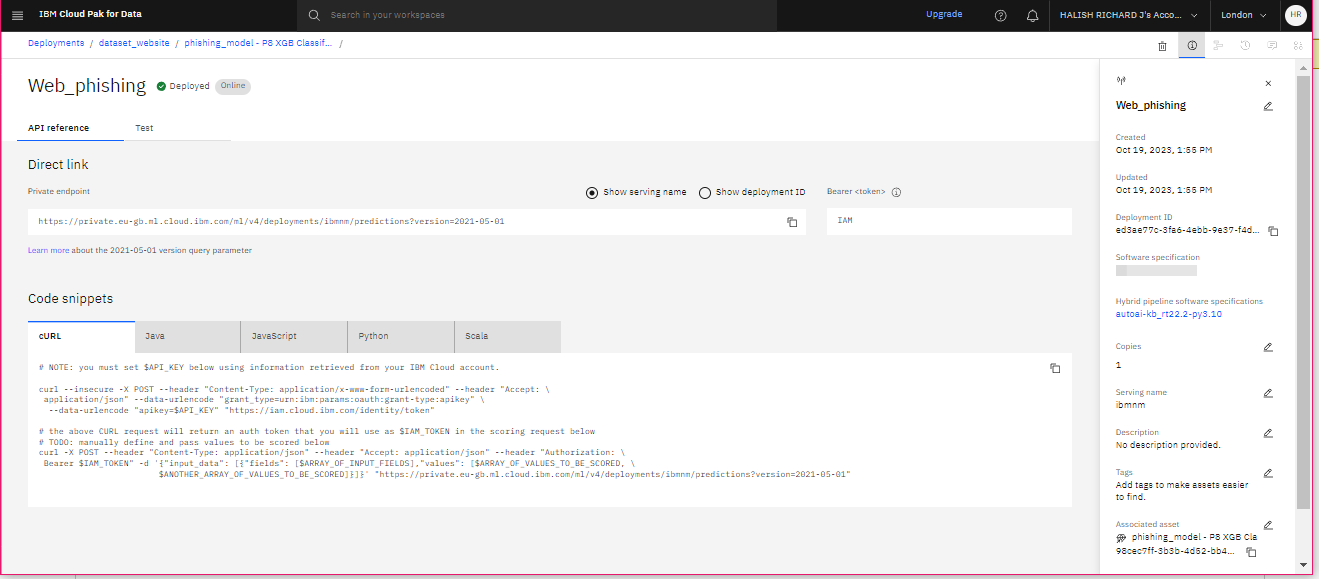
**CHAPTER 6**

**CODING & SOLUTIONING**

**6.1 Copy private end point link and API.**

'https://private.eu-gb.ml.cloud.ibm.com/ml/v4/deployments/ed3ae77c-3fa6-4ebb-9e37-f4d99de5823b/predictions?version=2021-05-01'

That can be connect data model deployment to for our logic program.



**Fig-21**

**6.2 Develop flask program for get the Data model**

Develop a flask application for connect IBM Watson Studio to getting information about our machine learning model.

**app.py**

from flask import Flask, render\_template, request

import numpy as np

import pandas as pd

from sklearn import metrics

import warnings

import pickle

warnings.filterwarnings('ignore')

from features import FeatureExtraction

from flask\_mysqldb import MySQL

import requests

# NOTE: you must manually set API\_KEY below using information retrieved from your IBM Cloud account.

API\_KEY = "6NZgUAO90FAOvaPc-aMSH5Z3l-yfOJPNrSq8cYcGNEde"  # for security reasons haven't displayed API key

token\_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":

                                                                                     API\_KEY,

                                                                                 "grant\_type": 'urn:ibm:params:oauth:grant-type:apikey'})

mltoken = token\_response.json()["access\_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

app = Flask(\_\_name\_\_)

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'root'

app.config['MYSQL\_PASSWORD'] = ''

app.config['MYSQL\_DB'] = 'report'

mysql = MySQL(app)

xgb = pickle.load(open("XGBoostClassifier.pkl", "rb"))

@app.route("/", methods=["GET", "POST"])

def home():

    if request.method == "POST":

        url = request.form["url"]

        obj = FeatureExtraction(url)

        x = np.array(obj.getFeaturesList()).reshape(1, 13)

        print(x)

        t = [obj.getFeaturesList()]

        print("t")

        print(t)

        # NOTE: manually define and pass the array(s) of values to be scored in the next line

        payload\_scoring = {"input\_data": [

            {"fields": [['f0', 'f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10', 'f11', 'f12']],

             "values": t}]}

        response\_scoring = requests.post(

            'https://private.eu-gb.ml.cloud.ibm.com/ml/v4/deployments/ed3ae77c-3fa6-4ebb-9e37-f4d99de5823b/predictions?version=2021-05-01',

            json=payload\_scoring, headers={'Authorization': 'Bearer ' + mltoken})

        print("Scoring response")

        print(response\_scoring.json())

        y\_pred = xgb.predict(x)[0]

        print(y\_pred)

        y\_pro\_phishing = xgb.predict\_proba(x)[0, 0]

        print(y\_pro\_phishing)

        y\_pro\_non\_phishing = xgb.predict\_proba(x)[0, 1]

        print(y\_pro\_non\_phishing)

        if (y\_pro\_phishing \* 100 < 60):

            msg = "Trick or Treat?\n Look at this tweet.\n This site is elite!\n"

            flag = 1

        else:

            msg = "Trick or Treat?\n Don't get deceit.\n This site is creep!\n"

            flag = -1

        return render\_template('result.html', msg=msg, url=url, val=flag)

    return render\_template("index.html")

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

**6.3 Develop web site for UI**

For UI to be develop a web page detect web phishing so We develop web page for single text input form and button for submit as Detect.

As click the Detect button that can be redirect to result page.

**Index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<!-- Primary Meta Tags -->

<title>HookPhish</title>

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="title" content="HookPhish">

<meta name="author" content="HookPhish">

<!-- Favicon -->

<link rel="apple-touch-icon" sizes="180x180" href="../static/assets/img/favicon/apple-touch-icon.png">

<link rel="icon" type="image/png" sizes="32x32" href="../static/assets/img/favicon/favicon-32x32.png">

<link rel="icon" type="image/png" sizes="16x16" href="../static/assets/img/favicon/favicon-16x16.png">

<link rel="manifest" href="../static/assets/img/favicon/site.webmanifest">

<!-- Fontawesome -->

<link type="text/css" href="../static/vendor/@fortawesome/fontawesome-free/css/all.min.css" rel="stylesheet">

<!-- Pixel CSS -->

<link type="text/css" href="../static/css/neumorphism.css" rel="stylesheet">

</head>

<body>

    <main>

        <!-- Hero -->

        <div class="section section-header ">

            <div class="container">

                <div class="row justify-content-center">

                    <div class="col-12 col-lg-8 text-center">

                        <h1 class="display-2 mb-4">We are HookPhish</h1>

                           <form action="/" method ="post" class="lead mb-5">

                            <div class="form-group">

                                <div class="input-group">

                                    <div class="input-group-prepend">

                                        <span class="input-group-text"><span class="fa fa-link"></span></span>

                                    </div>

                                    <input class="form-control" id="url" name="url" placeholder="URL (https://www.google.com/)" type="text" aria-label="url-input" required>

                                </div>

                            </div>

                            <button type="submit" class="btn btn-danger"><span class="fa fa-user-secret mr-2"></span>Detect</button>

                        </form>

                </div>

            </div>

        </div>

        <!-- End of Hero section -->

        </div>

    </main>

    <!-- Core -->

<script src="../static/vendor/jquery/dist/jquery.min.js"></script>

<script src="../static/vendor/popper.js/dist/umd/popper.min.js"></script>

<script src="../static/vendor/bootstrap/dist/js/bootstrap.min.js"></script>

<script src="../static/vendor/headroom.js/dist/headroom.min.js"></script>

<!-- Vendor JS -->

<script src="../static/vendor/onscreen/dist/on-screen.umd.min.js"></script>

<script src="../static/vendor/nouislider/distribute/nouislider.min.js"></script>

<script src="../static/vendor/bootstrap-datepicker/dist/js/bootstrap-datepicker.min.js"></script>

<script src="../static/vendor/waypoints/lib/jquery.waypoints.min.js"></script>

<script src="../static/vendor/jarallax/dist/jarallax.min.js"></script>

<script src="../static/vendor/jquery.counterup/jquery.counterup.min.js"></script>

<script src="../static/vendor/jquery-countdown/dist/jquery.countdown.min.js"></script>

<script src="../static/vendor/smooth-scroll/dist/smooth-scroll.polyfills.min.js"></script>

<script src="../static/vendor/prismjs/prism.js"></script>

<script async defer src="https://buttons.github.io/buttons.js"></script>

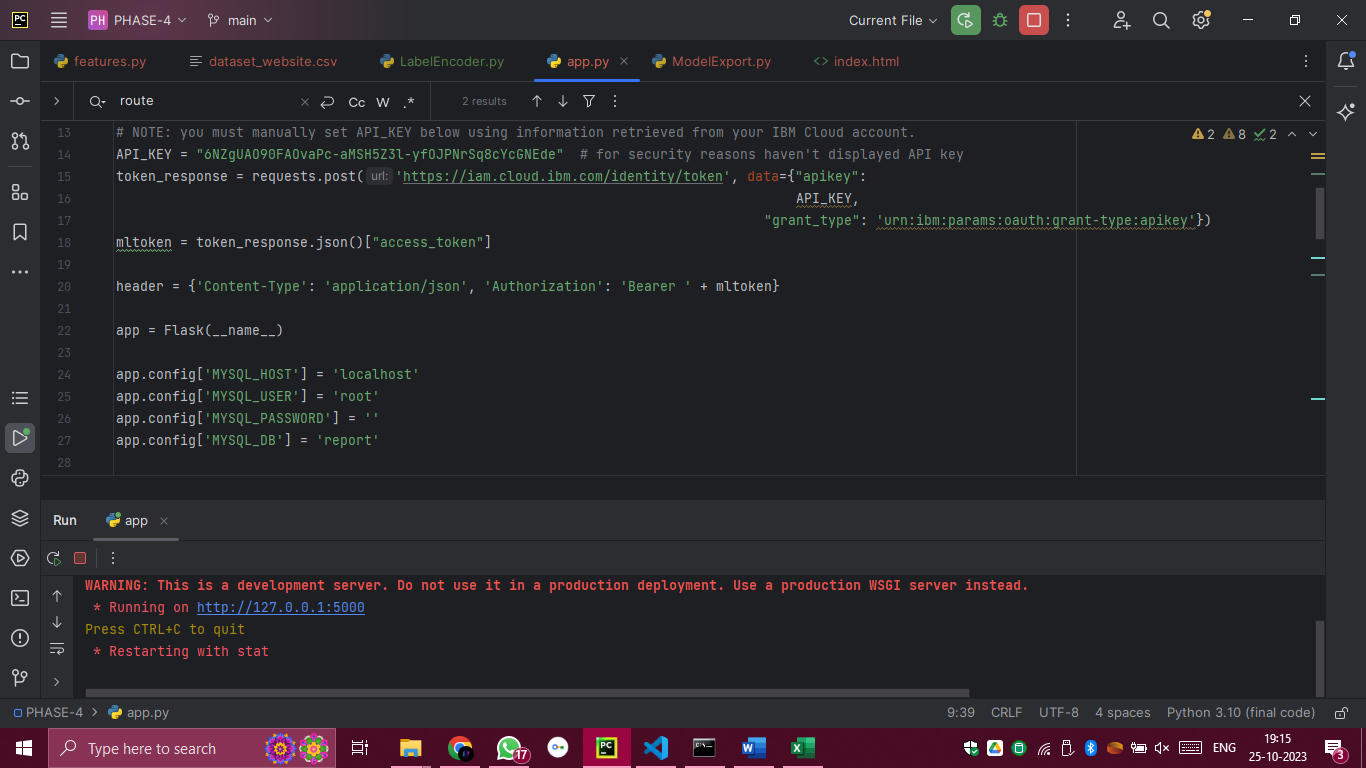
<!-- Neumorphism JS -->

<script src="../static/assets/js/neumorphism.js"></script>

</body>

</html>

Just run Flask application that can running on <http://127.0.0.1:5000/> .

**Fig-21**

**result.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<!-- Primary Meta Tags -->

<title>HookPhish</title>

<!-- Fontawesome -->

<link type="text/css" href="../static/vendor/@fortawesome/fontawesome-free/css/all.min.css" rel="stylesheet">

<!-- Pixel CSS -->

<link type="text/css" href="../static/css/neumorphism.css" rel="stylesheet">

</head>

<body>

    <header class="header-global">

</header>

    <main>

        <!-- Result -->

        <div class="section section-header pb-7">

            <div class="container">

                <div class="row justify-content-center">

                    <div class="col-12 col-lg-8 text-center">

                        <h1 class="display-2 mb-4">Result</h1>

                        <p class="lead">Is <a href= {{ url }} target="\_blank">{{ url }}</a> safe to use?</p>

                        <br>

                        <h3><pre>{{ msg }}</pre></h3>

                        <br>

                        <br>

                        <br>

                        <br>

                        <button type="submit" class="btn btn-primary button1" onclick="window.open('{{url}}')" target="\_blank"><span class="fa fa-arrow-up mr-2"></span>Proceed to this site!</button>

                    </div>

            </div>

        </div>

    </div>

    </main>

    <!-- Core -->

<script src="../static/vendor/jquery/dist/jquery.min.js"></script>

<script src="../static/vendor/popper.js/dist/umd/popper.min.js"></script>

<script src="../static/vendor/bootstrap/dist/js/bootstrap.min.js"></script>

<script src="../static/vendor/headroom.js/dist/headroom.min.js"></script>

<!-- Vendor JS -->

<script src="../static/vendor/onscreen/dist/on-screen.umd.min.js"></script>

<script src="../static/vendor/nouislider/distribute/nouislider.min.js"></script>

<script src="../static/vendor/bootstrap-datepicker/dist/js/bootstrap-datepicker.min.js"></script>

<script src="../static/vendor/waypoints/lib/jquery.waypoints.min.js"></script>

<script src="../static/vendor/jarallax/dist/jarallax.min.js"></script>

<script src="../static/vendor/jquery.counterup/jquery.counterup.min.js"></script>

<script src="../static/vendor/jquery-countdown/dist/jquery.countdown.min.js"></script>

<script src="../static/vendor/smooth-scroll/dist/smooth-scroll.polyfills.min.js"></script>

<script src="../static/vendor/prismjs/prism.js"></script>

<script async defer src="https://buttons.github.io/buttons.js"></script>

<!-- Neumorphism JS -->

<script src="../static/assets/js/neumorphism.js"></script>

<script>

    let x= '{{val}}'

    if(x==1){

        document.getElementById('phishing').style.display = 'none';

    }else{

        document.getElementById('legitimate').style.display = 'none';

    }

</script>

</body>

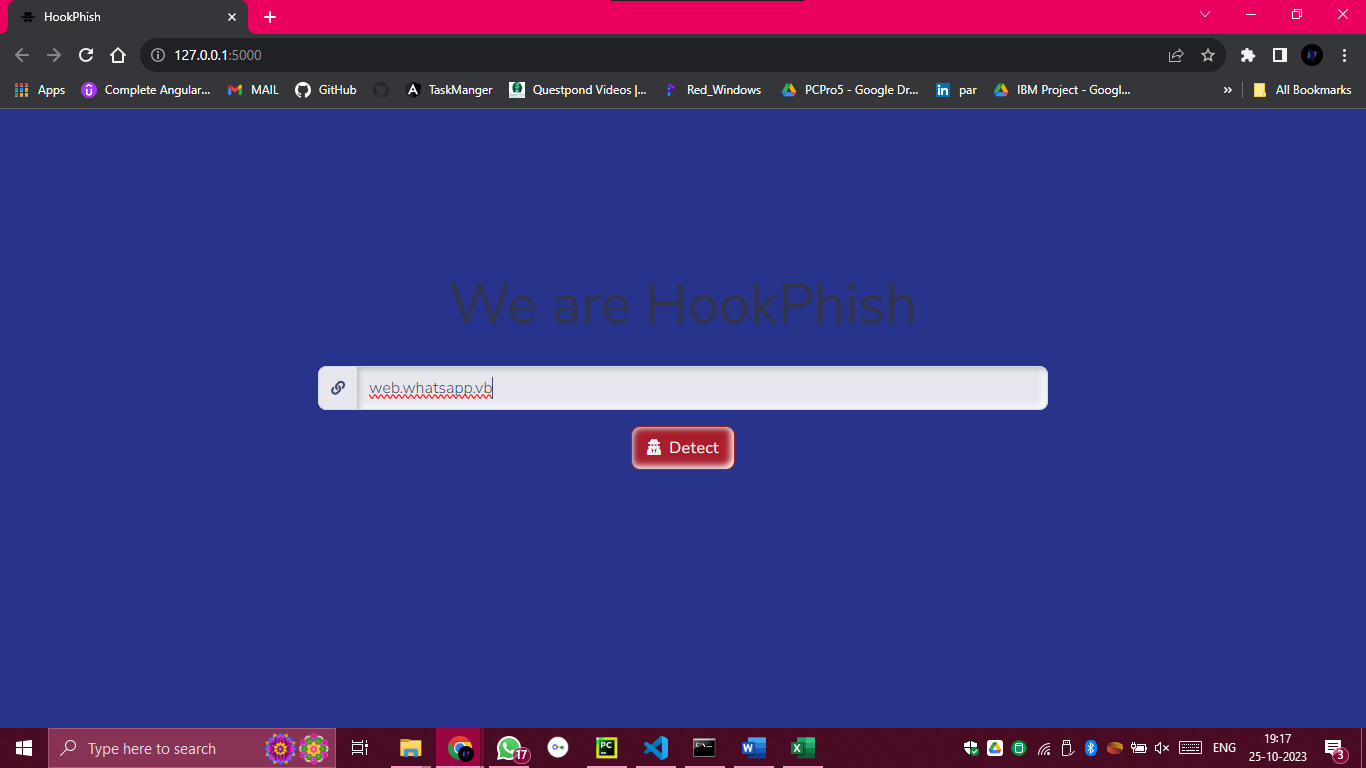
</html>

**CHAPTER 7**

**TESTING & RESULT**

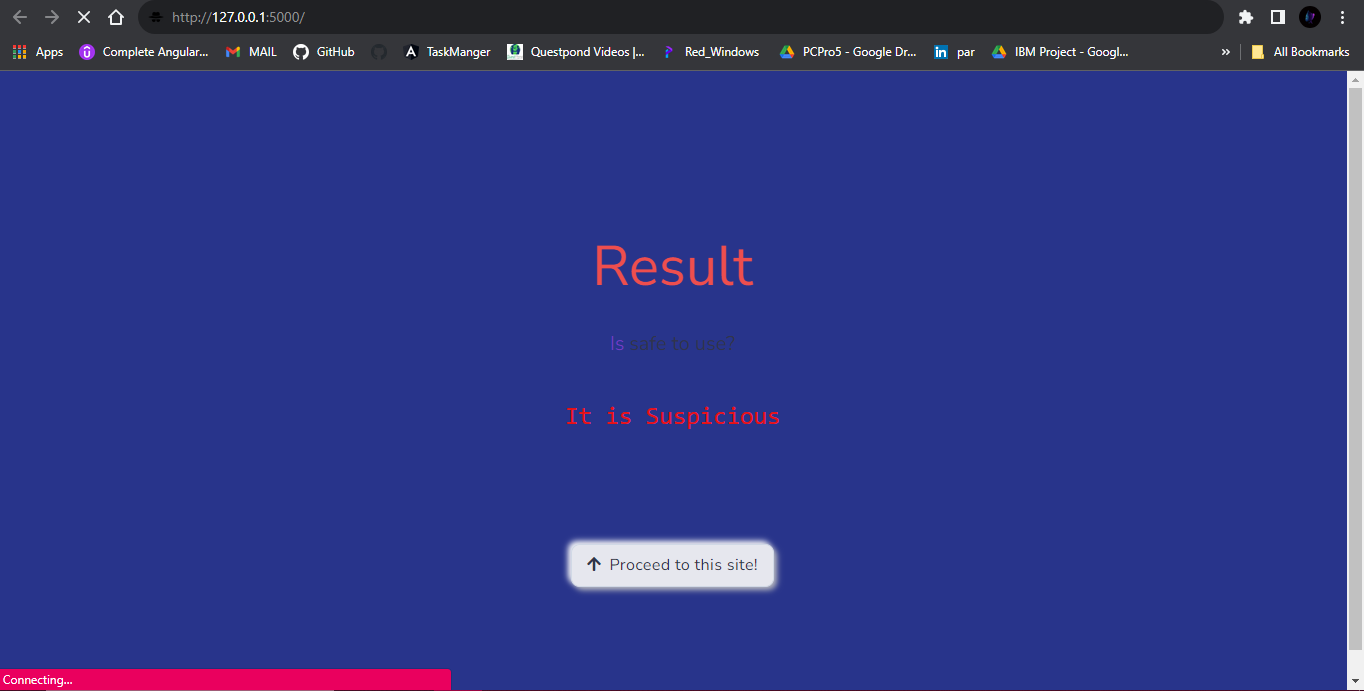
**7.1 Output**

We just give example for [**https://web.whatsapp.vb**](https://web.whatsapp.vb)as whatapp link.



**Fig-22**

A whatapp link is Suspicious.



**Fig-23**

The median efficiency is used to assess each categorization model's effectiveness. The final item will appear in the way it was envisioned. Graphical representations are used to depict information during classification. The percentage of predictions made using the testing dataset is used to gauge accuracy. By dividing the entire number of forecasts even by properly predicted estimates, it is simple to calculate. The difference between actual and anticipated output is used to calculate accuracy

**7.2 Conclusion**

Phishing detection is now an area of great interest among the researchers due to its significance in protecting privacy and providing security. There are many methods to perform phishing detection. Our system aims to enhance the detection method to detect phishing websites using machine learning technology. We achieved a high detection accuracy, and the results show that the classifiers give better performance when we use more data as training data. In future, hybrid technology will be implemented to detect phishing websites more accurately.

**CHAPTER 8**

**APPENDIX**

**8.1 GitHub & Project Demo Link**

GITHUB: <https://github.com/Halishrichard17/Cloud-Application-Development-NM>

**8.12 References**

* <https://www.kaggle.com/datasets/ahmednour/website-phishing-data-set>
* <https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=services-watson-machine-learning>