AMBIENT SYSTEM

Phishing + entrainement de modèles

Membres du groupe :

-CHANA Housna

-JOUFFRAULT Guillaume

-HASSANE Abdoul-Aziz

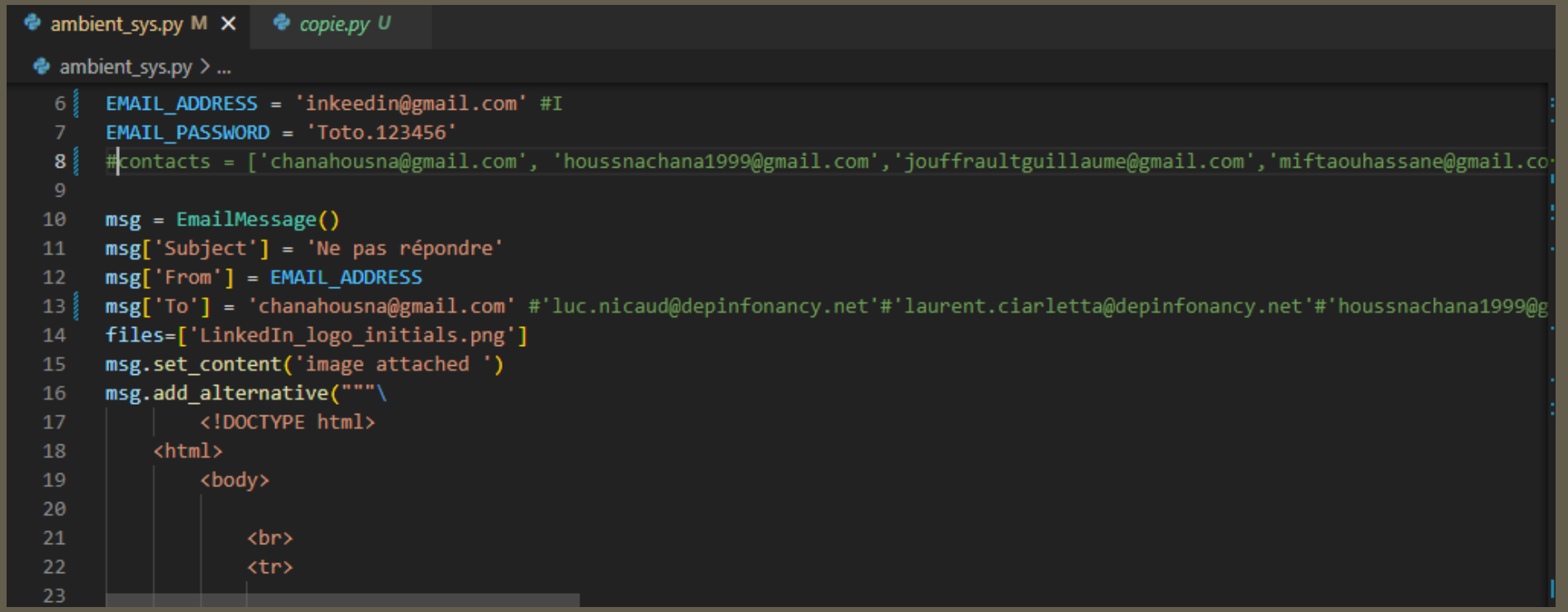
-NICAUD Luc

2021/2022

1) L’ENVOI DU MAIL

- Il s’agit d’envoyer un mail qui contient le lien du site du phishing, qui dans notre cas, s’agit d’une page LinkedIn. Nous avons choisi d’écrire un script python qui automatisera cette tache.
- Pour ceci voici ce qu’il nous faut :

1. L’adresse mail et le mot de passe de l’expéditeur (du coup , nous avons crée un compte Gmail sous le nom Inkeedin)
2. L’adresse ou les adresses du/des destinataire(s)
3. le message ou le contenu de l’email codé en html
4. le service smtp (Simple Message Transfert Protocole) : c’est un protocole utilisé pour transférer les messages électroniques sur les réseaux.



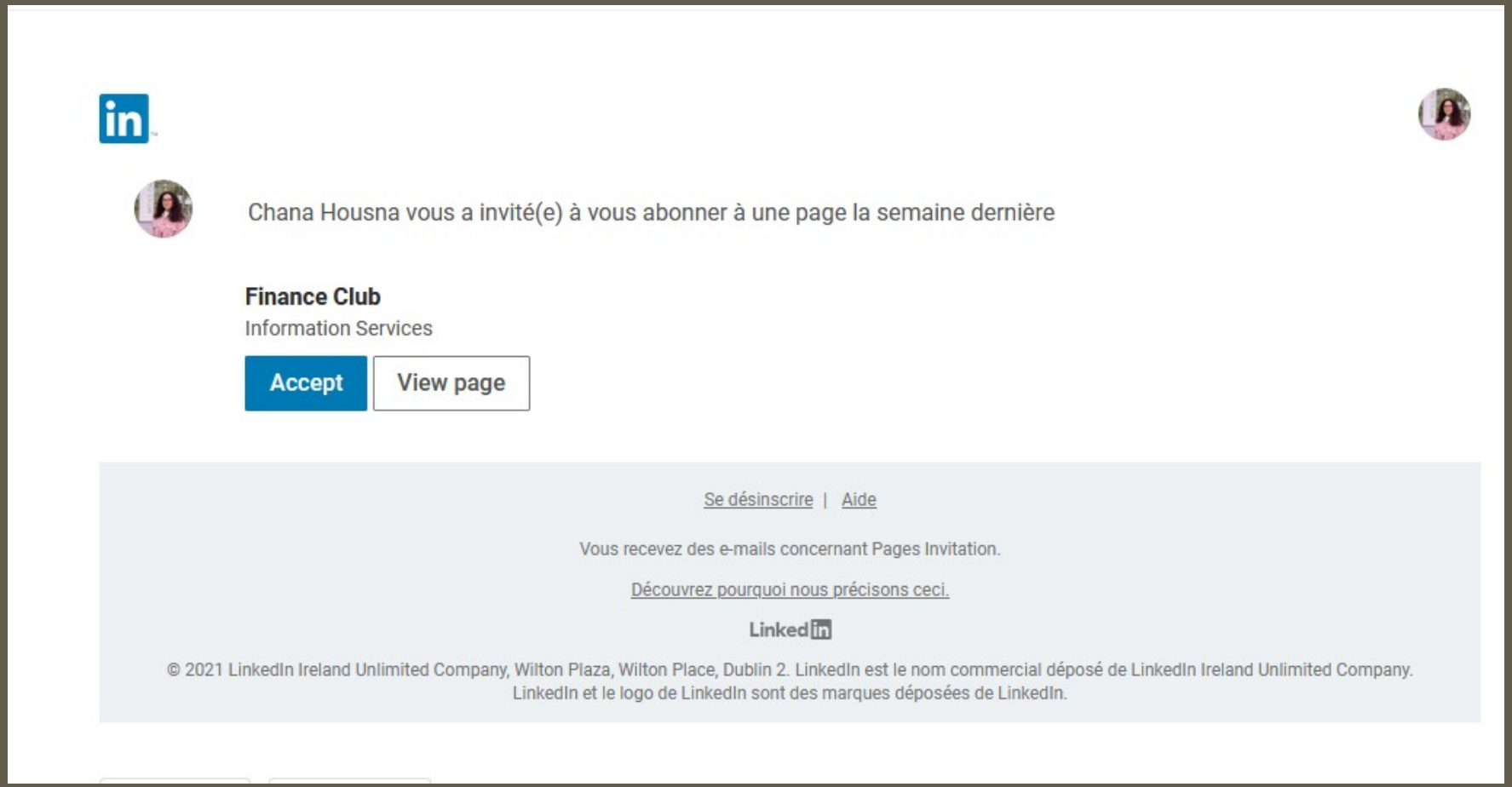
Code python (1)

ambient_sys.py M copie.py U

ambient_sys.py > ...

```
171
172
173  with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp: #or port 465
174      try :
175          print('loging')
176          smtp.login(EMAIL_ADDRESS, EMAIL_PASSWORD)
177          print('sending')
178          smtp.send_message(msg)
179          print('done')
180
181      except smtplib.SMTPResponseException as e:
182          error_code = e.smtp_code
183          error_message = e.smtp_error
```

Code python (2)



La forme de l'email envoyé

2) SITE DE PHISHING

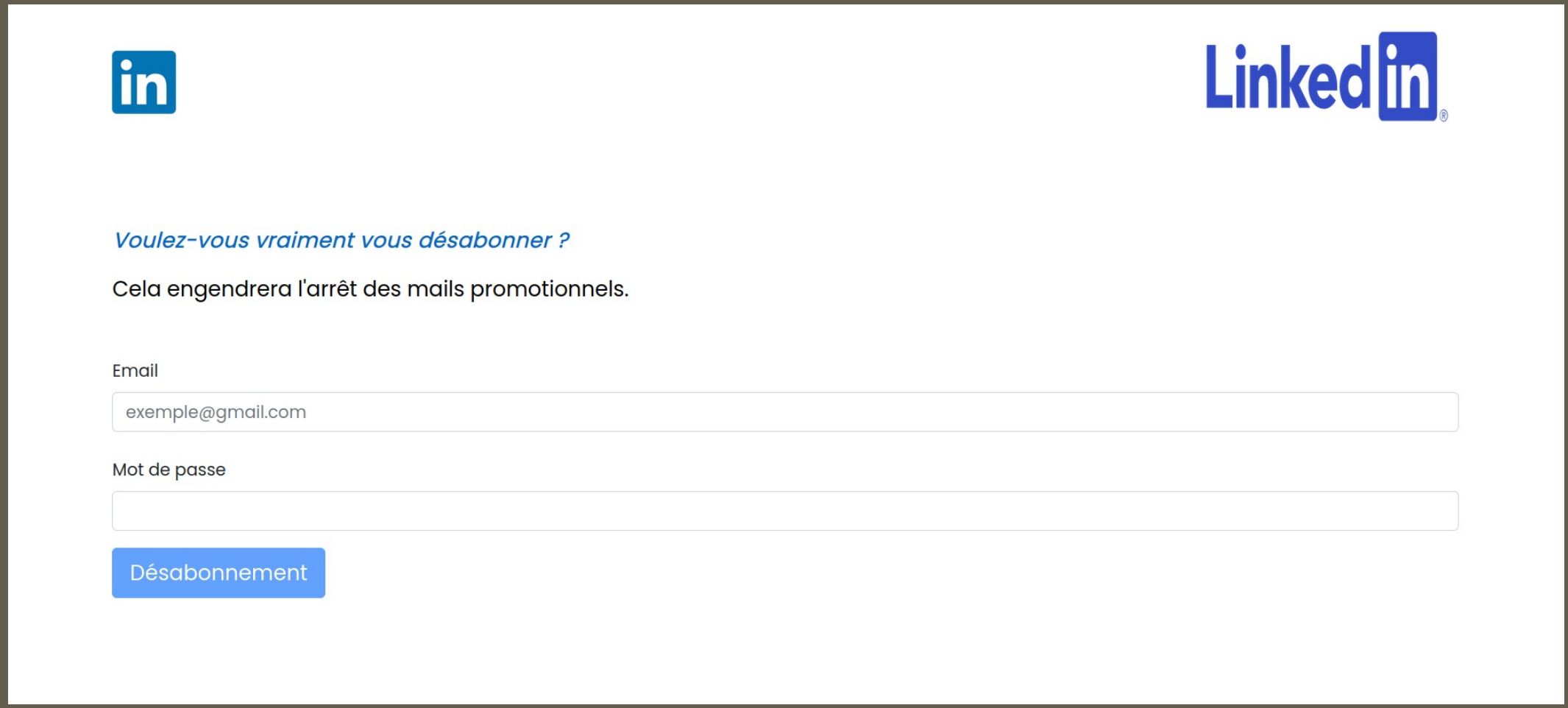
Lien github de l’application : https://github.com/Guillaume-Jouffrault/unsubscribe

Lien du site de phising : logistilink.xyz

Lorsqu’il clique sur le bouton de désabonnement, l’utilisateur arrive sur un site web.

L’url contient en paramètre son mail, ce qui permet de mettre dans la base de donnée (firebase) le mail de chaque personne ayant cliquée, mais aussi de récupérer son ip et sa localisation (pays, ville).

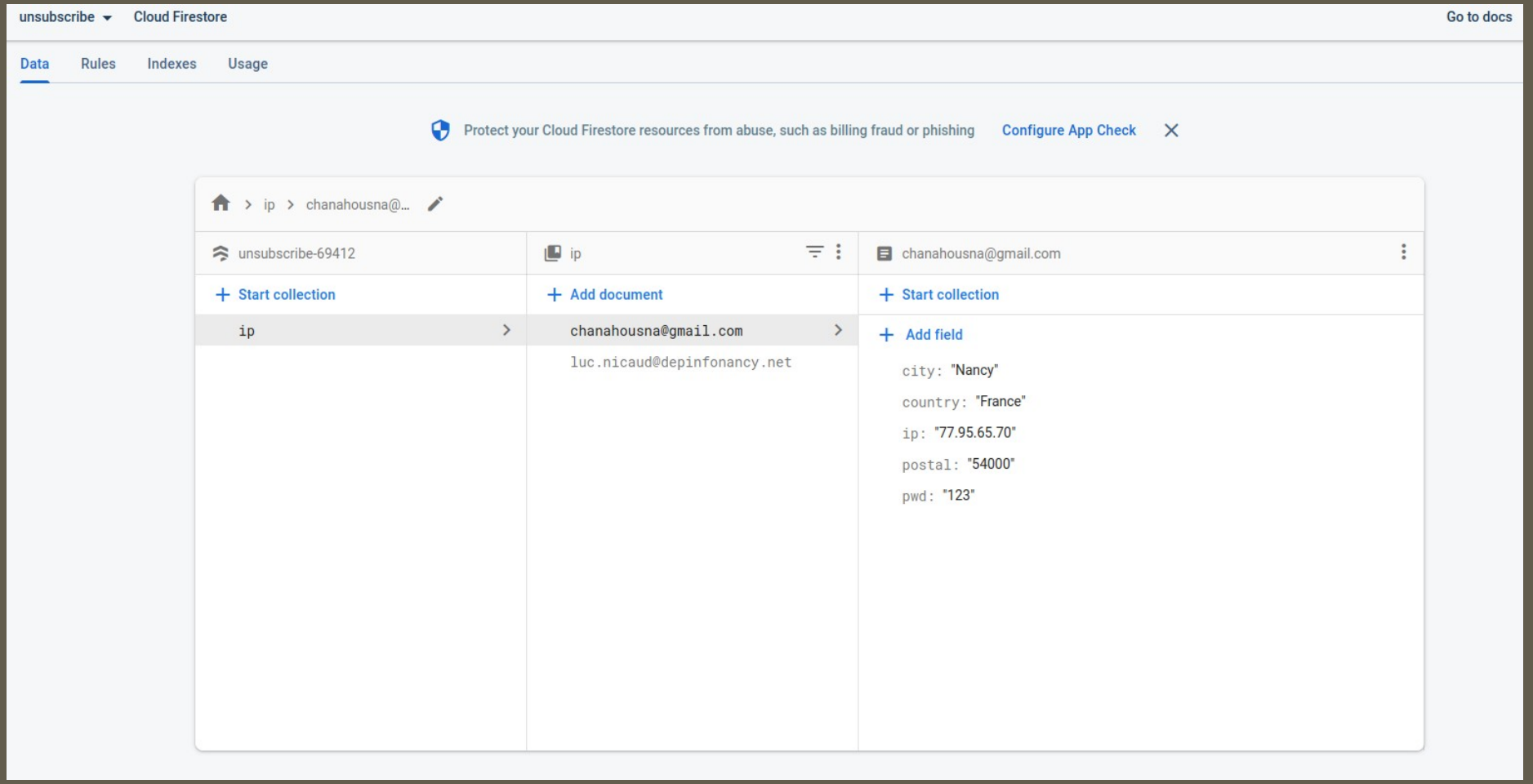
L’utilisateur peut rentrer ses logs, il croira s’être désabonné alors que ses logs sont aussi stockés sur la bdd.



Page d’acceuil du site de phishing

Désabonnement effectué.

Page succédant une récupération des logs de l'utilisateur



Base de donnée firebase obtenue lors du phising
(le pwd n'apparait que si les logs sont rentrés sur le site)

3)PARTIE ENTRAINEMENT :

- Dans cette partie, Housna a décidé de tester si l’URL de notre site sera détecté comme du phishing ou non par des modèles d’apprentissage automatique.
- Voici les étapes suivies :

- Etape 1 :

1. Récupérer une dataset contenant presque 700,000 URLs
2. Nettoyer la dataset (on s'intéresse uniquement à deux types phishing ou site bénin)
3. Supprimer les « WWW » des URLs de la dataset
4. On catégorise les classes : 1 pour phishing et 0 pour l'URL bénin
5. Faire de la « Feature extraction » :

   5.1) calculer la longueur des URLs

   5.2) Montrer si l'URL contient un https ou un http (1 pour https et 0 pour http)

   5.3)Extraire le tld (top level domain) de chaque URL

   5.3) calculer le nombre de caractères numériques pour chaque URL (combien de @, combien de « ,», …)

   5.4) calculer le nombre de lettres pour chaque URL

   5.5) vérifier si l'URL utilise un service de shortening (bit…)

   5.6) extraire le type d'IP (IPv4 ou IPv6 grâce aux regex)

- Etape 2 :

1. Diviser la dataset en deux datasets : train et test :

{ X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2);

avec X représente toutes les colonnes de la dataset à part la colonne "category" et y c'est la colonne "category "; par category on désigne soit 1: phishing ou 0: bénin }

2. Entrainer quelques modèles
3. Appliquer ces modèles sur la dataset test
4. Faire de la détection sur un nouveau URL , dans ce cas ca serait notre « site »
5. Conclure si notre site a été détecté comme du phishing ou non (et par quels modèles)

- Après avoir analyser les résultats , je prends les 3 meilleurs algorithmes qui ont donné le meilleur score d’accuracy et je les utilise sur notre lien
- En effet, j’ai crée une fonction qui fait à peu près toutes les étapes mentionnées auparavant :

```
def URL_Converter(urls):
    data= pd.DataFrame()
    data['url'] = pd.Series(urls)


    data['url_len'] = data['url'].apply(lambda x: len(str(x)))
    data['domain'] = data['url'].apply(lambda i: process_tld(i))
    feature = ['@','?','-','=','.','#','%','+','$','!','*',',','//']
    for a in feature:
        data[a] = data['url'].apply(lambda i: i.count(a))
    data['abnormal_url'] = data['url'].apply(lambda i: abnormal_url(i))
    data['https'] = data['url'].apply(lambda i: httpSecure(i))
    data['digits']= data['url'].apply(lambda i: digit_count(i))
    data['letters']= data['url'].apply(lambda i: letter_count(i))
    data['Shortining_Service'] = data['url'].apply(lambda x: Shortining_Service(x))
    data['having_ip_address'] = data['url'].apply(lambda i: having_ip_address(i))
    print(data.columns)
    X = data.drop(['url','domain'],axis=1)

    return X
```

- Puis j’applique cette fonction sur les nouveaux liens(le 1er et le 2ème liens de la liste urls sont les liens de notre site) ,
- je réentraîne puis je détecte :

```
urls=['https://logistilink.xyz',
      'https://iridescent-croquembouche-4b8508.netlify.app/',
      'apple-search.info',
      'www.itcarezone.com/xlrmp/files/anna.exe',
      'https://towardsdatascience.com/',
     ]
```

```
test_data= URL_Converter(urls)
```

```
Index(['url', 'url_len', 'domain', '@', '?', '-', '=', '.', '#', '%', '+', '$',
       '!', '*', ',', '//', 'abnormal_url', 'https', 'digits', 'letters',
       'Shortining_Service', 'having_ip_address'],
      dtype='object')
```

+ Code + Markdown

```
models = [DecisionTreeClassifier,RandomForestClassifier,KNeighborsClassifier]
for m in models:
    print('############################################')
    print('######-Model =>\033[07m {} \033[0m'.format(m))
    model_ = m()
    model_.fit(X_train, y_train)
    pred = model_.predict(test_data)
    print(pred)
```

RÉSULTATS :

```
models = [DecisionTreeClassifier,RandomForestClassifier,KNeighborsClassifier]
for m in models:
    print('##############################################')
    print('######-Model =>\033[07m {} \033[0m'.format(m))
    model_ = m()
    model_.fit(X_train, y_train)
    pred = model_.predict(test_data)
    print(pred)
```

```
##############################################
######-Model => <class 'sklearn.tree._classes.DecisionTreeClassifier'>
[1 1 1 1 1]
##############################################
######-Model => <class 'sklearn.ensemble._forest.RandomForestClassifier'>
[1 1 1 1 1]
##############################################
######-Model => <class 'sklearn.neighbors._classification.KNeighborsClassifier'>
[1 0 1 0 1]
```

On remarque que les 2 premiers modèles détectent les deux liens comme du phishing; alors que le troisième reconnait qu'un seul.

Toutefois, il faut revoir les features que j'ai choisi (à l'aide de ce que j'ai trouvé sur quelques notebooks) car , par exemple, le dernier url s'agit d'un site bénin pourtant il a été détecté comme du phishing ?!

Conclusion : c'était un bon exemple pour s'exercer sur un simple entrainement de modèles, et de combiner entre deux sujets qui paraissent indépendants.
Néanmoins, afin d'avoir des résultats pertinents il faut chercher d'autres caractéristiques, parce que