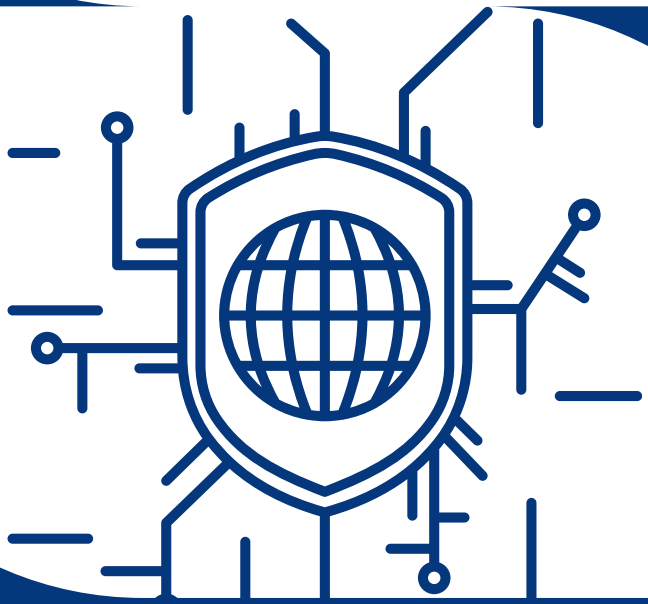
Rapport de Projet

MODULE
CYBERSECURITE

PDE
CYBERSECURITY



RÉDIGÉ PAR KOUEMOU SAH JEAN EMAC

1. Introduction

Ce rapport détaille le développement et la sécurisation d'une application web de détection de phishing d'email. L'application utilise Python (Flask) pour le backend et JavaScript pour le frontend, et elle intègre un modèle de machine learning pré-entraîné pour classifier les emails comme "Safe Email" ou "Phishing Email". Le projet met un accent particulier sur la sécurisation de l'API et l'amélioration de l'expérience utilisateur.

2. Objectifs du Projet

L'objectif principal du projet est de créer une API Flask robuste, incorporant une protection d'authentification complète et des mécanismes de chiffrement de données. L'application exploite un dataset de cybersécurité pour entraîner un modèle de machine learning performant, sauvegarder ce modèle pour une utilisation ultérieure, et l'intégrer de manière transparente dans l'API Flask.

3. Étapes du Projet

Le projet a été réalisé en plusieurs étapes clés, chacune contribuant à la réalisation de l'objectif global :

- **Prérequis :**
    - Création d'un compte ngrok et obtention d'une clé d'authentification pour exposer l'application localement.
    - Installation des bibliothèques Python nécessaires (e.g., Flask, scikit-learn, pandas, ngrok).

- **Partie I : Analyse des Données**

  - Téléchargement du dataset pertinent depuis Kaggle, une plateforme reconnue pour ses ressources de données.

  - Analyse Exploratoire des Données (EDA) : Cette phase cruciale impliquait l'examen approfondi du dataset pour comprendre sa structure, ses types de données, les valeurs manquantes, et les distributions statistiques.

  - L'EDA a permis d'identifier les caractéristiques importantes pour la détection de phishing, telles que la présence de certains mots-clés, la longueur des e-mails.

  - Analyse Graphique des Données :La visualisation des données a joué un rôle clé dans l'identification des tendances.

  - Des histogrammes, des diagrammes à barres, ont été utilisés pour représenter la distribution des variables. Par exemple, la distribution des e-mails phishing vs. Safe Email a été visualisée pour comprendre le déséquilibre des classes.

- **Partie II : Modélisation**

  Encodage des Variables Nécessaires :

  - La variable catégorielles (Email type) a été encodée en variable numérique binaire (0 et 1) pour être compatibles avec les algorithmes de machine learning.

- **Vectorisation des Données :**
  - Le texte des e-mails a été transformé en vecteurs numériques, représentant les caractéristiques textuelles.
  - La technique TF-IDF (Term Frequency-Inverse Document Frequency) a été utilisée pour pondérer l'importance des mots dans chaque e-mail.
- **Séparation du Dataset :**
  - Le dataset a été divisé en données d'entraînement (pour entraîner le modèle) et données de test (pour évaluer ses performances).
  - Une séparation typique est 80% pour l'entraînement et 20% pour le test.
- **Sélection du Modèle :**
  - Le modèle de Régression Logistique a été choisi en raison de son efficacité pour les problèmes de classification binaire (phishing ou non-phishing).
  - D'autres modèles (comme les arbres de décision, les forêts aléatoires, ou les SVM) auraient pu être considérés et comparés.
- **Entraînement du Modèle :**
  - L'algorithme de Régression Logistique a été entraîné sur les données d'entraînement, ajustant ses paramètres pour minimiser l'erreur de prédiction.

- **Évaluation du Modèle :**

Les performances du modèle ont été évaluées sur les données de test en utilisant des métriques telles que :

  - Précision : Proportion de prédictions correctes.
  - Rappel : Capacité du modèle à identifier tous les e-mails phishing.
  - Score F1 : Moyenne harmonique de la précision et du rappel.
  - Matrice de Confusion : Visualisation des vrais positifs, vrais négatifs, faux positifs, et faux négatifs.
  - Courbe ROC (Receiver Operating Characteristic): Évaluation de la capacité de discrimination du modèle.

	precision	recall	f1-score	support
0	0.98	0.96	0.97	2264
1	0.95	0.97	0.96	1466
accuracy			0.97	3730
macro avg	0.96	0.97	0.97	3730
weighted avg	0.97	0.97	0.97	3730

- **Enregistrement et Exportation du Modèle**

  - Le modèle entraîné (et le vectoriseur) ont été sauvegardés dans des fichiers (par exemple, .pkl) pour être chargés et utilisés par l'application Flask.

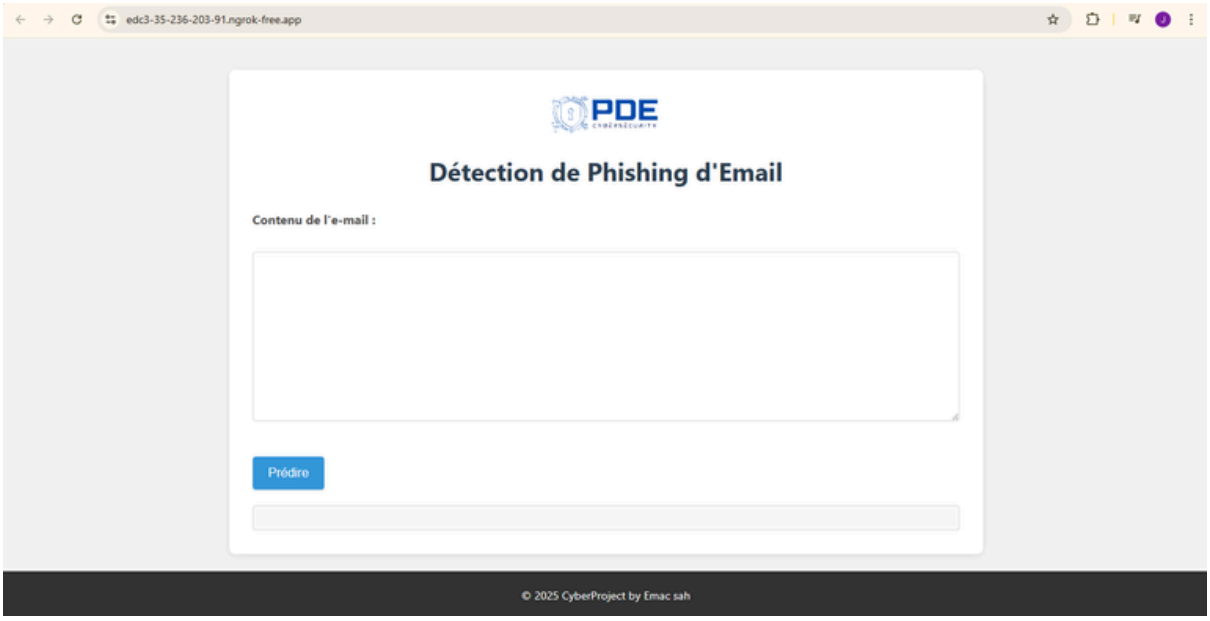
- **Partie III : Application Flask**

  - **Copie du Auth Token depuis ngrok :** Le token d'authentification ngrok a été récupéré pour permettre à l'application Flask de fonctionner avec le service de tunneling.

- **Charger le token d'authentification dans l'application :** Le token a été intégré dans le code Flask pour établir la connexion avec ngrok.

- **Créer l'interface d'affichage graphique (index.html) :** Une interface HTML simple a été créée pour permettre aux utilisateurs de saisir le contenu des e-mails et d'afficher les résultats de la prédiction.

- **Exécuter et tester l'application :** L'application Flask a été lancée, démarrant un serveur web local. L'application était accessible via une URL générée par ngrok. L'interface utilisateur a été testée pour s'assurer qu'elle fonctionnait correctement.



Modèle et vectoriseur chargés avec succès.
Public URL: NgrokTunnel: "https://4855-35-236-203-91.ngrok-free.app" -> "http://localhost:5000"
** Serving Flask app '__main__'*
** Debug mode: off*
INFO:werkzeug:WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
** Running on http://127.0.0.1:5000*
INFO:werkzeug:Press CTRL+C to quit

Partie IV : Cryptographie (Sécurité)

- **Génération de la clé de cryptage (Fernet) :** Une clé symétrique a été générée à l'aide de la bibliothèque Fernet pour chiffrer et déchiffrer les données sensibles.

- **Enregistrement de la clé :** La clé Fernet a été stockée de manière sécurisée (dans un fichier protégé).

- **Authentification de l'API (Protection de l'accès à l'endpoint /predict) :** Un mécanisme d'authentification (basé sur une clé d'API) a été mis en place pour contrôler l'accès à l'API de prédiction.

4. Développement de l'Application Flask (Développé)

- **Structure de l'Application :** Le projet est organisé avec les dossiers suivants :

  - /content/drive/MyDrive/cybersecurite/Cyberproject/ : Racine du projet.

  - **app.py (ou cryp.py) :** Le fichier principal de l'application Flask. Il gère les routes web, le chargement du modèle de machine learning, la logique de prédiction et l'authentification de l'API.
  - **models/ :** Contient les fichiers du modèle ML (phishing_model.pkl) et du vectoriseur (tfidf_vectorizer.pkl).
  - **templates/ :** Contient les templates HTML (index.html).
  - **static/ :** Contient les fichiers statiques (CSS, images).
  - **keys/ :** Stocke les clés de chiffrement (Fernet).

- **Fonctionnalités de l'API :**

  - La fonction **load_model_and_vectorizer()** charge les fichiers du modèle et du vectoriseur au démarrage de l'application.
  - La route **/** sert la page d'accueil (index.html).
  - La route **/predict** : Récupère le contenu de l'e-mail depuis une requête POST ensuite vectorise le texte de l'e-mail. puis utilise le modèle ML pour la prédiction et enfin retourne la prédiction en JSON.
  - L'application inclut une vérification de la langue de l'email (doit être en anglais).
  - **Déploiement :** L'application Flask est déployée en utilisant ngrok pour créer un tunnel public et sécurisé (HTTPS).

Sécurité de l'Application :

- **Authentification de l'API :**
    - Une clé d'API est générée côté serveur et stockée dans une variable d'environnement (API_KEY).
    - Le client (la page web) doit inclure cette clé dans l'en-tête X-API-Key de chaque requête à /predict pour être autorisé à utiliser l'API.
    - Un décorateur Flask (@require_auth) a été mis en place pour vérifier la présence et la validité de la clé d'API dans les requêtes entrantes.
    - Pour masquer la clé d'API côté client, un proxy côté serveur (/submit_email) a été introduit. Le client envoie les données de l'e-mail à /submit_email, et le serveur ajoute la clé d'API avant de faire la requête à /predict.

- **Chiffrement (Implémentation Partielle) :**
    - Une clé Fernet est générée et stockée pour potentiellement chiffrer des données sensibles.
    - La clé Fernet est chargée depuis une variable d'environnement (FERNET_KEY) au démarrage de l'application Flask.
    - L'application Flask est configurée pour utiliser cette clé Fernet pour des opérations de chiffrement si nécessaire.

- **Sécurisation des Communications :**
    - Les communications entre le navigateur de l'utilisateur et le serveur Flask sont sécurisées en utilisant HTTPS via ngrok.

Analyse des Résultats

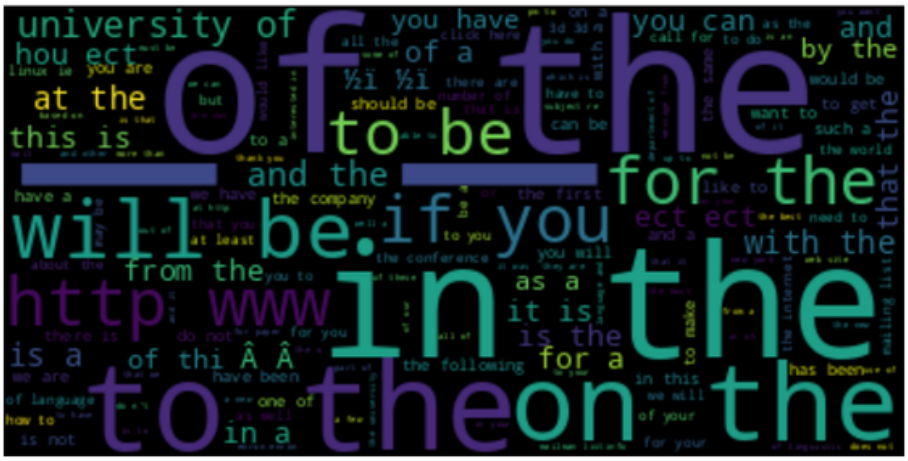
- **Partie I : Analyse des Données :**
    - L'EDA a fourni des informations cruciales sur la nature du dataset, permettant de comprendre les caractéristiques les plus discriminantes entre les e-mails phishing et les e-mails légitimes.
    - L'analyse graphique a aidé à visualiser ces différences, guidant les choix de prétraitement des données et de sélection de modèle. Par exemple, si l'analyse a révélé que certains mots-clés sont fortement corrélés avec les e-mails phishing, ces mots-clés pourraient être pondérés plus fortement lors de la vectorisation
- **Partie II : Modélisation**
    - Le choix de la Régression Logistique comme modèle de classification était justifié par sa capacité à fournir de bonnes performances pour les problèmes de classification binaire et son interprétabilité relative.
    - Les métriques d'évaluation ont permis de quantifier les performances du modèle. Le rappel élevé est crucial dans la détection de phishing pour minimiser les faux négatifs (e-mails phishing classés comme sûrs).
    - La précision élevée indique que le modèle fait peu d'erreurs en général.

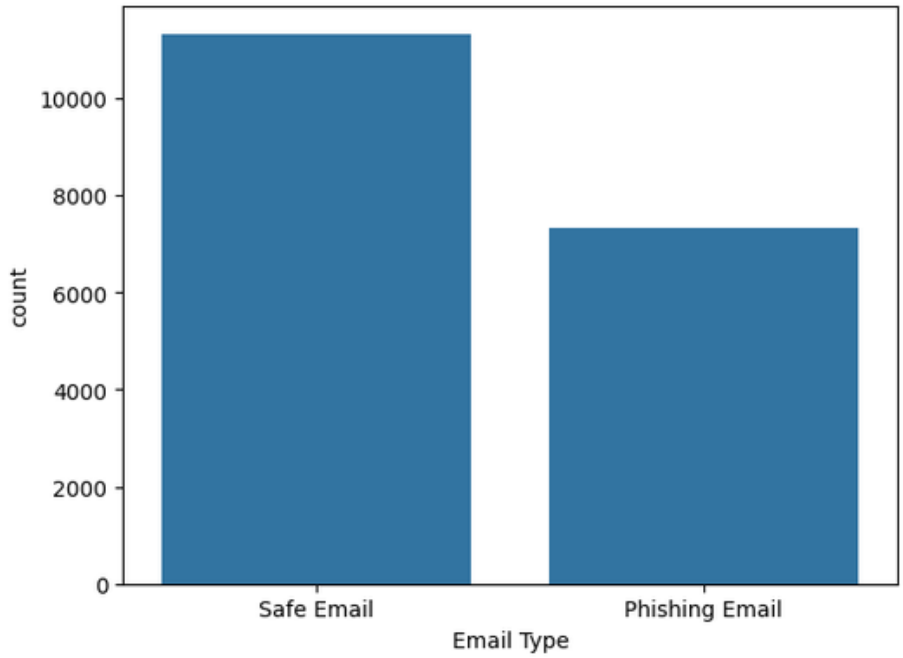
Analyse des Résultats

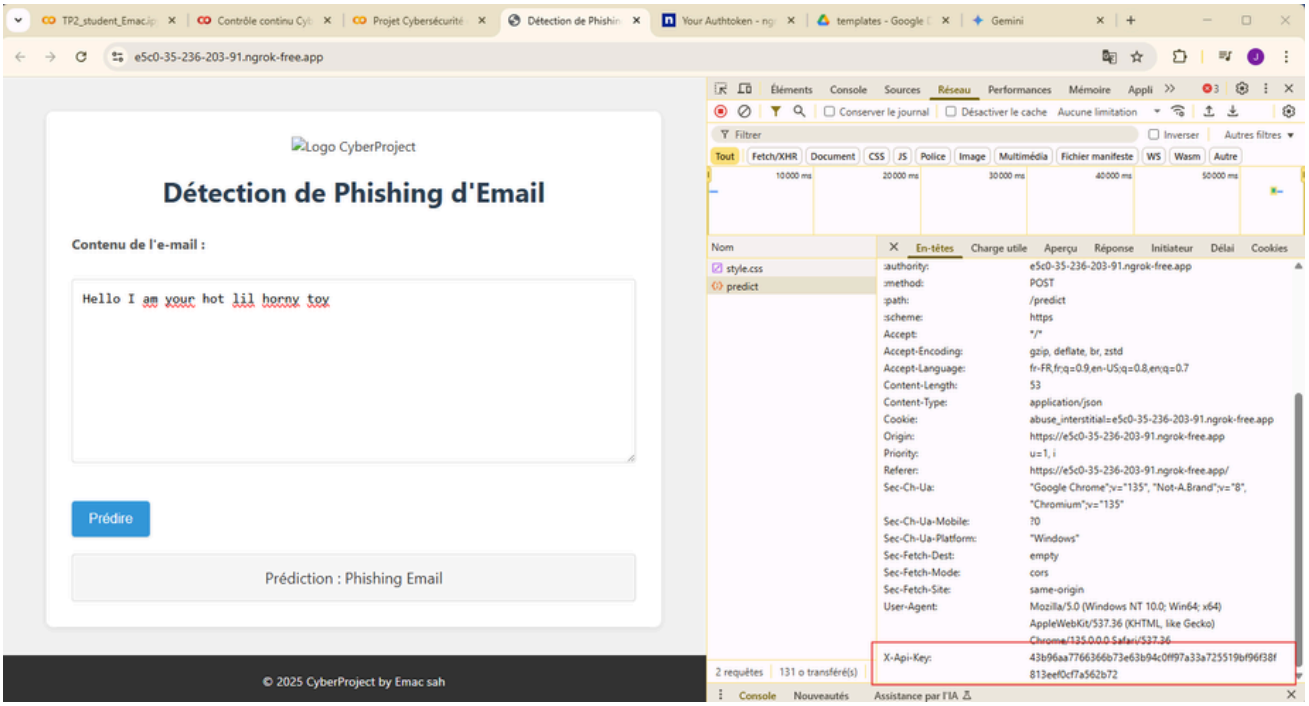
- **Partie III : Application Flask**
    - L'application Flask a fourni une interface web fonctionnelle pour interagir avec le modèle de détection de phishing.
    - L'utilisation de ngrok a permis de déployer rapidement l'application pour des tests.
- **Partie II : Partie IV : Cryptographie (Sécurité)**
    - L'authentification de l'API a ajouté une couche de sécurité, empêchant les accès non autorisés à l'API de prédiction.
    - Le proxy côté serveur a amélioré la sécurité en masquant la clé d'API côté client.
    - L'intégration de Fernet a préparé l'application pour le chiffrement de données sensibles à l'avenir.

Sécurité Avancée (Discussions) : Pour renforcer davantage la sécurité, les méthodes d'authentification basées sur des sessions ou des tokens (JWT) sont recommandées à la place de la simple transmission de la clé d'API. Ces méthodes offrent une meilleure protection contre les attaques potentielles.

Conclusion : L'application de détection de phishing d'email a été développée avec succès, intégrant des fonctionnalités de sécurité essentielles et une interface utilisateur fonctionnelle. Les prochaines étapes pourraient inclure l'amélioration de la sécurité via des sessions ou des tokens, et d'autres optimisations de l'interface utilisateur et des performances du modèle.


university of
you have
you can
and
hou ect
of a
by the
at the
this is
of the
to be
and the
for the
will be
if you
ect ect
with the
from the
http www
in the
as a
it is
is the
for a
is a
of thi Â Â
to the
on the
in a


10000
8000
6000
4000
2000
0
count
Safe Email
Phishing Email
Email Type


e5c0-35-236-203-91.ngrok-free.app
Logo CyberProject
Détection de Phishing d'Email
Contenu de l'e-mail :
Hello I am your hot lil horny toy
Prédire
Prédiction : Phishing Email
© 2025 CyberProject by Emac sah
X-Api-Key:
43b96aa7766366b73e63b94c0ff97a33a725519bf96f38f813eef0cf7a562b72