

PROJET BIG DATA

Thème :   
  
Analyse des risques de la cybersécurité.

Membre du Projet :

* Dommoe Sifoua Elsa
* Dogac Burak
* Eone Abina Gilbert Freddy
* Essomba Essomba Kevin Emmanuel

Encadrant : **M Sala-Eddine Tbahriti**

Table des matières

1. Introduction
2. Mise en place de l’architecture de travail
3. Collecte, traitement et analyse des données pour les différents modèles
4. Test et déploiement du modèle de lien via une interface
   1. Introduction

La cybersécurité est un enjeu majeur à l'ère numérique, où les activités en ligne sont de plus en plus courantes. Les risques associés aux cyberattaques sont nombreux et peuvent avoir des conséquences graves pour les entreprises et les particuliers. Les cybercriminels utilisent des techniques sophistiquées pour infiltrer les réseaux et les systèmes informatiques, voler des données sensibles, extorquer de l'argent ou perturber les activités.

Selon McAfee , le coût annuel probable de la cybercriminalité pour l'économie mondiale est de plus de 400 milliards de dollars. Le coût des cybercrimes comprend les dommages causés aux performances des entreprises et aux  économies nationales par les cyberattaques, ainsi que les effets du vol d'informations. Comme indiqué par McAfee, en 2020.

Parmi ces menaces, on note particulièrement le Phishing. Ce dernier   est une menace sérieuse pour la sécurité en ligne, et peut exposer les utilisateurs à de nombreux risques. Cette technique permet aux cybercriminels de voler des informations personnelles et financières sensibles, provoquer des pertes financières et endommager la réputation des entreprises. Les risques d'exposition au phishing augmentent si l'utilisateur ne prend pas de mesures de sécurité appropriées parmi lesquelles: cliquer sur des liens suspects et ouvrir des courriels de phishing ou télécharger des fichiers malveillants. Les utilisateurs doivent être conscients de ces risques et prendre des mesures de prévention pour se protéger.

Pour faire face à de telles attaques, nous devons être capables de les identifier et de les caractériser. Un facteur de complication dans ce cas est l'énorme augmentation du volume de données disponibles qui s'est produite ces dernières années. Près de 90 % des données dans le monde aujourd'hui ont été créées au cours des deux dernières années seulement, et chaque jour, environ 2,5 quintillions d'octets de données sont créés.

Dans le cadre notre projet, nous avons mis en place une interface graphique et application pour détecter respectivement des URLs vers des sites Phishing. Notre application collecte des données à partir de honeypots (pots de miel) situés dans différents pays et continents, et stockés dans HDFS. La principale contribution de l'application est sa capacité à identifier les e-mails de phishing dans un ensemble de spams.  
En utilisant la régression logistique, nous avons réalisé deux modèles entraînés sur plusieurs types datasets :

* Un dataset combiné, constitué de 2 datasets “Phishing Site URLs” et “Phishing website Data” pour les URLs
* Un dataset “Enron-spam dataset” de spams répertoriés pour les mails.

* 1. Mise en place de l'environnement de travail

Afin d’effectuer au mieux notre projet Big Data, nous avons créé une image Hadoop HDFS, comportant plusieurs services. Le fichier YML se presente comme suit

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Le code ci-dessus s’agit de notre fichier docker-compose.yml. Il définit une série de services dans un environnement de cluster Hadoop.

* Le premier service est “namenode”, qui définit un nœud de nom Hadoop. Il utilise l'image Docker bde2020/hadoop-namenode:2.0.0-hadoop3.2.1-java8 et est nommé "namenode". Il expose les ports 9870 et 9000, qui sont utilisés par le service pour fournir des informations sur le cluster et permettre aux clients d'accéder aux fichiers stockés sur HDFS. Il utilise un volume nommé hadoop\_namenode pour stocker les données du nœud de nom.
* Le deuxième service est “datanode”, qui définit un nœud de données Hadoop. Il utilise l'image Docker bde2020/hadoop-datanode:2.0.0-hadoop3.2.1-java8 et est nommé "datanode". Il utilise un volume nommé hadoop\_datanode pour stocker les données du nœud de données et utilise une variable d'environnement SERVICE\_PRECONDITION pour indiquer que le service a besoin d'un nœud de nom sur le port 9870.
* Le troisième service est “resourcemanager”, qui définit un gestionnaire de ressources Hadoop. Il utilise l'image Docker bde2020/hadoop-resourcemanager:2.0.0-hadoop3.2.1-java8 et est nommé "resourcemanager". Il utilise une variable d'environnement SERVICE\_PRECONDITION pour indiquer qu'il a besoin du nœud de nom et du nœud de données pour fonctionner.
* Le quatrième service est “nodemanager1”, qui définit un gestionnaire de nœud Hadoop. Il utilise l'image Docker bde2020/hadoop-nodemanager:2.0.0-hadoop3.2.1-java8 et est nommé "nodemanager". Il utilise une variable d'environnement SERVICE\_PRECONDITION pour indiquer qu'il a besoin du nœud de nom, du nœud de données et du gestionnaire de ressources pour fonctionner.
* Le cinquième service est “historyserver”, qui définit un historique de serveur Hadoop. Il utilise l'image Docker bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8 et est nommé "historyserver". Il utilise une variable d'environnement SERVICE\_PRECONDITION pour indiquer qu'il a besoin du nœud de nom, du nœud de données et du gestionnaire de ressources pour fonctionner. Il utilise un volume nommé hadoop\_historyserver pour stocker l'historique des tâches.
* Le dernier service est jupyter, qui définit un environnement Jupyter pour l'analyse de données. Il utilise l'image Docker jupyter/all-spark-notebook et est nommé "jupyter". Il expose le port 8888 pour permettre aux clients d'accéder à l'environnement Jupyter et utilise un volume nommé notebooks pour stocker les notebooks Jupyter. fez
  1. Collecte et traitement des données.

Concernant le phishing sur les liens, nous avons utilisé ces deux fichiers

o   Combined\_dataset.csv qui contient 12 colonnes mais nous nous sommes axés sur deux colonnes domain et label. Domain qui contient l’URL d’un site et label qui contient 0 si le lien est légitime et 1 si c’est un lien de phishing ou de spam

o   phishing\_site\_urls.csv qui contient deux colonnes URL et Label. URL qui contient l’URL d’un site et Label qui contient good si le lien est légitime et bad si c’est un lien de phishing ou de spam

Pour effectuer la prédiction nous avons d’abord:

* construire deux dataframes lisant les fichiers phishing\_site\_urls.csv et combined\_dataset.csv

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

* renommer les colonnes d’un dataframe pour que les dataframe ont les même colonnes

Une image contenant texte

Description générée automatiquement

* convertir le good en 0 et le bad en 1

Une image contenant texte

Description générée automatiquement

* fusionner les dataframes  pour en un obtenir un au final

Une image contenant texte

Description générée automatiquement

* supprimer les doublons dans ce nouveau dataframe

Une image contenant texte

Description générée automatiquement

Ensuite, suite au nouveau dataframe nous avons cherché les mots les plus utilisés dans liens frauduleux et dans les liens légitimes (plus la taille du mot est élevée plus le mot est utilisé). Pour ce faire nous avons créer une fonction qui crée et affiche un nuage de mots en utilisant le module WordCloud de la bibliothèque wordcloud avec les paramètres:

* text : la chaîne de texte sur laquelle le nuage de mots sera basé
* mask : un masque d'image facultatif à utiliser pour la forme du nuage de mots
* max\_words : le nombre maximal de mots à inclure dans le nuage de mots
* max\_font\_size : la taille maximale de police pour les mots dans le nuage de mots
* figure\_size : la taille de la figure de la trame
* title : un titre facultatif pour le tracé
* title\_size : la taille de police du titre

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

* 1. Développement du modèle d’apprentissage

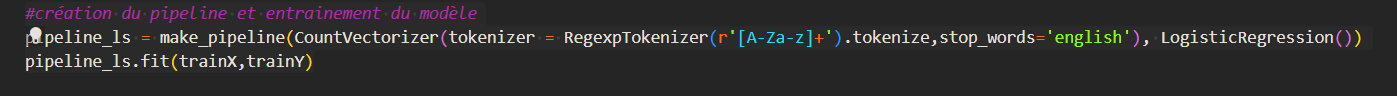
Concernant le développement du modèle d’analyse des URLs, à partir du nouveau dataframe df2 nous avons créer un dataset appeleé “newset.csv” et c’est à partir de ces données que nous déployer le model:

* Création des données  d’entrainement et de test

Une image contenant texte, capture d’écran, intérieur, écran

Description générée automatiquement

* création un pipeline et entrainement de données



* Visualisation du taux de précision des données test et d’entrainement

Une image contenant texte

Description générée automatiquement

* Tester le model

Une image contenant texte

Description générée automatiquement

Concernant le modèle chargé de détection de spam, nous avons utilisé le « Erom Spam dataset », constitué d’un ensemble de mail de spam répertorié.

Les étapes de traitement ont été réalisé comme suit :

1 - Une fois les sources de données récoltés et formatés de façon conforme à l’exploitation (ici en l'occurrence c’est notre fichier csv contenant les mails), nous utilisons les librairies tels que NumPy, Pandas, Scikit-learn qui va permettre d’appliquer l'algorithme de régression logistique.

2 - Charger les données à partir d'un fichier CSV à l'aide de la bibliothèque Pandas.

Une image contenant texte

Description générée automatiquement

3 - Diviser les données en ensembles de formation et de test en utilisant la fonction **train\_test\_split()** de Scikit-learn.

Une image contenant texte

Description générée automatiquement

4 - Convertir les données de texte en caractéristiques TF-IDF à l'aide de la classe **TfidfVectorizer()** de Scikit-learn.

Une image contenant texte

Description générée automatiquement

5 - Entraîner un modèle de régression logistique à l'aide de la classe **LogisticRegression()** de Scikit-learn.

Une image contenant texte

Description générée automatiquement

6 - Prédire les étiquettes de classe pour l'ensemble de test à l'aide de la méthode **predict()** du modèle.

Une image contenant texte

Description générée automatiquement

7 - Calculer la précision du modèle à l'aide de la fonction **accuracy\_score()** de Scikit-learn.

Une image contenant texte

Description générée automatiquement

Voici le dataset récupérer sur kaggle (Enron-Spam dataset) :

Une image contenant table

Description générée automatiquement

* 1. Test et déploiement du modèle via l’interface Jupyter

Présentation de l’interface de détection de spam :

Une image contenant texte

Description générée automatiquement

Ce code est une implémentation d'un modèle de classification de courrier électronique (spam / non-spam) en utilisant une technique de traitement automatique du langage naturel. Voici une explication du test ci-dessus :

1 -La variable input\_mail1 contient le texte du courrier électronique qui doit être classifié.

2 - *feature\_extraction* est une instance d'un objet qui a été entraîné sur un ensemble de données contenant des exemples de courrier électronique pour extraire des caractéristiques importantes des données textuelles. La fonction **transform** est appelée sur cet objet pour convertir le texte du courrier électronique en vecteurs de caractéristiques qui peuvent être utilisés en entrée pour le modèle.

3 - La variable model contient le modèle entraîné pour prédire si un courrier électronique est un spam ou non. La fonction **predict** est appelée sur cet objet pour prédire la classe du courrier électronique.

4 - Si la valeur de prédiction est égale à 1, cela signifie que le modèle a prédit que le courrier électronique est un courrier électronique non-spam (ham). Dans ce cas, le message "Ham mail" sera imprimé.

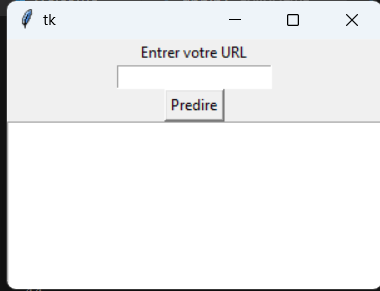
5 - Si la valeur de prédiction n'est pas égale à 1, cela signifie que le modèle a prédit que le courrier électronique est un spam. Dans ce cas, le message "Spam mail" sera imprimé.

Pour ce faire nous avons utilisé le fichier model.plk pour le modèle d’apprentissage et pour les données nous avons utilisés le fichier newset.csv.

Voici le code utiliser et les résultats

Présentation de l’application de détection de URLs :

Voici comment se présente l’interface :



Nous avons entré un lien légitime  https://www.linkedin.com/in/elsa-d-sifoua-2b7181236/

Une image contenant texte

Description générée automatiquement

Nous avons aussi testé avec un lien de phising fazan-pacir.rs/temp/libraries/ipad

Une image contenant texte

Description générée automatiquement