المعهد الوطني للبريد والمواصلات
ⵙⵜⵉⵜⵓⵜ ⵓⵏⵎⵓⵔⵓ ⵏ ⵜⵙⵙⵓⵏⵜ ⴰ ⵜⵎⵙⵙⵓⵏⵜ
Institut National des Postes et Télécommunications
Département Mathématiques, Informatique et Réseaux

Projet d'Ouverture Technologique

ARTIFICAL INTELLIGENCE IN ORDER TO IMPROVE INTRUSION DETECTION

Réalisé par :

SAMAOUI Khalid

DRISSI EL-BOUZAIDI Mouna

Encadré par :

Prof. IDRISSI HAMZA Kamal

Année Académique 2021-2022

Résumé

Récemment, les énormes quantités de données et leur augmentation progressive ont changé l'importance de la sécurité de l'information et des systèmes d'analyse des données pour le Big Data.

Le système de détection d'intrusion (IDS) est un système qui surveille et analyse les données pour détecter toute intrusion dans le système ou le réseau. Le volume, la variété et la vitesse élevés des données générées dans le réseau ont rendu très difficile le processus d'analyse des données pour détecter les attaques par des techniques traditionnelles.

Dans ce projet, on a utilisé des techniques de Big Data, à savoir le Machine Learning et le Deep Learning, dans les IDS pour traiter les Big Data afin d'obtenir un processus d'analyse des données précis et efficace.

Remerciements

Avant tout développement sur ce projet, il apparait opportun de commencer notre rapport par un remerciement à **M. IDRISSI HAMZA Kamal** de nous avoir aidé à réaliser ce projet et d'avoir sacrifié beaucoup de temps pour le rendre très profitable. Nous le remercions aussi pour ses conseils et ses remarques pertinentes.

Un grand merci à nos professeurs et toute l'équipe pédagogique de l'**Institut National des Postes et Télécommunications** pour leurs efforts durant ces deux dernières années qui nous ont été nécessaires à la compréhension des fondements théoriques utilisées pour réaliser un tel travail.

Table des matières

Introduction

La dataset choisie est **CICIDS2017** [1]. Elle contient les attaques courantes les plus récentes qui ressemblent aux véritables données du monde réel (*PCAP*). Elle comprend des flux étiquetés en fonction de l'horodatage, des IP source et destination, des ports source et destination, des protocoles et des attaques (fichiers *CSV*).

La période de capture des données a commencé à 9 heures, le lundi 3 juillet 2017 et s'est terminée à 17 heures, le vendredi 7 juillet 2017, soit un total de 5 jours. Les attaques mises en œuvre comprennent la force brute FTP, la force brute SSH, DoS, Heartbleed, l'attaque Web, l'infiltration, le botnet et le DDoS. Elles ont été exécutées le matin et l'après-midi des mardi, mercredi, jeudi et vendredi.

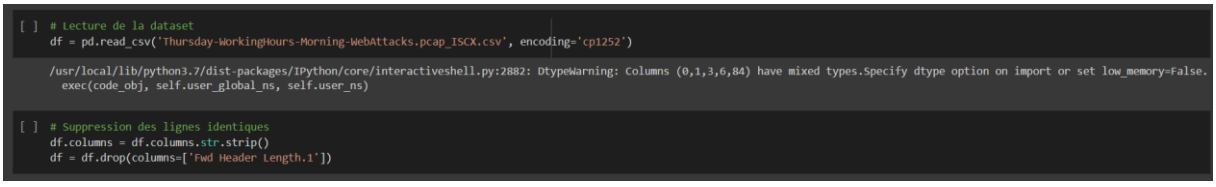
En raison de simplification et vu la grande taille de la Dataset, on a choisi de travailler sur la capture réalisée pendant le jeudi et uniquement sur l’attaque Web [2].

Application des Algorithmes de l’IA

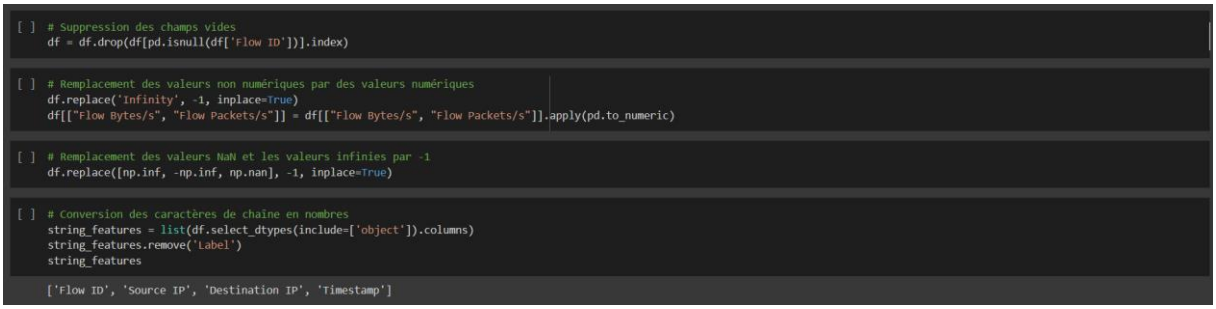
2.1 Pre-processing

La première phase est le nettoyage des données, elle comprend :

- Suppression des lignes identiques ;
- Suppression des champs vides ;
- Remplacement des valeurs non numériques par des valeurs numériques ;
- Remplacement des valeurs NaN et les valeurs infinies par -1 ;
- Conversion des caractères de chaîne en nombres.

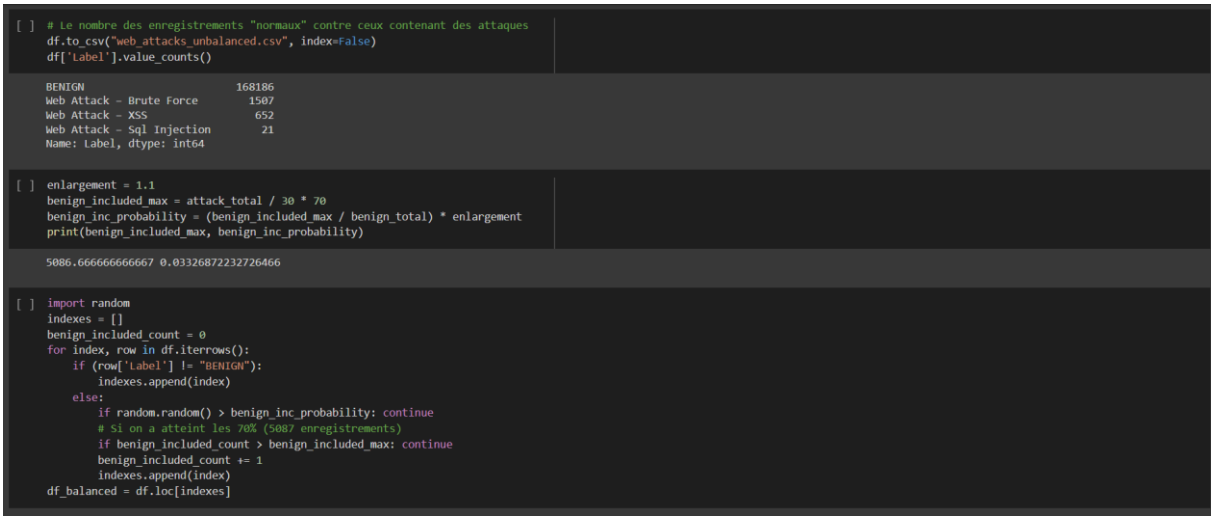


```
[ ]  # Lecture de la dataset
     df = pd.read_csv('Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv', encoding='cp1252')

     /usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882: DtypeWarning: Columns (0,1,3,6,84) have mixed types.Specify dtype option on import or set low_memory=False.
       exec(code_obj, self.user_global_ns, self.user_ns)

[ ]  # Suppression des lignes identiques
     df.columns = df.columns.str.strip()
     df = df.drop(columns=['Fwd Header Length.1'])
```



```
[ ]  # Suppression des champs vides
     df = df.drop(df[pd.isnull(df['Flow ID'])].index)

[ ]  # Remplacement des valeurs non numériques par des valeurs numériques
     df.replace('Infinity', -1, inplace=True)
     df[["Flow Bytes/s", "Flow Packets/s"]] = df[["Flow Bytes/s", "Flow Packets/s"]].apply(pd.to_numeric)

[ ]  # Remplacement des valeurs NaN et les valeurs infinies par -1
     df.replace([np.inf, -np.inf, np.nan], -1, inplace=True)

[ ]  # Conversion des caractères de chaine en nombres
     string_features = list(df.select_dtypes(include=['object']).columns)
     string_features.remove('Label')
     string_features

     ['Flow ID', 'Source IP', 'Destination IP', 'Timestamp']
```

Les données sont maintenant normalisées. Cependant, elles sont non équilibrées : les attaques ne présentent que 2180 sur la totalité 170366 des enregistrements. Pour cela, nous allons recourir à un sous-échantillonnage : les enregistrements contenant les attaques seront copiés à la nouvelle dataset et le nombre total des enregistrement « normaux » ne dépassera pas 5087.



```
[ ] # Le nombre des enregistrements "normaux" contre ceux contenant des attaques
    df.to_csv("web_attacks_unbalanced.csv", index=False)
    df['Label'].value_counts()

    BENIGN                      168186
    Web Attack - Brute Force      1507
    Web Attack - XSS               652
    Web Attack - Sql Injection      21
    Name: Label, dtype: int64

[ ] enlargement = 1.1
    benign_included_max = attack_total / 30 * 70
    benign_inc_probability = (benign_included_max / benign_total) * enlargement
    print(benign_included_max, benign_inc_probability)

    5086.666666666667 0.03326872232726466

[ ] import random
    indexes = []
    benign_included_count = 0
    for index, row in df.iterrows():
        if (row['Label'] != "BENIGN"):
            indexes.append(index)
        else:
            if random.random() > benign_inc_probability: continue
            # Si on a atteint les 70% (5087 enregistrements)
            if benign_included_count > benign_included_max: continue
            benign_included_count += 1
            indexes.append(index)
    df_balanced = df.loc[indexes]
```

Maintenant on va préparer les données pour l'entraînement.
On va coder les données comme suit :

Normal (BENIGN) = 0, Attaque = 1

On va aussi exclure les caractéristiques inutiles.

```
[ ] # Normal (BENIGN) = 0, Attaque = 1
    df['Label'] = df['Label'].apply(lambda x: 0 if x == 'BENIGN' else 1)

[ ] # Exclusion des caractéristiques inutiles
    excluded = ['Flow ID', 'Source IP', 'Source Port', 'Destination IP', 'Destination Port', 'Protocol', 'Timestamp']
    df = df.drop(columns=excluded, errors='ignore')
    df.shape

    (7267, 77)
```

2.2 Supervised Learning

Les modèles de l'apprentissage supervisé appliqués sur la dataset qu'on a sont : *Naive Bayes*, *Decision Tree*, *Linear SVC*, *SVC*, *KNN*, *Logistic Regression*, *Random Forest* et le *Gradient Boosting Classifier*. On a ensuite affiché le rapport de classification et la matrice de confusion de chacun d'eux.

Pré-acquis :

Naive Bayes Classifier **:**

Naive Bayes Classifier est un algorithme populaire en Machine Learning. C'est un algorithme du *Supervised Learning* utilisé pour la classification. Il est particulièrement utile pour les problématiques de classification de texte. Un exemple d'utilisation du *Naive Bayes* est celui du filtre anti-spam.

Le *Naive Bayes Classifier* se base sur le théorème de Bayes. Ce dernier est un classique de la théorie des probabilités. Ce théorème est fondé sur les probabilités conditionnelles. [4]

***Decision Tree / Random Forest* :**

Un arbre de décision est une structure de type organigramme dans laquelle chaque nœud interne représente un "test" sur un attribut, chaque branche représente le résultat du test et chaque nœud feuille représente une étiquette de classe. [5]

Les forêts aléatoires ou forêts de décision aléatoires sont une méthode d'apprentissage d'ensemble pour la classification, la régression et d'autres tâches qui fonctionne en construisant une multitude d'arbres de décision au moment de la formation.

***SVC / Linear SVC* :**

La méthode SVC (Support Vector Classifier) linéaire applique une fonction noyau linéaire pour effectuer la classification et elle est performante avec un grand nombre d'échantillons. Si nous le comparons au modèle SVC, le SVC linéaire possède des paramètres supplémentaires. [6]

***Logistic Regression* :**

La régression logistique est un processus de modélisation de la probabilité d'un résultat discret en fonction d'une variable d'entrée. [7]

Gradient Boosting Classifier :

Les classificateurs par boosting de gradient sont un groupe d'algorithmes d'apprentissage automatique qui combinent plusieurs modèles d'apprentissage faibles pour créer un modèle prédictif fort. Les arbres de décision sont généralement utilisés dans le cadre du gradient boosting.

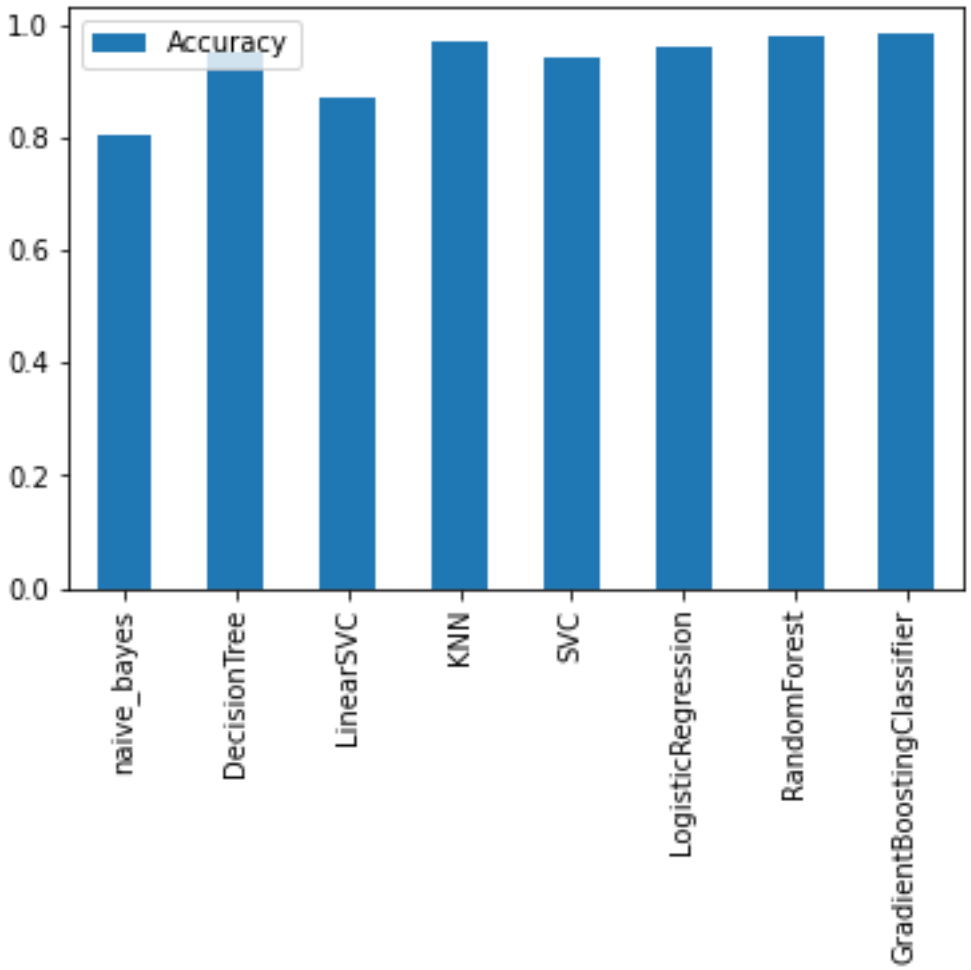
Résultats :

```
[ ]
--------------------Classification report for naive_bayes --------------------
              precision    recall  f1-score   support

           0       1.00      0.75      0.86      1010
           1       0.63      1.00      0.78       444

    accuracy                           0.82      1454
   macro avg       0.82      0.87      0.82      1454
weighted avg       0.89      0.82      0.83      1454

--------------------Confusion Matrix for naive_bayes --------------------
[[755 255]
 [  1 443]]
--------------------Classification report for DecisionTree --------------------
              precision    recall  f1-score   support

           0       0.98      0.96      0.97      1010
           1       0.92      0.96      0.94       444

    accuracy                           0.96      1454
   macro avg       0.95      0.96      0.96      1454
weighted avg       0.96      0.96      0.96      1454

--------------------Confusion Matrix for DecisionTree --------------------
[[973  37]
 [ 17 427]]
```

```
--------------------Classification report for LinearSVC --------------------
              precision    recall  f1-score   support

           0       0.95      0.94      0.95      1010
           1       0.87      0.89      0.88       444

    accuracy                           0.93      1454
   macro avg       0.91      0.92      0.91      1454
weighted avg       0.93      0.93      0.93      1454

--------------------Confusion Matrix for LinearSVC --------------------
[[950  60]
 [ 49 395]]
--------------------Classification report for KNN --------------------
              precision    recall  f1-score   support

           0       0.99      0.98      0.98      1010
           1       0.95      0.98      0.96       444

    accuracy                           0.98      1454
   macro avg       0.97      0.98      0.97      1454
weighted avg       0.98      0.98      0.98      1454

--------------------Confusion Matrix for KNN --------------------
[[988  22]
 [ 11 433]]
```

```
--------------------Classification report for SVC --------------------
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1010
           1       0.92      0.93      0.92       444

    accuracy                           0.95      1454
   macro avg       0.94      0.95      0.94      1454
weighted avg       0.95      0.95      0.95      1454

--------------------Confusion Matrix for SVC --------------------
[[975  35]
 [ 33 411]]
--------------------Classification report for LogisticRegression --------------------
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      1010
           1       0.96      0.93      0.95       444

    accuracy                           0.97      1454
   macro avg       0.97      0.96      0.96      1454
weighted avg       0.97      0.97      0.97      1454

--------------------Confusion Matrix for LogisticRegression --------------------
[[994  16]
 [ 31 413]]
```

```
--------------------Classification report for RandomForest --------------------
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1010
           1       0.97      0.98      0.98       444

    accuracy                           0.98      1454
   macro avg       0.98      0.98      0.98      1454
weighted avg       0.98      0.98      0.98      1454

--------------------Confusion Matrix for RandomForest --------------------
[[998  12]
 [ 10 434]]
--------------------Classification report for GradientBoostingClassifier --------------------
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1010
           1       0.97      0.98      0.98       444

    accuracy                           0.98      1454
   macro avg       0.98      0.98      0.98      1454
weighted avg       0.98      0.98      0.98      1454

--------------------Confusion Matrix for GradientBoostingClassifier --------------------
[[997  13]
 [  9 435]]
```



Interprétation :

Pour tous les algorithmes utilisés, on remarque que la diagonale de la matrice de confusion comporte des nombres très élevées, ce qui veut dire que tous les modèles sont plus ou moins bons pour le traitement de ces données vu leur bonne interprétation.

Concernant le F1-score, qui prend normalement en considération tous les autres paramètres, il a été très élevé pour le « *Gradient Boosting Classifier* » ce qui veut dire que de manière passive, sans tenir compte de notre préférence, cet algorithme reste le plus fort.

De même, si on prend en considération juste l'« *accuracy* » , l'algorithme « *Gradient Boosting Classifier* » sera le plus efficace. En revanche, ce qui nous intéresse dans la détection des intrusions c'est de capturer le maximum des *True Positif*, dans ce cas on va travailler avec le modèle « *Naive Bayes* » surtout si on utilise un IDS passif, sinon on va miser plus sur la précision pour ne pas réagir à de fausses alertes.

2.2 Unsupervised Learning

Le modèle de l'apprentissage non supervisé appliqué sur la dataset est le *K-Means*.

Pré-acquis :

K-Means **:**

K-means est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper en K clusters distincts les observations du data set. Ainsi les données similaires se retrouveront dans un même cluster. [8]

Résultats :

```
[ ] TP = 0
    TN = 0
    FP = 0
    FN = 0
    for i in range (1454):
      if klabel[i] == 0 and y_test[i] == 0:
          TN += 1
      if klabel[i] == 0 and y_test[i] == 1:
          FN += 1
      if klabel[i] == 1 and y_test[i] == 0:
          FP += 1
      if klabel[i] == 1 and y_test[i] == 1:
          TP += 1
    print(TP)
    print(TN)
    print(FN)
    print(FP)

    0
    880
    444
    130
```

Interprétation :

Etant donné que $TP = 0$, ce modèle semble être inutile vu qu'il ne détecte aucune attaque. Cela revient au fait qu'on a cinq classes mais on a fait la classification seulement sur deux. L'algorithme n'a donc pas pu trouver les similitudes entre les différentes classes d'attaques.

2.3 Deep Learning

On a travaillé avec le *CNN* (*Convolutional Neural Network*).

Pré-acquis :

***CNN*:**

Les CNN sont également connus sous le nom de réseaux de neurones artificiels invariants en translation ou invariants dans l'espace (SIANN), en raison de l'architecture à poids partagé des noyaux ou filtres de convolution qui glissent le long des caractéristiques d'entrée et fournissent des réponses équivariantes en translation, appelées cartes de caractéristiques.

Normalisation :

Cette phase comprend :

- Transformation des données en format numérique et leurs normalisations entre 0 et 1 ;
- Normalisation des valeurs entre 0 et 255 ;
- Sélection des données liées aux trafics "normaux" ;
- Sélection les données liées aux trafics malicieux. [9]

```
[ ] # Importation des modules et librairies nécessaires
    import os
    import cv2
    import math
    import random
    import shutil
    from sklearn.preprocessing import QuantileTransformer
    from PIL import Image
    import warnings
    warnings.filterwarnings("ignore")
```

```
[ ] # Transformation des données en format numérique et leurs normalisation entre 0 et 1
    numeric_features = df.dtypes[df.dtypes != 'object'].index
    scaler = QuantileTransformer()
    df[numeric_features] = scaler.fit_transform(df[numeric_features])
```

```
[ ] # Normalisation des valeurs entre 0 et 255
    df[numeric_features] = df[numeric_features].apply(
        lambda x: (x*255))
```

```
[ ] df['Label'].value_counts()

    0.0      5087
    255.0    2180
    Name: Label, dtype: int64
```

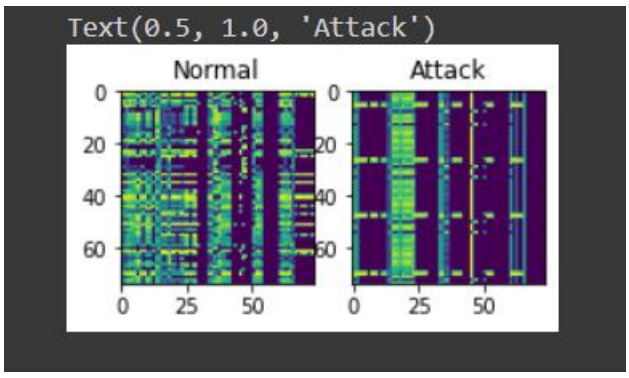
```
[ ] # Selectionner les données liées aux trafics "normaux"
    df0=df[df['Label']==0].drop(['Label'],axis=1)
    df0
```

```
[ ] # Selectionner les données liées aux trafics malicieux
    df1=df[df['Label']==255].drop(['Label'],axis=1)
    df1
```

	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	Bwd Packet Length Max	...	act_data_pkt_fwd	min_seg_size_forward	Active Mean	Active Std	Active Max
402	162.597264	217.094595	223.093093	235.141574	224.437736	223.343584	0.0	243.768769	235.653424	223.139390	...	212.500000	174.594595	0.0	0.0	0.0
403	152.611119	226.411411	223.093093	228.198198	244.493447	225.791506	0.0	228.882324	231.280652	253.213213	...	201.906907	174.594595	0.0	0.0	0.0
410	27.184685	0.000000	83.978979	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	...	0.000000	174.594595	0.0	0.0	0.0
411	169.020639	217.094595	217.477477	234.080952	221.589556	238.755357	0.0	242.568961	243.872299	223.336774	...	201.906907	174.594595	0.0	0.0	0.0
414	149.767194	217.094595	212.244745	234.139141	221.589556	238.933770	0.0	242.613313	244.022398	223.336774	...	201.906907	174.594595	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
5276	148.280232	212.244745	207.267267	222.837838	223.416918	240.450450	0.0	240.434840	247.199899	245.300300	...	161.576577	174.594595	0.0	0.0	0.0
5280	149.428947	212.244745	217.477477	222.837838	231.788951	240.450450	0.0	240.434840	247.199899	247.351041	...	161.576577	174.594595	0.0	0.0	0.0
5289	148.624806	204.969970	212.244745	222.837838	223.416918	240.450450	0.0	244.495927	248.082489	245.300300	...	161.576577	174.594595	0.0	0.0	0.0
5290	24.759760	0.000000	83.978979	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	...	0.000000	174.594595	0.0	0.0	0.0
5291	30.885886	0.000000	83.978979	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	...	0.000000	174.594595	0.0	0.0	0.0

2180 rows × 74 columns

Visualisation :



Création du model :

```
[ ] model = Sequential([
      Conv2D(num_filters, filter_size, input_shape=(224, 224, 3)),
      MaxPooling2D(pool_size=pool_size),
      Flatten(),
      Dense(2, activation='softmax'),
    ])

[ ] # Compile the model.
    model.compile(
      'adam',
      loss='categorical_crossentropy',
      metrics=['accuracy'],
    )

[ ] model.fit(
      train_generator,
      steps_per_epoch=len(train_generator),
      epochs=3,
      validation_data=validation_generator,
      validation_steps=len(validation_generator),
      callbacks=[history_this],
    )
```

Résultats :

```
CNN accuracy: 1.0
precision: 1.0
recall: 1.0
f1: 1.0
[[10  0]
 [ 0  9]]
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        10
           2       1.00      1.00      1.00         9

    accuracy                           1.00        19
   macro avg       1.00      1.00      1.00        19
weighted avg       1.00      1.00      1.00        19
```

Interprétation :

Les résultats sont satisfaisants mais la dataset n’a pas été trop large, on ne peut donc pas miser sur ce modèle.

Conclusion

D’après les résultats obtenus, les résultats de l’apprentissage supervisé semblent être les plus efficace et qui aboutissent aux résultats qu’on cherche.

Pour l’apprentissage non supervisé, les résultats obtenus n’ont pas été bien traité vu qu’il a fallu changer les paramètres d’initialisation pour que le modèle fonctionne correctement.

Par contre, dans le Deep Learning, les résultats ont été parfaits. On risque que le modèle apprenne par cœur exactement ces données de test et pas autres, et nous sera donc pas efficace dans les prochains traitements de données qui vont contenir de nouvelles datasets.

Bibliographie

[1] *https://www.unb.ca/cic/datasets/ids-2017.html*

[2] *https://github.com/fisher85/ml-cybersecurity/blob/master/python-web-attack-detection/datasets/Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv.zip?raw=true*

[3] *https://colab.research.google.com/github/fisher85/ml-cybersecurity/blob/master/python-web-attack-detection/web-attack-detection.ipynb#scrollTo=5PElFLh5oYzU*

[4] *https://mrmint.fr/naive-bayes-classifier*

[5] *https://en.wikipedia.org/wiki/Decision_tree#:~:text=A%20decision%20tree%20is%20a%20flowchart%2Dlike%20structure%20in%20which,taken%20after%20computing%20all%20attributes).*

[6] *https://www.datatechnotes.com/2020/07/classification-example-with-linearsvm-in-python.html*

[7] *https://www.sciencedirect.com/topics/computer-science/logistic-regression#:~:text=Logistic%20regression%20is%20a%20process,%2Fno%2C%20and%20so%20on.*

[8] *https://mrmint.fr/algorithme-k-means*

[9] *https://github.com/Western-OC2-Lab/Intrusion-Detection-System-Using-CNN-and-Transfer-Learning*