

# Rapport

## Analyse de Réseaux Sociaux

## Détection d'intrusions Cyber

**Présenté par :**

KAFANDO Tounwensida Bertrand

BASLAM Ismail

# Plan

## Introduction

I- Contexte et Problématique

II- Solution et Résultat

1. Construction d'un réseau et choix de la composante connexe
2. Classification des données de base
3. Génération des mesures et classification
4. Evaluation de l'importance des variables ajoutées

## Conclusion

# Introduction

En raison de la croissance massive des réseaux informatiques et des applications, de nombreux défis se posent pour la recherche en cybersécurité. Les intrusions/attaques peuvent être définies comme un ensemble d'événements capables de compromettre les principes des systèmes informatiques, par exemple la disponibilité, l'autorité, la confidentialité et l'intégrité. Pour répondre à ce besoin, le monde de la cybersécurité utilise les technologies et connaissances d'autres domaines comme l'intelligence artificiel (IA) et les graphes pour renforcer la sécurité des réseaux informatiques. A juste titre, la détection d'intrusion réseau (IDS) peut se faire d'une manière supervisée ou non supervisée avec l'IA. Notre travail à travers ce document consistera à la mise en place d'un IDS supervisé à partir des données UNSW-NB15 Dataset récolté par IXIA PerfectStorm tool in the Cyber Range Lab of UNSW Canberra.

Pour réaliser cela, nous construirons d'abord un graphe à partir des données du dataset. Ensuite nous récupérerons sa composante connexe la plus intéressante pour la suite de l'étude. Nous générerons enfin de cette composante connexe des mesures que nous ajouterons à notre dataset pour améliorer le modèle.

# I-Contexte et Problématique

## 1. Contexte

Le projet est réalisé dans le cadre de notre module Analyse de Réseaux Sociaux. Nous utiliserons donc les notions de réseau pour la phase de preprocessing des données pour optimiser le modèle.

Les données contiennent au total 49 caractéristiques avec l'étiquette de classe. Ces caractéristiques sont décrites dans le fichier UNSW-NB15\_features.csv. Le nombre total d'enregistrements est de deux millions et 540 044, qui sont stockés dans les quatre fichiers CSV, à savoir, UNSW-NB15\_1.csv, UNSWNB15\_2.csv, UNSW-NB15\_3.csv et UNSW-NB15\_4.csv. La table de vérité est nommée UNSW-NB15\_GT.CSV et la liste des fichiers d'événements est appelée UNSW-NB15\_LIST\_EVENTS.

L'objectif de ce projet est de mobiliser les techniques d'analyse des réseaux afin d'enrichir l'ensemble d'attributs d'un paquet par des mesures calculés à partir de la topologie du réseau d'interaction afin d'améliorer les performances de la classification automatique des paquets.

## 2. Problématique

L'analyse rapide des données nous montre que la répartition des classes n'est pas équilibrée. Cela peut réduire la capacité de généralisation du modèle. Vous pouvez le remarquer dans la figure 1.

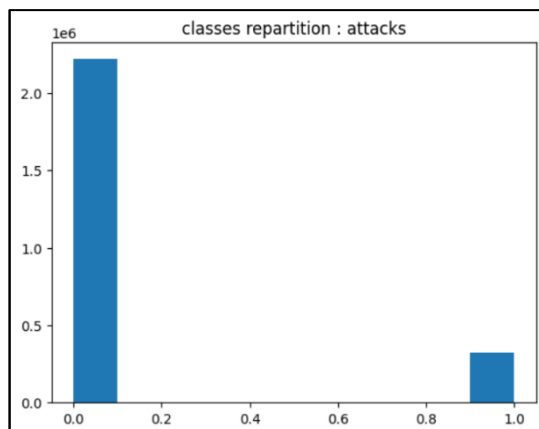


Figure 1: répartition des classes

Lorsque nous construisons un réseau avec toutes les données, nous avons un graphe de 1112287 de nœuds et 2540047 de connections. En regardant les figures 2 et 3 on s'aperçoit qu'il y a certains nœuds qui sont isolés. Ils n'apporteront donc pas d'information intéressante pour la classification.

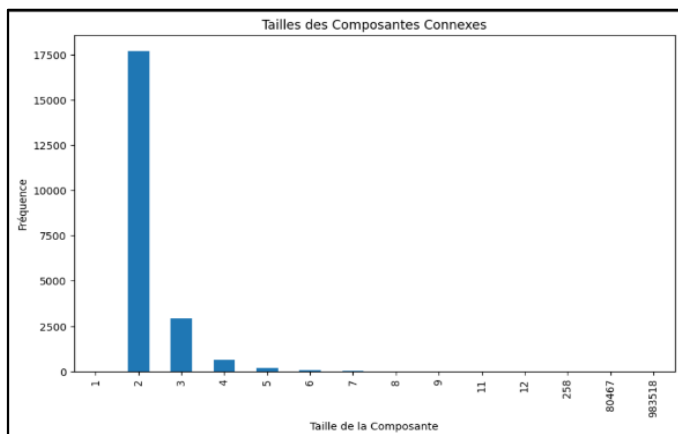


Figure 3: taille des composantes connexes

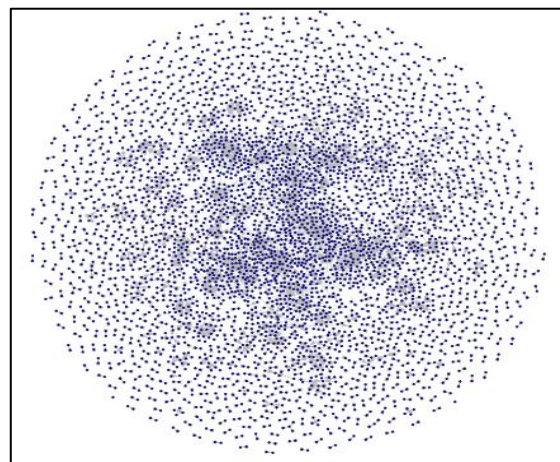


Figure 2: réseau

De ces observations découlent ces questions : Quelle composante connexe du graphe utilisée pour l'amélioration de la classification ? Quelles mesures générées pour l'amélioration du modèle de classification ?

Nous répondrons à ces différentes questions dans les lignes suivantes.

## II-Solution et Résultat

Pour avoir toutes les données, nous avons récupérés les données des quatre fichiers que nous avons fusionner. Nous avons pu récupérer le nom de chaque features à partir de la documentation. Le dataframe général obtenu nous sommes passés à la construction du réseau.

### 1. Construction du réseau et choix de la composante connexe

Pour construire le réseau, nous avons opté d'utiliser la paire adresses ip + port comme nœud. Grace à la librairie igraph nous avons obtenu le réseau correspondant aux données. Ci-joint une figure illustrant le réseau avec quelques nœuds.

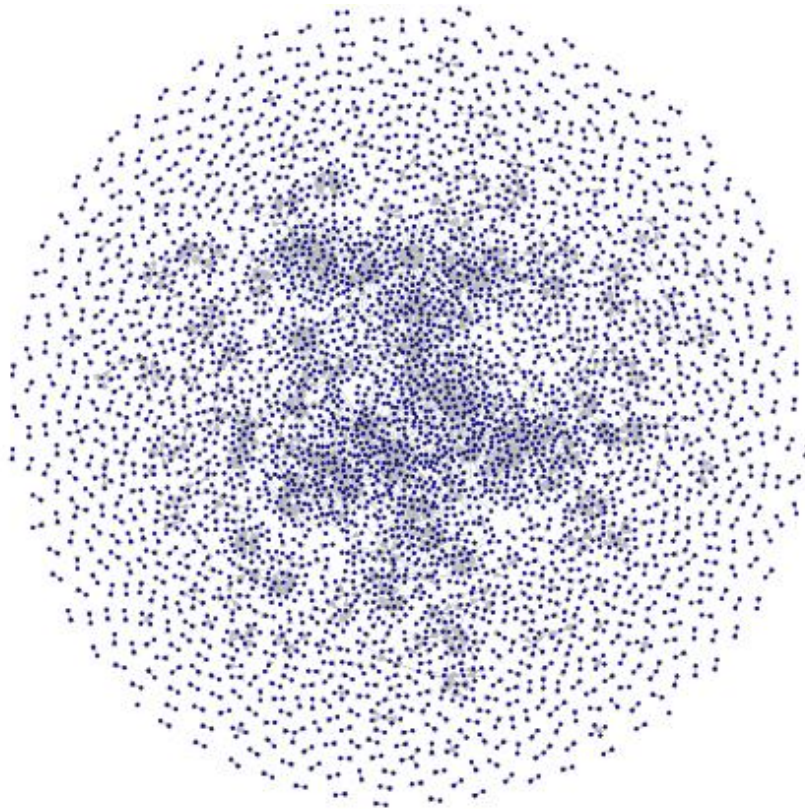


Figure 4: réseau

Comme illustrés plus haut dans la problématique, certains nœuds sont isolés, pour avoir des mesures optimales pour l'amélioration du modèle nous avons non seulement besoin de la plus grande composante connexe du graphe mais aussi la composante qui nous donnent une répartition équilibrée des différents labels de nos données.

De cette analyse, on déduit que la composante connexe 4, la deuxième plus grande composante connexe est celle qui répondent aux critères définis là-haut. Pour la suite de notre analyse nous utiliserons uniquement les lignes des données appartenant à cette composante.

### i. Preprocessing

- Avant de procéder à l'entraînement de notre modèle, une exploration visuelle des variables nominale a été réalisée pour comprendre leur influence potentielle sur la classification des connexions réseau. Pour ce faire, trois variables clés ont été prises en compte : le type de protocole (proto), l'état de la connexion (state) et le type de service (service).



Les graphiques de distribution pour chacune de ces variables nominales sont dans la figure 6 et 7 :

Ces graphiques offrent une vision claire de la corrélation entre les labels attribués aux connexions et les variables nominales sélectionnées. Par exemple, ils montrent que bien que certains états de connexion et certains services soient communément utilisés dans le trafic régulier, ils peuvent aussi être exploités dans des connexions malveillantes. Cette connaissance est essentielle pour l'élaboration de notre modèle de prédiction, car elle nous permet d'ajuster les poids des caractéristiques en fonction de leur pertinence.

Suite à notre exploration des caractéristiques catégorielles, nous avons tourné notre attention vers la sélection et le traitement des variables quantitatives, qui joueront un rôle crucial dans la performance de notre modèle de prédiction.

- Etude des variables numériques

Pour chaque variable quantitative, nous avons utilisé des graphiques en forme de violon afin d'examiner la distribution des valeurs en fonction des labels attribués. Ces visualisations détaillées nous ont permis de saisir la manière dont la variation de chaque variable peut influencer la distinction entre une connexion normale et une suspecte.

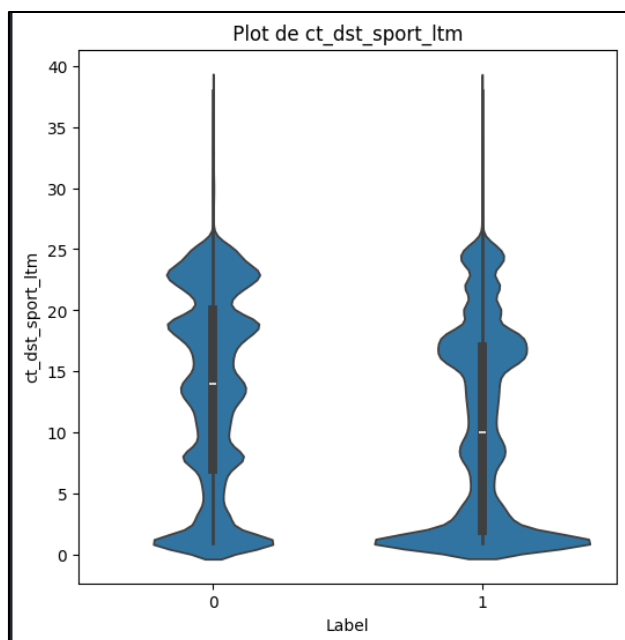


Figure 8: plot de ct\_des\_sport\_ltm

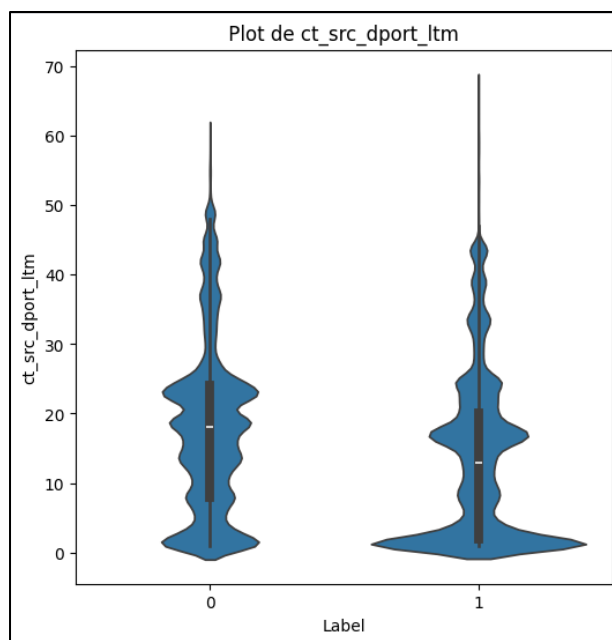


Figure 9: plt de ct\_src\_dport\_ltm

Les graphiques en forme de violon nous montrent non seulement où la majorité des données se situe, mais aussi où se trouvent les points extrêmes. Cela nous donne un aperçu visuel de la façon dont chaque caractéristique peut affecter la classification d'une connexion en tant que normale ou suspecte.



- Étude de la corrélation inter-variables

Afin de saisir pleinement les dynamiques internes de notre ensemble de données, nous avons procédé à un calcul de corrélation entre les variables quantitatives. La matrice de corrélation (figure 10) résultante nous fournit une vue d'ensemble des liens potentiels entre les différentes mesures de trafic réseau. À l'aide de notre matrice de corrélation initiale, nous avons détecté et supprimé les variables qui partageaient une forte corrélation (supérieure à 0,8), ce qui indique souvent une redondance d'informations. Ce nettoyage a permis de mieux distinguer les caractéristiques uniques de chaque variable, facilitant ainsi leur interprétation dans le contexte de notre modèle.

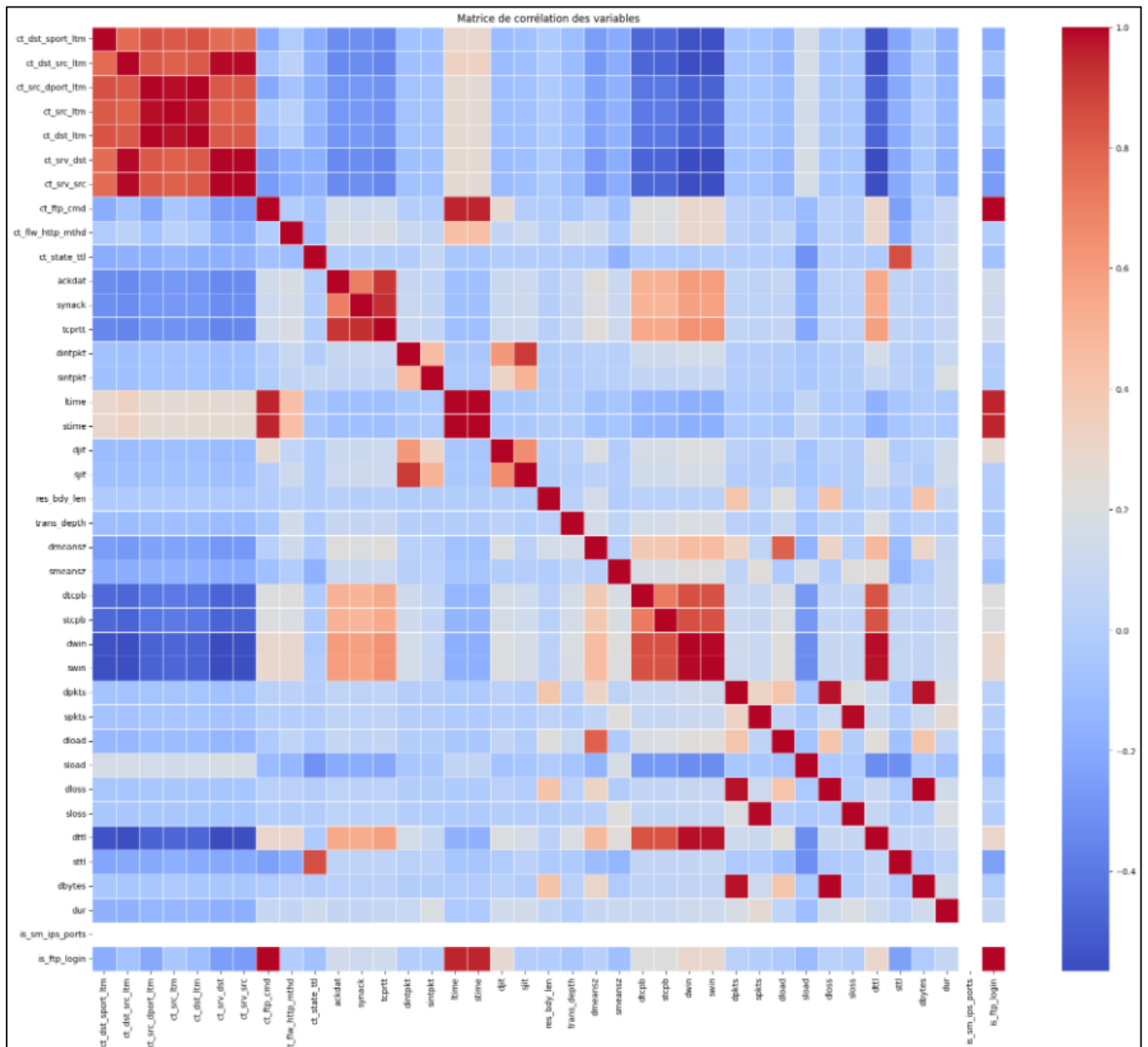


Figure 10: matrice de corrélation des variables

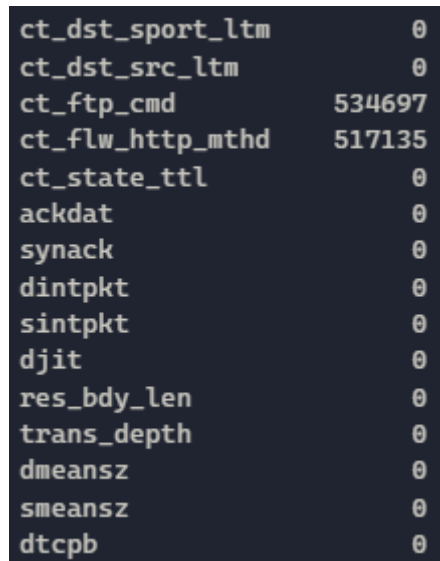
- Normalisation des Données pour une Analyse Cohérente

Après avoir affiné notre ensemble de caractéristiques, nous avons abordé la question de la mise à l'échelle des variables numériques. Pour que notre modèle puisse traiter chaque variable sur un pied d'égalité, sans être influencé par la gamme de valeurs de chacune, nous avons normalisé nos données en utilisant la technique de MinMaxScaler. Cette méthode réajuste les données dans un intervalle compris entre 0 et 1.

- Gestion des Valeurs Manquantes

L'étape suivante a consisté à garantir l'intégrité complète de notre ensemble de données en traitant les valeurs manquantes. Les valeurs manquantes peuvent poser de sérieux problèmes lors de l'entraînement de modèles de machine learning, car elles peuvent introduire des biais ou des erreurs dans les résultats.

Pour aborder ce problème, nous avons commencé par identifier où se situaient ces valeurs manquantes. Voici le décompte des valeurs nulles obtenues pour chaque variable :



ct_dst_sport_ltm	0
ct_dst_src_ltm	0
ct_ftp_cmd	534697
ct_flw_http_mthd	517135
ct_state_ttl	0
ackdat	0
synack	0
dintpkt	0
sintpkt	0
djit	0
res_bdy_len	0
trans_depth	0
dmeansz	0
smeansz	0
dtcpb	0

Figure 11: valeur manquantes par variables

Après avoir identifié les valeurs manquantes dans notre jeu de données, nous avons pris la décision de retirer certaines variables qui présentaient un nombre important de valeurs non renseignées. Les colonnes `ct_ftp_cmd` et `ct_flw_http_mthd` ont été supprimées de notre ensemble final de données en raison de leur proportion élevée de valeurs manquantes.

## ii. Entraînement et Évaluation du Modèle de Machine Learning

L'étape finale dans le traitement de nos données a été la préparation des ensembles d'entraînement et de test, qui sont fondamentaux pour la construction et l'évaluation de notre modèle de classification. Nous avons d'abord séparé les caractéristiques (X) des étiquettes (y), où 'Label' est la variable que nous cherchons à prédire.

Afin d'intégrer les variables catégorielles dans notre modèle, nous avons converti ces caractéristiques en variables dummy, une pratique standard qui transforme les catégories en une série de variables binaires. Cela a été réalisé via `pd.get_dummies`, excluant la première catégorie pour éviter le piège de la multicollinéarité.

Avec les données préparées, nous avons divisé notre ensemble en un ensemble d'entraînement (80%) et un ensemble de test (20%), en utilisant une séparation aléatoire mais reproductible grâce à un `random_state` fixé.

Nous avons choisi un modèle SVM (Machine à Vecteurs de Support) avec un noyau radial pour sa capacité à gérer les espaces de caractéristiques de grande dimension et sa flexibilité dans la gestion des classifications non linéaires. Après avoir entraîné le modèle sur notre ensemble d'entraînement, nous avons procédé à des prédictions sur l'ensemble de test.

Les performances du modèle ont été évaluées en utilisant la matrice de confusion et le rapport de classification. La matrice de confusion nous donne une vue d'ensemble des prédictions correctes et incorrectes, tandis que le rapport de classification fournit des métriques plus détaillées telles que la précision, le rappel et le score F1 pour chaque classe.

	precision	recall	f1-score	support
0	0.98	0.90	0.94	50619
1	0.93	0.99	0.96	63239
accuracy			0.95	113858
macro avg	0.96	0.95	0.95	113858
weighted avg	0.95	0.95	0.95	113858

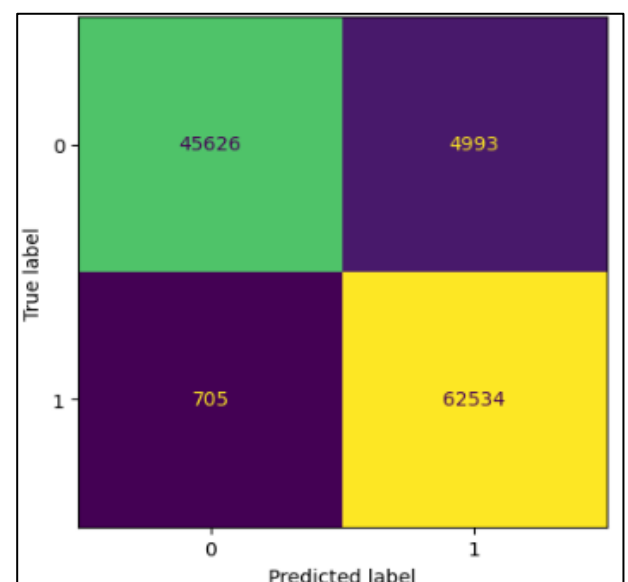


Figure 12: matrice de confusion

### 3. Génération des mesures et classification

#### i. Génération des mesures

Un graphe possède plusieurs propriétés intéressantes. Nous avons décidé de générer principalement des mesures de centralités pour l'amélioration du modèle pour chaque nœud destination et source. Nous avons ajouté neuf (9) mesures dont 18 caractéristiques en plus. Voici les caractéristiques générées :

- **Le degré** : La centralité de degré mesure l'importance d'un nœud en fonction du nombre de liens qui y aboutissent ou en partent. Un nœud avec un degré élevé est souvent considéré comme un point de passage ou de distribution clé dans le réseau, indiquant qu'il peut influencer ou être influencé par de nombreux autres nœuds.
- **La Centralité Closeness** : La centralité de proximité quantifie à quel point un nœud est proche de tous les autres nœuds dans le réseau. Un score élevé signifie qu'un nœud peut rapidement interagir avec tous les autres nœuds, ce qui le rend stratégique pour la diffusion rapide d'informations ou de ressources à travers le réseau.
- **La centralité betweenness** : La centralité d'intermédierité mesure la fréquence à laquelle un nœud se trouve sur le chemin le plus court entre deux autres nœuds. Les nœuds ayant une forte centralité d'intermédierité ont un grand contrôle sur la circulation des informations dans le réseau, car ils agissent comme des ponts entre différents groupes ou communautés.
- **Centralité de vecteur propre** : La centralité de vecteur propre évalue l'influence d'un nœud en tenant compte non seulement de son nombre de connexions mais aussi de la qualité de ces connexions. Un nœud connecté à de nombreux nœuds très connectés obtient un score élevé, ce qui suggère qu'il est bien positionné dans un réseau d'influenceurs.

- **Le coefficient de Clustering** : Le coefficient de regroupement mesure la tendance des nœuds à créer des clusters ou des groupes. Un coefficient élevé indique que les voisins d'un nœud sont également connectés entre eux, ce qui peut signifier une forte cohésion sociale ou une robustesse dans le réseau.
- **PageRank** : PageRank est une forme de mesure de la centralité qui attribue une importance à un nœud en fonction de l'importance de ses voisins. Initialement conçu pour classer les pages web, PageRank est utile pour identifier les nœuds influents dans un large éventail de réseaux.
- **La centralité K-core** : La centralité de K-core mesure l'appartenance d'un nœud à un "core" ou un noyau dans le réseau. Un score élevé indique qu'un nœud fait partie d'un groupe central dense de nœuds hautement interconnectés, ce qui peut être un indicateur de résilience du réseau.
- **Authority Score** : Le score d'autorité, issu de l'algorithme HITS (Hyperlink-Induced Topic Search), identifie les nœuds qui contiennent des informations précieuses et sont référencés par de nombreux autres nœuds. Les nœuds avec un score d'autorité élevé sont considérés comme des sources de contenu de haute qualité dans le réseau.
- **Hub score** : Le score de hub, également issu de l'algorithme HITS, mesure l'aptitude d'un nœud à bien se connecter à des nœuds d'autorité élevée. Les hubs agissent comme des pointeurs vers des sources d'information précieuses, et un score élevé indique un nœud qui fait efficacement le lien avec des nœuds importants.

## ii. Classification et évaluation

Après la génération de nouvelles mesures, nous avons réentraîné notre modèle SVM.

Avant nous avons fait une phase de preprocessing. La matrice de corrélation ci-dessous (figure 6) montre que parmi les champs générés, pagerank\_dst, closeness\_centrality\_dst, authority\_score\_src. Elles seront donc supprimées pour optimiser l'entraînement.

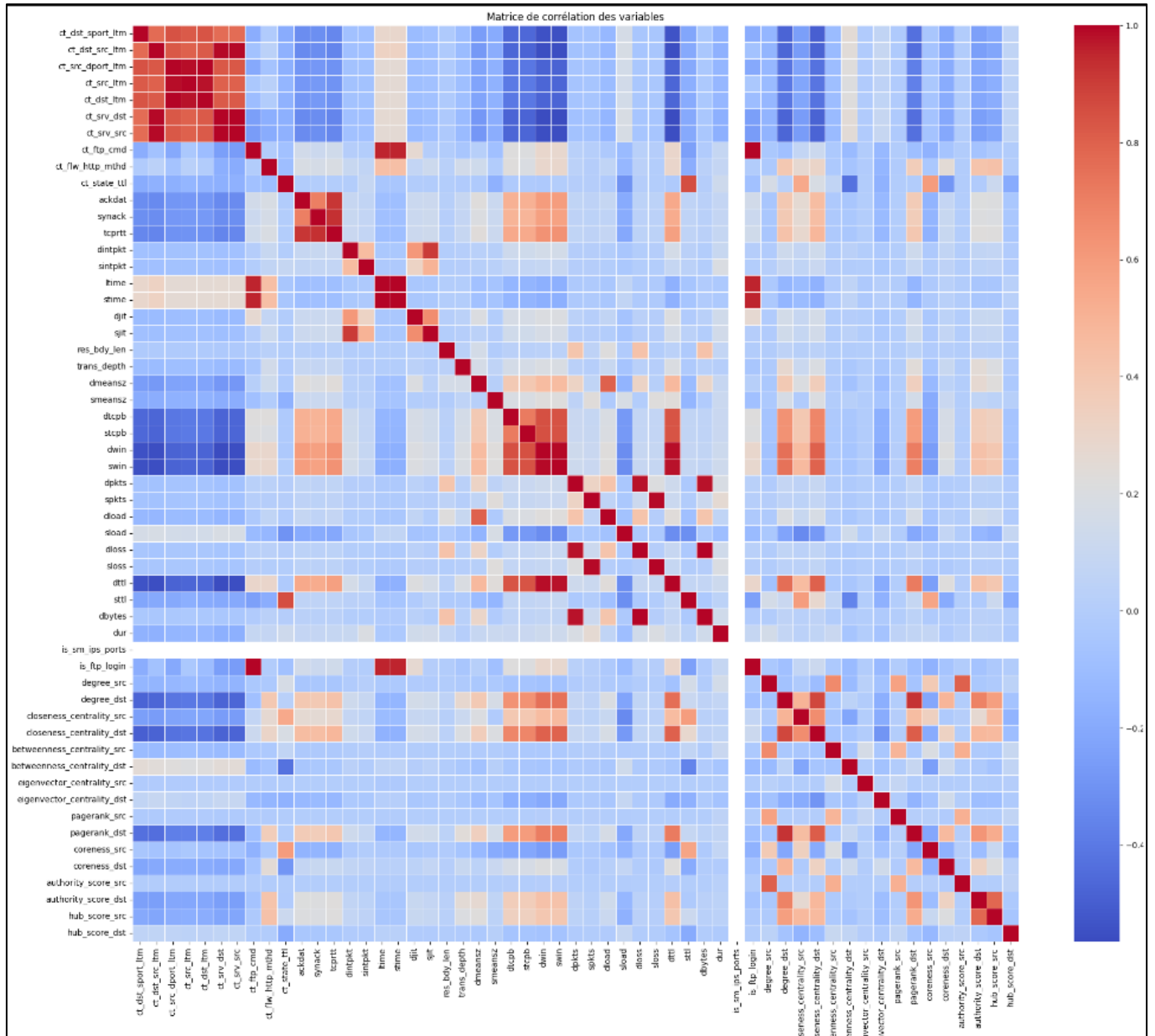


Figure 13: matrice de corrélation v2

Le code ci-dessous nous permet d'entraîner notre modèle une nouvelle fois et d'afficher la matrice de confusion.

```
# Séparation des données en ensembles d'entraînement et de test
X = final_update_df.drop(columns='Label')
y = final_update_df['Label']

# Création de variables dummy pour les variables catégorielles
X_dummies = pd.get_dummies(X, drop_first=True)

# Séparation des données en ensembles d'entraînement et de test après avoir créé des variables dummy
X_train, X_test, y_train, y_test = train_test_split(X_dummies, y, test_size=0.2, random_state=42)

# Entraînement du modèle SVM avec un noyau radial ('rbf' est le choix par défaut)
svm_model = SVC(kernel='rbf')
svm_model.fit(X_train, y_train)

# Prédiction sur l'ensemble de test
y_pred = svm_model.predict(X_test)

# Évaluation du modèle
print("Confusion Matrix update df:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report update df:")
print(classification_report(y_test, y_pred))
```

On obtient une précision, un recall, un f1-score très élevée de 96%. Ces scores sont meilleurs comparés au résultat précédent. Ce qui traduit la capacité du modèle à généraliser d'où une amélioration du modèle. Les figures suivantes nous montrent les résultats.

Confusion Matrix update df:					
[[48315 2304]					
[ 1919 61320]]					
Classification Report update df:					
	precision	recall	f1-score	support	
0	0.96	0.95	0.96	50619	
1	0.96	0.97	0.97	63239	
accuracy			0.96	113858	
macro avg	0.96	0.96	0.96	113858	
weighted avg	0.96	0.96	0.96	113858	

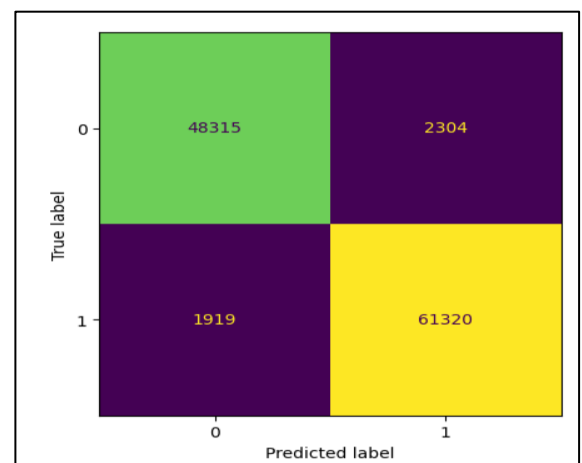


Figure 14: résultat prédiction

On obtient un résultat une amélioration du modèle. Mais nous ne savons pas vraiment si cette performance est obtenue grâce aux nouvelles variables. Dans la suite nous allons voir si les nouvelles variables impactent la classification.

#### 4. Evaluation de l'importance des variables ajoutées

Pour avoir l'importance des nouvelles variables sur le modèle, nous allons entraîner un modèle RandomForest qui nous retournera cette mesure en utilisant le code ci-dessous.

```

# Assurer que la colonne 'Label' existe dans votre DataFrame
if 'Label' in final_update_df.columns:
    # Séparation des données en ensembles d'entraînement et de test
    X = final_update_df.drop(columns='Label')
    y = final_update_df['Label']

    # Création de variables dummy pour les variables catégorielles
    X_dummies = pd.get_dummies(X, drop_first=True)

    # Séparation des données
    X_train, X_test, y_train, y_test = train_test_split(X_dummies, y, test_size=0.2, random_state=42)

    # Entraînement du modèle Random Forest
    rf_model = RandomForestClassifier(random_state=42)
    rf_model.fit(X_train, y_train)

    # Prédiction et évaluation
    y_pred = rf_model.predict(X_test)
    #print("Confusion Matrix update df:")
    #print(confusion_matrix(y_test, y_pred))
    #print("\nClassification Report update df:")
    #print(classification_report(y_test, y_pred))

    # Afficher l'importance des variables
    feature_importances = rf_model.feature_importances_
    importance_df = pd.DataFrame({'Feature': X_dummies.columns, 'Importance': feature_importances})
    print("\nFeature Importances:")
    print(importance_df.sort_values(by='Importance', ascending=False))
else:
    print("La colonne 'Label' est manquante dans le DataFrame.")

```

Figure 15: model random forest

La figure ci-dessous nous montre le résultat. Nous pouvons souligner que les mesures hub score, le degree, la centralité closeness, betweenness et vecteur propre permettent au modèle de classifier. Notre expérience est très concluante.

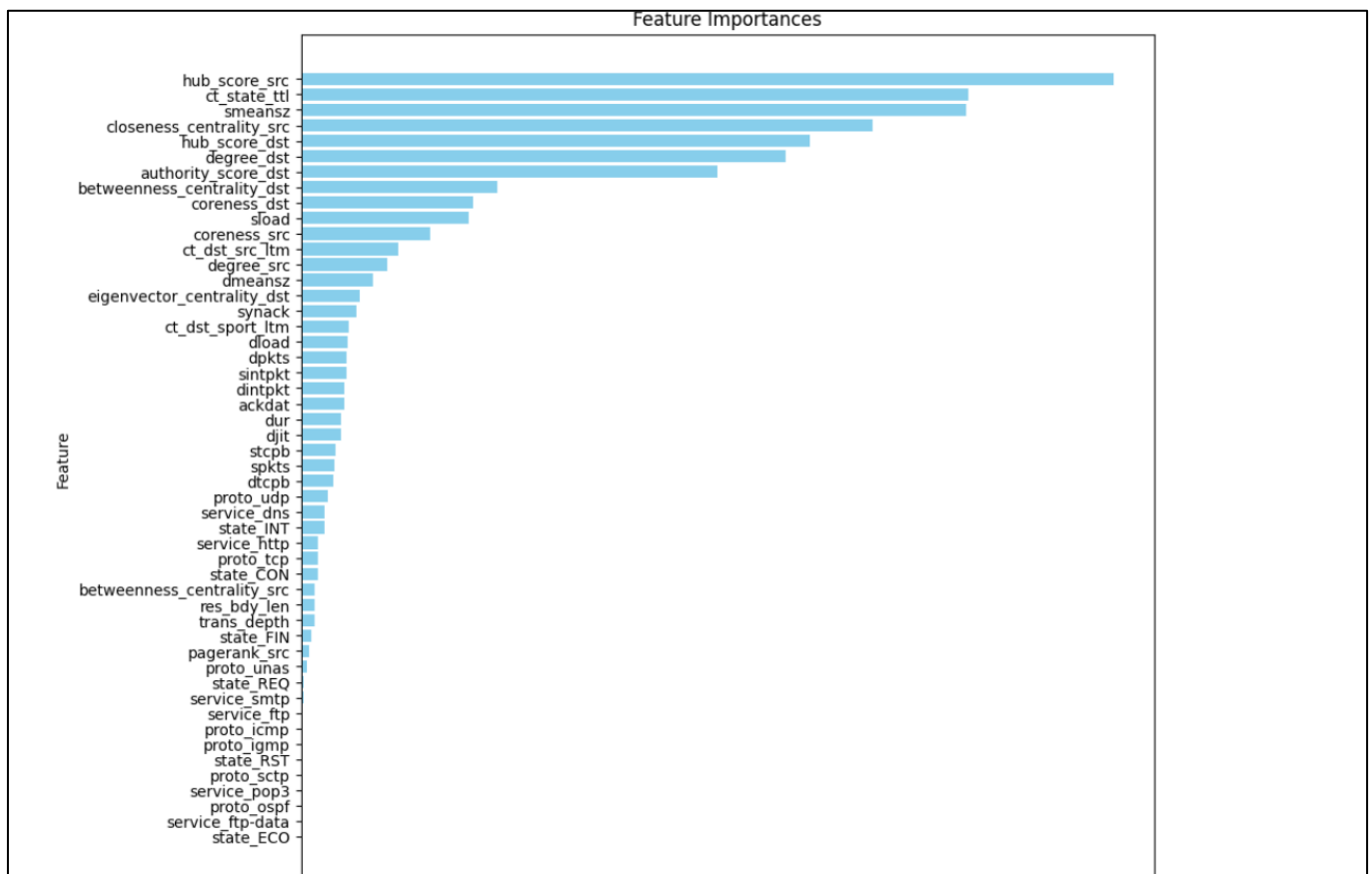


Figure 16: Importance des features



## Conclusion

En conclusion, nous avons élaboré un graphe à partir des données disponibles. Grâce à l'exploration des propriétés de ce graphe, nous avons extrait un ensemble de données représentatif et optimisé pour l'entraînement de notre modèle. Nous avons ensuite enrichi ce graphe avec diverses mesures de centralité, qui, selon les résultats obtenus avec le modèle de forêt aléatoire, jouent un rôle significatif dans la classification des données.

Ce projet a constitué une expérience enrichissante, nous offrant l'occasion d'appliquer nos connaissances en analyse des réseaux sociaux. Cependant, le chemin n'a pas été exempt d'obstacles. Étant donné la volumétrie des données, l'entraînement de nos modèles s'est avéré particulièrement long, dépassant les 3 heures. De plus, la taille conséquente de la composante connexe sélectionnée nous a empêchés de l'afficher dans son intégralité.

Pour conclure, nous tenons à souligner l'intérêt profond que nous portons à ce projet. Nous proposons d'explorer une nouvelle voie en utilisant le Deep Learning pour la suite de cette recherche, une perspective que nous jugeons prometteuse.