

Projet:

Support Vector Machine pour détecter les cyberattaques dans les environnements IOT

Ce projet montre le rôle de l'intelligence artificielle (le machine learning) dans la cybersécurité en permettant la détection proactive des menaces.

Ce rôle peut aller jusqu'à l'analyse rapide des attaques et l'automatisation des réponses, renforçant ainsi la protection des systèmes face aux cyberattaques toujours plus sophistiquées.



Elaboré par:
Abir JLASSI

Sommaire

- I. Aperçu du problème
 - 1. Objectifs
 - 2. Présentation du dataset
- II. Théorie des Machines à Vecteurs de Support (SVM)
 - 1. Définition
 - 2. Fonctionnement
 - 3. Fonctions noyaux
 - 4. Paramètres des SVMs
 - 5. Evaluation des performances
- III. Implémentation: Applications des SVMs pour détecter les cyberattaques
 - 1. Exploration des données
 - 2. Préparation des données
 - 3. Sélection des caractéristiques
 - 4. Entraînement du modèle
 - 5. Evaluation du modèle
 - 6. Améliorations: Optimisation du modèle
 - 6.1 Optimisation des hyperparamètres
 - 6.2 Gestion du déséquilibre des classes

Conclusion

I. Aperçu du problème

L'adoption rapide des dispositifs de l'Internet des objets (IoT) dans les industries et les foyers a accru le risque de cyberattaques. Les attaquants exploitent les vulnérabilités de l'IoT pour lancer des attaques telles que le déni de service (DoS), le déni de service distribué (DDoS), les injections de données et les accès non autorisés.

Les mesures de sécurité traditionnelles peinent à détecter ces menaces en raison de la diversité des dispositifs IoT et de leurs mécanismes de sécurité limités. Par conséquent, les systèmes de détection d'intrusion (IDS) basés sur l'apprentissage automatique (ML) sont apparus comme une solution efficace.

Ce projet explore l'utilisation des machines à vecteurs de support (SVM) pour détecter le trafic réseau malveillant dans un environnement IoT en utilisant le jeu de données TON-IoT, développé par l'Université de New South Wales (UNSW). Ce jeu de données contient des données de télémétrie, du trafic réseau et des journaux système pour aider à identifier les activités malveillantes.

Objectif

L'objectif de ce projet est de développer un système de détection d'intrusion (IDS) basé sur l'apprentissage automatique en utilisant les machines à vecteurs de support (SVM) pour identifier les activités malveillantes dans les environnements IoT. Plus précisément, le projet se concentre sur l'analyse des données de trafic réseau du jeu de données TON-IoT afin de distinguer les comportements normaux des comportements d'attaque.

Pourquoi utiliser les SVM ?

- Les machines à vecteurs de support (SVM) sont particulièrement adaptées à cette tâche en raison de leur capacité à :
- Gérer efficacement les données de haute dimension.
- Obtenir de bonnes performances avec des jeux de données de petite à moyenne taille.
- Modéliser des frontières de décision complexes grâce à des fonctions noyau (linéaire, RBF, etc.).
- Fournir des performances de généralisation robustes lorsqu'elles sont correctement paramétrées.

Présentation du dataset TON-IOT

Le jeu de données TON-IoT se compose de trois principales sources de données :

- Données de télémétrie IoT (journaux basés sur les capteurs)
- Journaux du système d'exploitation (activité des systèmes Windows/Linux)
- Données de trafic réseau (caractéristiques basées sur les paquets et les flux)

Source de données	Description	Pertinence pour les SVM
Données de télémétrie IoT	Relevés des capteurs IoT (température, humidité, etc.)	Pertinence limitée
Journaux du système d'exploitation	Journaux d'activité des systèmes Windows/Linux	Utile pour la détection d'intrusion basée sur l'hôte, mais moins adaptée aux SVM sans traitement NLP
Données de trafic réseau	Caractéristiques des paquets et des flux réseau	Très pertinentes pour les SVM (les attaques réseau telles que le DDoS ou les injections peuvent être identifiées par les motifs de trafic)

Pour ce projet, je travaillerai avec les données de trafic réseau car :

- La plupart des cyberattaques dans les environnements IoT sont basées sur le réseau (DDoS, MITM, reconnaissance, injection).
- Les SVM fonctionnent mieux avec des caractéristiques numériques structurées, abondantes dans les données de flux réseau.
- Les données de trafic réseau contiennent des étiquettes bien définies, ce qui les rend idéales pour la classification.

Le jeu de données de trafic réseau comprend plusieurs caractéristiques telles que :

- Adresses IP et ports source & destination
- Protocole & état de la connexion
- Tailles des paquets (src_bytes, dst_bytes)
- Durée des flux de trafic
- Étiquettes d'attaque (normal ou type d'attaque : DDoS, injection, etc.)

IP & Port Information (**src_ip**, **dst_ip**, **src_port**, **dst_port**)

Traffic Metrics (**duration**, **src_bytes**, **dst_bytes**, **src_pkts**, **dst_pkts**, etc.)

Connection State (**conn_state**)

Protocol & Service (**proto**, **service**)

DNS & SSL Features

HTTP Features

Attack Labels (**label**, **type**)

II. Théorie des Machines à Vecteurs de Support (SVM)

1. Définition

Les Machines à Vecteurs de Support (SVM) sont des algorithmes d'apprentissage supervisé utilisés principalement pour la classification et la régression. Initialement conçues pour des tâches de classification binaire, elles ont été généralisées pour les problèmes multiclassés. L'idée principale des SVM est de trouver un hyperplan optimal qui sépare les données de différentes classes avec la plus grande marge possible.

2. Fonctionnement

Les SVM cherchent à maximiser la marge entre les données des différentes classes, c'est-à-dire la distance entre l'hyperplan séparateur et les exemples les plus proches de chaque classe (appelés vecteurs de support). Cette approche permet d'augmenter la capacité de généralisation du modèle.

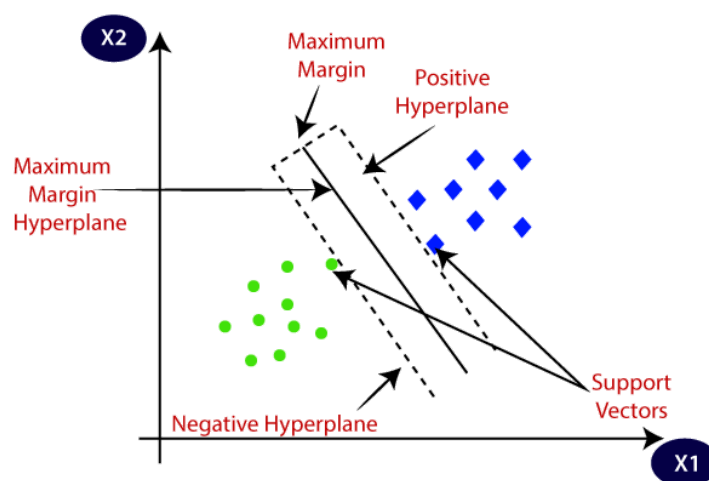


Figure 1 : fonctionnement des SVM

- **Classification linéaire** : Lorsqu'une séparation linéaire est possible, l'algorithme cherche l'hyperplan optimal qui maximise la marge.
- **Classification non linéaire** : Pour des données non linéairement séparables, les SVM utilisent des fonctions noyaux (kernels) pour projeter les données dans un espace de dimension supérieure où une séparation linéaire est possible.

3. Les fonctions noyaux

Les SVM utilisent différentes fonctions noyaux pour traiter les problèmes non linéaires

- **Noyau linéaire** : $K(x, x') = x \cdot x'$
- **Noyau polynomial** : $K(x, x') = (x \cdot x' + c)^d$
- **Noyau gaussien (RBF)** : $K(x, x') = \exp(-\gamma \|x - x'\|^2)$
- **Noyau sigmoïde** : $K(x, x') = \tanh(\alpha x \cdot x' + c)$

4. Paramètres des SVMs

- **C (coût)** : Contrôle le compromis entre maximisation de la marge et minimisation des erreurs de classification. Un C élevé réduit les erreurs mais risque de surapprentissage.
- **Gamma (γ)** : Paramètre du noyau RBF qui contrôle l'influence d'un exemple de formation. Un γ élevé réduit la portée, tandis qu'un γ faible l'étend.

5. Évaluation des performances

Pour évaluer les performances d'un modèle SVM, plusieurs métriques peuvent être utilisées :

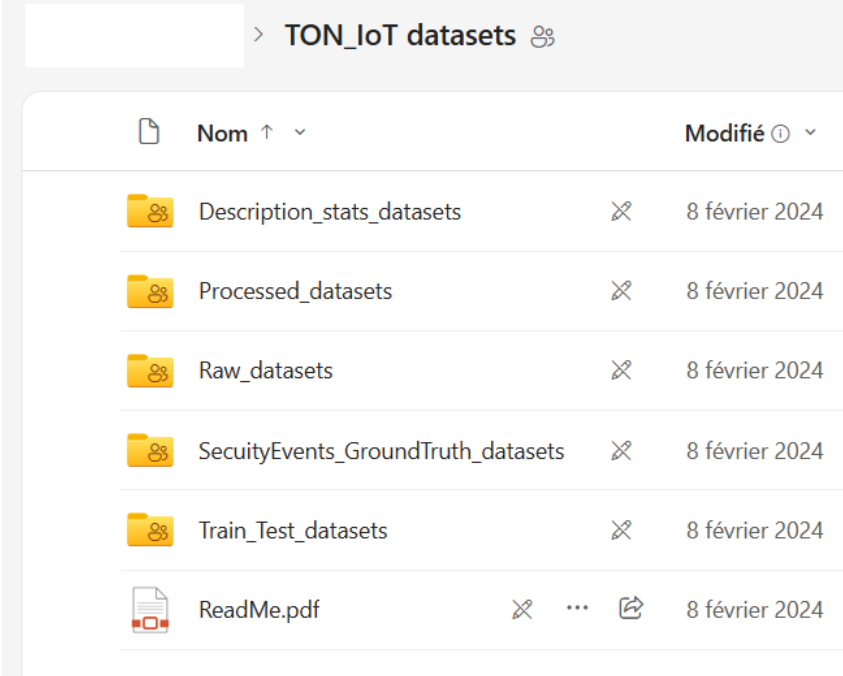
- **Précision (Accuracy)** : Pourcentage correct de prédictions.
- **Précision (Precision)** : Capacité à éviter les faux positifs.
- **Rappel (Recall)** : Capacité à détecter les vrais positifs.
- **F-mesure (F1-Score)** : Moyenne harmonique de la précision et du rappel.
- **Matrice de confusion** : Permet de visualiser les performances du modèle en fonction des classes prédites et réelles.

III. Implémentation: Applications des SVMs pour détecter les cyberattaques

1. Importation et exploration des données

Dataset:

Le jeu de données utilisé pour cette étude provient de 'TON_IoT' (Telecommunication Networks IoT), un ensemble de données conçu pour l'analyse des menaces et des cyberattaques dans les environnements IoT. Il a été développé par le Cyber Range Lab de l'Université de New South Wales (UNSW) dans le but d'améliorer la détection des cyberattaques dans les systèmes IoT et les infrastructures critiques.



> TON_IoT datasets			
	Nom ↑ ↓		Modifié ⓘ ↓
📁	Description_stats_datasets	✂	8 février 2024
📁	Processed_datasets	✂	8 février 2024
📁	Raw_datasets	✂	8 février 2024
📁	SecurityEvents_GroundTruth_datasets	✂	8 février 2024
📁	Train_Test_datasets	✂	8 février 2024
📄	ReadMe.pdf	✂ ... 🔗	8 février 2024

Figure 2 : TON IOT dataset folders available

Dans ce projet, nous utilisons la **version prétraitée du dataset**, où les données ont été nettoyées et converties sous forme de **caractéristiques numériques exploitables**. Ce dataset contient plusieurs types d'attaques :

- **Normal (trafic légitime)**
- **Backdoor**
- **DoS (Attaque par Déni de Service)**
- **Injection** (code injection, SQL injection, etc.)
- **Password** (attaques par force brute ou dictionnaire)
- **Scanning** (scan de ports et de vulnérabilités)
- **Ransomware**
- **XSS (Cross-Site Scripting)**
- **MITM (Man-in-the-Middle Attack)**

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA
1	src_ip	src_port	dst_ip	dst_port	proto	service	duration	src_bytes	dst_bytes	conn_state	missed_byte	src_pkts	dst_pkts	src_ip_bytes	dst_ip_bytes	dns_query	dns_class	dns_type	dns_rcode	dns_aa	dns_rd	dns_ra	dns_rejected	ssl_version	ssl_cipher	ssl_resumed	ssl_estab
2	192.168.1.37	4444	192.168.1.15	49178	tcp	-	290.371539	101568	2592	OTH	missed	0	108	108064	31	3832	-	0	0	0	-	-	-	-	-	-	-
3	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000102	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
4	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000148	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
5	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000113	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-
6	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.00013	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
7	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000403	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
8	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000137	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-
9	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000145	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
10	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.0002	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
11	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000009	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
12	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000113	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
13	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000204	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-
14	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000048	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-
15	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.0001	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
16	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000562	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
17	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000121	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
18	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000127	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-
19	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.00015	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
20	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000023	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-
21	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000127	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
22	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000145	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
23	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000127	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
24	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000155	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-
25	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.00019	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
26	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000094	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
27	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000095	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-
28	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000107	0	0	REJ	0	0	1	52	1	40	-	0	0	0	-	-	-	-	-	-	-
29	192.168.1.15	49180	192.168.1.37	8080	tcp	-	0.000105	0	0	REJ	0	0	1	48	1	40	-	0	0	0	-	-	-	-	-	-	-

Figure 3 : Aperçu sur les colonnes du dataset

Ce dataset est particulièrement pertinent pour la recherche en cybersécurité car il simule un environnement **réaliste** où les menaces sont représentatives des cyberattaques rencontrées dans des infrastructures IoT modernes.

Exploration:

L'exploration des données a pour objectif de découvrir et de comprendre davantage notre data, pour savoir ensuite la traiter à des fins analytiques ou pour la prédiction. Dans cette étape, on découvre les dimensions de notre jeu de données, les différents attributs, leurs types et leurs valeurs possibles...

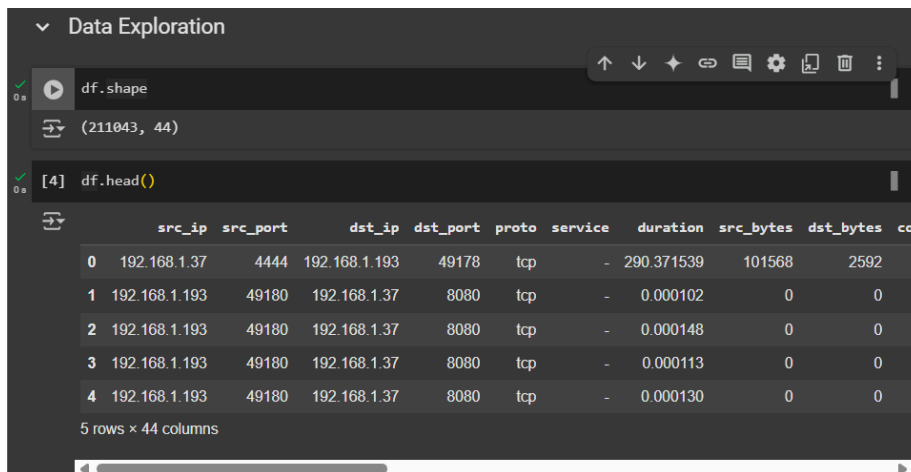


Figure 4 : Exploration des données

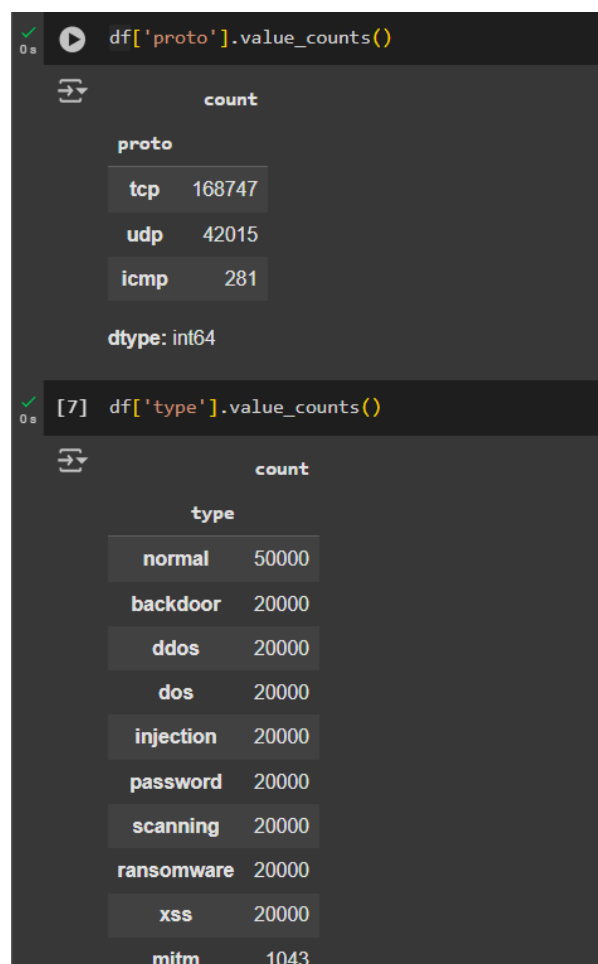


Figure 5 : Exploration des valeurs des colonnes

2. Préparation des données

Pour optimiser la qualité des résultats de notre modèle, il est essentiel d'optimiser au maximum la qualité des données qu'on va lui fournir en input. Pour cela, cette étape est essentielle.

On va éliminer les colonnes quasi-vides ainsi que les colonnes non numériques inutiles pour notre prédiction.

```
[10] # Drop non-numeric & mostly empty columns
df.drop(columns=["src_ip", "dst_ip", "dns_query", "dns_AA", "dns_RD", "dns_RA",
                 "dns_rejected", "ssl_version", "ssl_cipher", "ssl_resumed", "ssl_established",
                 "ssl_subject", "ssl_issuer", "http_trans_depth", "http_method", "http_uri",
                 "http_version", "http_request_body_len", "http_response_body_len", "http_status_cod",
                 "http_user_agent", "http_orig_mime_types", "http_resp_mime_types", "weird_name", "w
                 inplace=True)
```

Figure 6 : Suppression des colonnes inutiles

```
[9] #Checking missing values
df.isna().sum()
```

	0
src_ip	0
src_port	0
dst_ip	0
dst_port	0
proto	0
service	0

Figure 7 : Vérification des valeurs nulles

L'étape suivante est celle de l'encodage des données "encoding" qui est nécessaire pour transformer les variables catégorielles en valeurs numériques, car les algorithmes de Machine Learning, y compris les SVM, fonctionnent sur des valeurs numériques continues.

```

✓ [12] from sklearn.preprocessing import LabelEncoder

# Encode categorical columns
categorical_columns = ["proto", "service", "conn_state"]
encoder = LabelEncoder()

for col in categorical_columns:
    df[col] = encoder.fit_transform(df[col])

```

Figure 8 : Encodage des valeurs catégoriques

Pourquoi utiliser le Label Encoding ?

- Adapté aux variables catégorielles ordonnées ou lorsque le nombre de catégories est limité.
- Simple à mettre en œuvre.
- Compatible avec les SVM, qui interprètent les valeurs numériques comme des quantités continues.

Exemple de Label Encoding : Imaginons une variable "Type d'attaque" avec les catégories suivantes : ['DoS', 'Injection', 'Normal'].

Après l'encodage, nous obtenons :

- DoS : 0
- Injection : 1
- Normal : 2

Comparaison avec le One-Hot Encoding

Le One-Hot Encoding est une autre méthode couramment utilisée pour encoder des variables catégorielles. Elle consiste à créer une nouvelle colonne pour chaque catégorie avec une valeur binaire (0 ou 1).

Exemple de One-Hot Encoding pour "Type d'attaque" :

Dos	Injection	Normal
1	0	0
0	1	0
0	0	1

Avantages par rapport au Label Encoding :

- Évite l'ordre implicite entre les catégories (utile pour les algorithmes linéaires).
- Convient mieux aux modèles basés sur des distances euclidiennes.

Inconvénients :

- Génère un grand nombre de colonnes si le nombre de catégories est élevé.
- Moins adapté aux SVM pour des classes à grand nombre de catégories, car cela augmente la dimensionnalité des données.

Dans notre projet, nous avons privilégié le Label Encoding car il est plus efficace avec les SVM et les variables catégorielles ayant un nombre limité de classes.

Mise à l'échelle des données (Scaling)

Les SVM sont sensibles à la distance entre les vecteurs de données. Si les variables n'ont pas la même échelle, certaines caractéristiques pourraient dominer les autres, perturbant ainsi le modèle.

```
[15] from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
numeric_cols = df.columns.difference(["label", "type"]) # Exclude target columns
df[numeric_cols] = scaler.fit_transform(df[numeric_cols])

[16] df.head()
```

	src_port	dst_port	proto	service	duration	src_bytes	dst_bytes	conn_state	missed_bytes
0	-1.771488	4.482402	-0.492369	-0.673165	0.501064	-0.009157	-0.014214	-1.445311	-0.006544
1	0.545572	0.449865	-0.492369	-0.673165	-0.013650	-0.015099	-0.014358	-1.194698	-0.006544
2	0.545572	0.449865	-0.492369	-0.673165	-0.013650	-0.015099	-0.014358	-1.194698	-0.006544
3	0.545572	0.449865	-0.492369	-0.673165	-0.013650	-0.015099	-0.014358	-1.194698	-0.006544
4	0.545572	0.449865	-0.492369	-0.673165	-0.013650	-0.015099	-0.014358	-1.194698	-0.006544

Figure 9 : Normalisation des valeurs numériques

Pourquoi mettre à l'échelle ?

- Équilibrage des caractéristiques : Assurer qu'aucune caractéristique ne domine les autres en termes d'échelle.
- Accélération de la convergence : Les algorithmes basés sur les noyaux (kernel) sont plus efficaces sur des données normalisées.
- Amélioration des performances : Réduire le risque de mauvais ajustement (underfitting) ou de surapprentissage (overfitting).

Méthodes courantes de mise à l'échelle :

- **Standardisation (Standard Scaling)** : Transformation des données pour qu'elles aient une moyenne nulle (0) et un écart-type égal à 1.

$$X \text{ scaled} = (X - \text{moyenne}(X)) / \text{écart-type}(X)$$

- **Normalisation (Min-Max Scaling)** : Mise à l'échelle des données dans un intervalle [0, 1].

$$X \text{ scaled} = (X - X_{\min}) / X_{\max} - X_{\min}$$

Dans notre cas, la standardisation a été privilégiée pour assurer une bonne distribution des caractéristiques dans l'espace de décision des SVM.

3. Sélection des caractéristiques

La sélection des caractéristiques (*feature selection*) est une étape cruciale dans le processus de création de modèles de Machine Learning, y compris pour les SVM. Elle consiste à identifier et choisir les caractéristiques les plus pertinentes parmi l'ensemble des variables disponibles dans un jeu de données. Cette étape vise à améliorer les performances du modèle tout en réduisant la complexité computationnelle.

Pourquoi la sélection des caractéristiques est-elle importante ?

- Réduction de la dimensionnalité : Moins de caractéristiques signifie un espace de recherche réduit, ce qui peut accélérer l'apprentissage et diminuer les besoins en calcul.

- Amélioration des performances : Éliminer les caractéristiques non pertinentes ou redondantes peut améliorer la précision du modèle et éviter le surapprentissage (*overfitting*)
- Interprétabilité : Moins de caractéristiques facilitent l'interprétation du modèle, ce qui est crucial pour comprendre les facteurs influençant les prédictions.

Techniques de sélection des caractéristiques

A. Méthodes basées sur les filtres :

- Variance Threshold : Éliminer les caractéristiques avec une faible variance.
- Test du Khi-2 (Chi-Square Test) : Évaluer l'indépendance entre les caractéristiques et la variable cible pour les données catégorielles.
- Information mutuelle : Mesure de la dépendance entre les variables indépendantes et la variable cible.

B. Méthodes basées sur les wrappers :

- Recuit Simulé (Simulated Annealing) : Recherche d'un sous-ensemble optimal en minimisant une fonction de coût.
- Sélection récursive des caractéristiques (RFE) : Construction de modèles successifs pour éliminer les caractéristiques les moins significatives.

C. Méthodes basées sur les modèles :

- Utilisation des coefficients des modèles linéaires (régression logistique, SVM linéaire).
- Importance des caractéristiques basée sur les arbres de décision ou les forêts aléatoires (Random Forest).

```
from sklearn.feature_selection import mutual_info_classif

# Compute feature importance scores
feature_scores = mutual_info_classif(X, y)

# Create a DataFrame to visualize importance
feature_importance = pd.DataFrame({"Feature": X.columns, "Importance": feature_scores})
feature_importance = feature_importance.sort_values(by="Importance", ascending=False)

# Display the most important features
print(feature_importance.head(18))
```

	Feature	Importance
10	src_ip_bytes	0.422989
1	dst_port	0.421141
0	src_port	0.289102
12	dst_ip_bytes	0.252040
7	conn_state	0.210791
9	src_pkts	0.185936
5	src_bytes	0.184202
2	proto	0.171759
3	service	0.167915
4	duration	0.149848
14	dns_qtype	0.131331
6	dst_bytes	0.127734
11	dst_pkts	0.120118
13	dns_qclass	0.109847
15	dns_rcode	0.008913
8	missed_bytes	0.002680

Figure 10 : Application de la sélection des attributs

Pour les SVM, la sélection des caractéristiques est particulièrement importante car les SVM sont sensibles à la présence de caractéristiques non pertinentes.

En plus, la sélection des bonnes caractéristiques permet d'améliorer la séparation des classes dans l'espace de caractéristiques.

4. Entrainement du modèle

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

# Train model
svm_model = SVC(kernel="rbf", C=1, gamma="scale")
svm_model.fit(X_train, y_train)

# Predict
y_pred = svm_model.predict(X_test)
```

Figure 11 : Entrainement du modèle

SVC (Support Vector Classifier) est la classe utilisée pour créer un classificateur SVM dans scikit-learn.

Paramètres du modèle :

- **kernel="rbf"** : Utilisation du noyau gaussien RBF (Radial Basis Function), adapté pour les problèmes non linéaires.
- **C=1** : Paramètre de régularisation. Un C plus élevé donne la priorité à la classification correcte des points d'entraînement au détriment de la largeur de la marge. Ici, la valeur de 1 est un compromis entre marge large et erreur de classification.
- **gamma="scale"** : Définit automatiquement le paramètre γ en fonction du nombre de caractéristiques. Il contrôle l'influence d'un point d'entraînement.

L'entraînement est réalisé avec la méthode **.fit(X_train, y_train)**, où le modèle apprend à partir des données d'entraînement pour trouver l'hyperplan optimal séparant les différentes classes d'attaques IoT.

Après l'entraînement, nous effectuons des prédictions sur l'ensemble de test **X_test**. La sortie **y_pred** représente les types d'attaques prévus par le modèle.

5. Evaluation du modèle

Afin d'évaluer la performance d'un modèle entraîné, il y a plusieurs métriques à étudier:

Taux de Précision Globale (Accuracy)	<p>Le taux de précision est le rapport entre le nombre de prédictions correctes et le nombre total de prédictions. Elle est efficace lorsque les classes sont équilibrées.</p> $\text{Précision} = \frac{VP + VN}{VP + VN + FP + FN}$
Précision (Precision)	<p>La précision mesure la proportion de vraies attaques parmi les observations prédites comme des attaques. Elle est utile lorsqu'on souhaite limiter les faux positifs.</p> $\text{Précision} = \frac{VP}{VP + FP}$
Rappel (Recall) ou Sensibilité	<p>Le rappel évalue la proportion d'attaques détectées parmi toutes les attaques présentes dans les données. Il est crucial lorsqu'il est important de minimiser les faux négatifs.</p>

	$\text{Rappel} = \frac{VP}{VP + FN}$
F1-Score	<p>Le F1-Score est la moyenne harmonique entre la précision et le rappel. Il est particulièrement utile lorsque les classes sont déséquilibrées.</p> $\text{F1-Score} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$
Matrice de confusion	<p>La matrice de confusion est un tableau qui montre les performances du modèle en termes de VP, VN, FP, FN. Elle aide à analyser la distribution des erreurs.</p>
Courbe ROC et AUC	<ul style="list-style-type: none"> - Courbe ROC (Receiver Operating Characteristic) : Représente la sensibilité (rappel) en fonction du taux de faux positifs. - AUC (Area Under the Curve) : Aire sous la courbe ROC. Plus l'AUC est proche de 1, plus le modèle est performant.

```
# Evaluate
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

→ Accuracy: 0.9703854628159871
Classification Report:

	precision	recall	f1-score	support
0	0.97	0.90	0.94	10021
1	0.97	0.99	0.98	32188
accuracy			0.97	42209
macro avg	0.97	0.95	0.96	42209
weighted avg	0.97	0.97	0.97	42209

Figure 12 : Métriques d'évaluation de l'algorithme

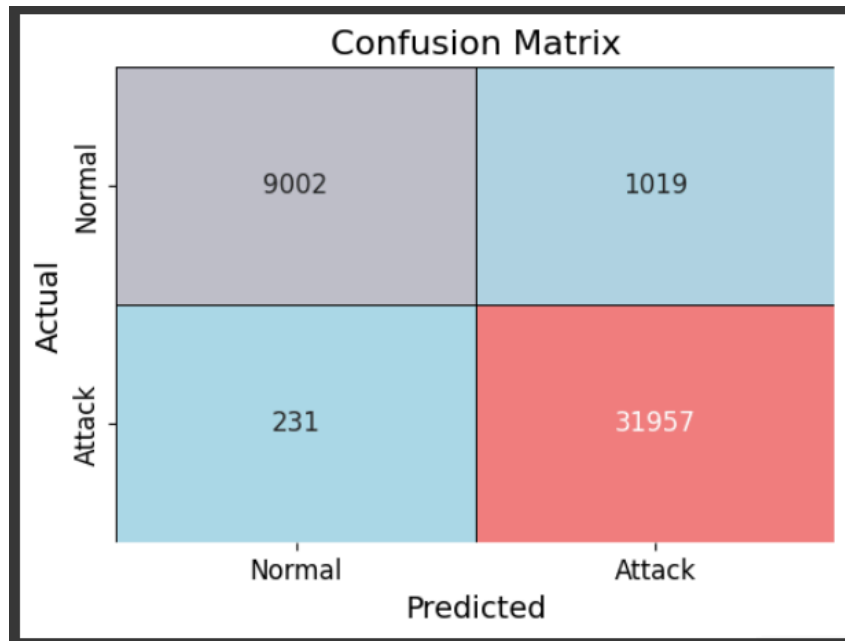


Figure 13 : Matrice de confusion obtenue

6. Améliorations: Optimisation du modèle

L'optimisation d'un modèle SVM est une étape cruciale pour améliorer ses performances, réduire les erreurs de classification et mieux généraliser sur des données nouvelles. Plusieurs approches peuvent être considérées pour affiner notre modèle de détection d'attaques IoT.

6.1 Optimisation des hyperparamètres

L'optimisation des hyperparamètres des SVM est essentielle pour ajuster le modèle de manière optimale. Deux méthodes principales ont été explorées pour cette optimisation :

- **Recherche en grille (Grid Search)** : méthode exhaustive testant toutes les combinaisons possibles d'hyperparamètres définis. Bien qu'efficace, elle peut être coûteuse en temps de calcul pour de grands ensembles de données.
- **Recherche aléatoire (Randomized Search)** : cette méthode explore de manière aléatoire un sous-ensemble des hyperparamètres définis, offrant un compromis entre efficacité de calcul et précision.

Contrairement à la Grid Search, Randomized Search est plus rapide et souvent tout aussi performante pour trouver des combinaisons optimales, il s'agit de la technique que nous allons utiliser pour ce projet.

6.2. Gestion du déséquilibre des classes

L'optimisation prend également en compte le déséquilibre des classes en utilisant la technique de **SMOTE** (Synthetic Minority Over-sampling Technique).

Cette méthode génère artificiellement de nouvelles instances de la classe minoritaire en interpolant les points proches de cette classe. Elle permet de créer un ensemble de données équilibré, améliorant la détection des attaques minoritaires et réduisant les erreurs.

```
from imblearn.over_sampling import SMOTE
from collections import Counter

# Affichage de la distribution initiale des classes
print("Distribution des classes avant équilibrage :", Counter(y_train))

# Application du suréchantillonnage SMOTE
smote = SMOTE(random_state=42)
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
```

Figure 14 : Équilibrage des données en utilisant la technique SMOTE

Conclusion

Ce projet de détection d'attaques IoT à l'aide des Machines à Vecteurs de Support (SVM) a permis d'explorer les capacités des algorithmes d'apprentissage supervisé pour résoudre des problématiques complexes de cybersécurité. À travers une approche théorique et pratique, nous avons pu comprendre le fonctionnement des SVM, leur capacité à gérer des données linéairement et non linéairement séparables, ainsi que leur potentiel à classer efficacement des données dans des environnements complexes comme les réseaux IoT.

L'implémentation pratique a montré que les SVM, combinés à une bonne préparation des données (encodage, normalisation, gestion du déséquilibre des classes), peuvent offrir des performances élevées en termes de précision, de rappel et de F1-Score. L'optimisation des hyperparamètres par Randomized Search a permis de raffiner davantage le modèle, réduisant les erreurs de classification et améliorant la robustesse des prédictions.

Cependant, certaines limitations subsistent. La sensibilité des SVM à des jeux de données volumineux et leur complexité computationnelle dans des espaces à haute dimension nécessitent des ajustements fins et une puissance de calcul suffisante. De plus, la gestion du déséquilibre des classes pourrait être approfondie à travers d'autres techniques telles que l'ensemble learning ou des approches hybrides combinant plusieurs modèles.

Perspectives futures

- Tester d'autres algorithmes de classification comme les réseaux de neurones profonds (CNN, RNN) pour explorer leur capacité à capturer les relations complexes entre les caractéristiques des attaques.
- Intégrer une détection en temps réel pour améliorer la réactivité des systèmes IoT face aux cyberattaques.
- Étendre l'analyse à d'autres jeux de données représentatifs d'environnements IoT variés pour évaluer la généralisation du modèle.

En résumé, ce projet a permis de mieux comprendre les enjeux de la cybersécurité dans les environnements IoT et d'appliquer efficacement les SVM pour la détection d'attaques, offrant ainsi une base solide pour des recherches et des développements futurs dans ce domaine.