TÉLÉCOM PARIS

SR2I 208

Machine learning based detection method for intrusion in CAN Bus

Réalisé PAR :
Ghalas Ihssane
EL Marchouk Kenza

Encadrant :
MR KHATOUN RIDA

2021/2022

Table des matières

1 Introduction

Les voitures autonomes interagissent entre elles et se connectent à l'internet. Et afin d'assurer la sécurité du conducteur et des passagers, les véhicules sont équipés de multiples connexions à l'internet, comme le système de bus CAN. Cela entraîne des problèmes et des vulnérabilités en raison de la possibilité accrue d'accès à distance au véhicule, puisque le système de bus CAN ne prend pas en charge les mécanismes d'authentification et d'autorisation. En effet, les messages CAN sont diffusés sans caractéristiques de sécurité de base. Par conséquent, il est facile d'attaquer le système de bus CAN de plusieurs façons, notamment par déni de service (DoS), fuzzing et spoofing.
Il est donc impératif de concevoir des méthodes permettant de protéger les voitures modernes contre les attaques susmentionnées [1].

Dans ce projet, nous utiliserons différentes méthodes d'apprentissage automatique pour détecter les intrusions, notamment les attaques de type fuzzing, spoofing, replay et fooding.

2 CAN bus

2.1 Définition

Bus CAN (Controller Area Network) : c'est un protocole de communication normalisé par l'Organisation internationale de normalisation (ISO), il est utilisé pour gérer le flux d'informations entre les unités de contrôle électronique (ECU) d'une voiture, il a été développé dans les années 1980 par Robert Bosch pour réduire la complexité et le poids du câblage dans les véhicules comme le montre la figure 1, la dernière version du protocole CAN 2.0 a été publiée en 1991.

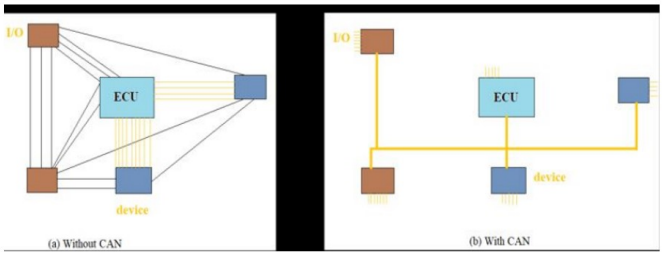


Figure 1 – Réduction du câblage avec CAN bu

Les différents ECM ou ECU communiquent entre eux par paires de fils. Il peut s'agir de la suspension, des systèmes de freinage ou de la climatisation.

2.2 Fonctionnemnt

Le protocole CAN fonctionne avec la communication réseau peer to peer, et la communication entre les ECU se fait via la transmission et la réception des messages qu'ils écrivent sur la trame CAN, ce message contient un ID unique pour identifier et classer les messages selon l'ordre de priorité.

1. Article IEEE, `https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9216166`

Quand plusieurs nœuds essaient de diffuser un message simultanément sur le bus CAN, la priorité sera donnée au message avec ID d'arbitrage le plus bas.
Le format du message :

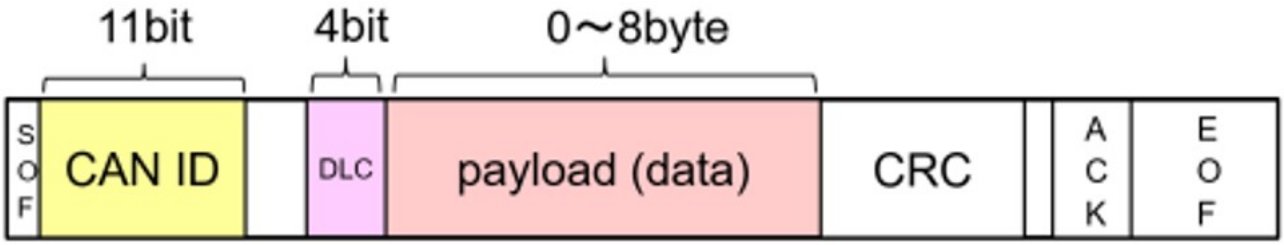


FIGURE 2 – Format du message CAN

Start of Frame (SOF) : utiliser pour synchroniser et notifier tous les nœuds.
ID CAN : est un champ d'arbitrage utiliser pour identifier et prioriser les messages.
Data Lenght Code (DLC) : indique la longueur d'octet du champ de donnée, Il est compris entre 0 et 8 octet.
Payload data : contient les données qui sont qui sont exécutées par les ECU.
CRC : utiliser pour détecter les erreurs de transmission du message.
ACK : champ d'acquittement, il est utilisé pour que le noud récepteur confirme la bonne réception du message, en cas d'erreur de transmission, l'émetteur peut envoyer le message CAN à nouveau.
End of frame (fin de trame) : ce champ indique la fin du message CAN.
Exemple du message bus CAN pour Toyota :

```
IDH: 07, IDL: E0, Len: 08, Data: 02 27 01 00 00 00 00 00
IDH: 07, IDL: E8, Len: 08, Data: 06 67 01 01 BB 8E 55 00
IDH: 07, IDL: E0, Len: 08, Data: 06 27 02 01 DB EE 55 00
IDH: 07, IDL: E8, Len: 08, Data: 02 67 02 00 00 00 00 00
```

FIGURE 3 – Exemple du message bus CAN

2.3 Les couches du bus CAN :

Le protocole CAN contient deux couches du modèle ISO, la couche liaison de données décrite par la norme ISO 11898-1 et la couche physique décrite par la norme ISO 11898-2.
Ce figure résume les couches ISO utilisées par le protocole CAN :
[2]

2. Gestion de bus CAN, https://www.electronique-mixte.fr/wp-content/uploads/2018/07/Formation-Interface-communication-43.pdf

<table>
<tr><td>Spécifications</td><td colspan="2">Couche OSI</td><td>Implémentation</td></tr>
<tr><td>à spécifier par l'utilisateur</td><td colspan="2">Couche application</td><td>Software embarqué ou non</td></tr>
<tr><td rowspan="5">Spécifications du protocole CAN</td><td rowspan="2">Couche de communication de données</td><td>Logical Link Control</td><td rowspan="3">On-chip hardware</td></tr>
<tr><td>Medium Access Control</td></tr>
<tr><td rowspan="3">Couche physique</td><td>Physical Signaling</td></tr>
<tr><td>Physical Medium Attachment</td></tr>
<tr><td>Medium Dependent Interface</td><td>Off-chip hardware</td></tr>
<tr><td></td><td colspan="2">Transmission Medium</td></tr>
</table>

FIGURE 4 – Couches OSI utilisées par le protocole CAN.

La couche liason : Cette couche est utilisée pour établir la connexion entre les couches et la maintenir par la correction des erreurs.
Elle contient des sous-couchess :

- LLC (Logical Link Control), elle permet d'effectuer le filtrage des messages, les notifications des surcharges, la récupération des erreurs.
- MAC (Medium Access Control) permet d'effectuer l'arbitrage, la détection des erreurs et leur signalisation.

La couche physique : cette couche est utilisée pour transmettre les bits via un support de transmission physique.
Les deux normes de la couche physique :
National Instruments implémente deux standards de couche physique : CAN haut débit (High-speed)et CAN bas débit(Low speed).

High-Speed CAN	— Les vitesses de transmission allant de 40 kbit/s à 1 Mbit/s(500 kbit/s est le débit le plus courant), selon la longueur du câble. — la norme la plus utilisée, puisque elle établit simplement une connexion entre les appareils par câble. — parmis les standards qui spécifié par : ISO-11898-1 et ISO-11783-2.
Low-Speed	— Les vitesses de transmission allant de 40 Kbit/s à 125 Kbits/sec. — Elle permet à la communication du bus CAN de se poursuivre en cas de défaillance du câblage (CAN tolérant aux défaillances). — Dans ce réseau chaque dispositif possède sa propre terminaison.

2.4 Les points faibles du potocole CAN bus

Confidentialité	le protocole CAN n'implémente pas le chiffrement des messages ils sont donc transmis en clair
Authentification	le protocole CAN ne valide pas l'origine d'un message et il permet à des nœuds non autorisés de rejoindre le réseau de communication
Intégrité	CAN possède le champ de contrôle CRC qui détecte les erreurs pendant la communication, mais si les données modifiées provenant de l'extérieur, cette modification ne peut pas être détectée
La disponibilité	quand lorsqu'un nœud avec une priorité plus élevée (ID le plus bas) transmet tout le temps, le bus ne peut pas être accessible par les autres nœuds.

3 La base de données

3.1 Description générale de la base de données :

La base de données utilisée dans ce projet a été publiée à la suite de la compétition de cybersécurité "Car Hacking : Attack Defense Challenge" en 2020. Ce challenge avait pour but de développer des techniques d' attaque et des outils pour les détecter à travers le CAN Bus du véhicule Hyundai Avante CN7.

La base de données consiste en une description de CAN Bus , collectée en deux modes : le mode stationnaire et le mode dynamique. Elle contient une description de chaque message, avec un label indiquant si le message est normal ou c'est plutôt une attaque avec son type.

Les attributs utilisés pour decrire les messages sont les suivants :

- Timestamp
- Arbitration_ID
- DLC
- Data
- Class
- SubClass

Ci-dessous quelques lignes de la base de données (Pre_train_S_1.csv) :

	Timestamp	Arbitration_ID	DLC	Data	Class	SubClass
327112	1.597759e+09	2B0	8	5D CB 02 76 75 A8 88 BE	Attack	Fuzzing
327113	1.597759e+09	130	8	F8 7F 54 80 00 00 05 40	Normal	Normal
327114	1.597759e+09	140	8	EC 7F 00 6E 20 00 05 7A	Normal	Normal
327115	1.597759e+09	251	8	F8 03 23 9C 00 FA 07 80	Normal	Normal
327116	1.597759e+09	2B0	6	D1 FF 00 07 83 76	Normal	Normal
327117	1.597759e+09	391	8	C3 A8 3C F2 87 E1 DE 0C	Attack	Fuzzing
327118	1.597759e+09	260	8	05 31 02 30 00 BD 4D 27	Normal	Normal
327119	1.597759e+09	7D4	8	69 41 D1 1B 3F BD CB 9C	Attack	Fuzzing

FIGURE 5 – Quelques lignes de la base de données

La base de données est constituée de 9 fichiers CSV : 8 en tant que "Preliminary" (6 en training et 2 en submission) et un seul fichier "Final".

3.1.1 Preliminary :

'Preliminary' est constitué de 6 bases de données destinées au training et 2 pour Submission. Les données ont été collectées pendant les deux modes stationnaire et dynamique du véhicule. Les deux bases de données Pre_train_D_0.csv et Pre_train_S_0.csv ne contiennet que des messages normaux.

La distribution des messages dans les différentes datasets est la suivantes, avec 0 pour un message normal et 1 pour une tentative d'attaque :

```
Preliminary , training
Number of data points in  Pre_train_D_0.csv  :  179346
0 179346
        --------**---------
Number of data points in  Pre_train_D_1.csv  :  806390
0 733752
1 72638
        --------**---------
Number of data points in  Pre_train_D_2.csv  :  889395
0 811532
1 77863
        --------**---------
Number of data points in  Pre_train_S_0.csv  :  180686
0 180686
        --------**---------
Number of data points in  Pre_train_S_1.csv  :  799292
0 727749
1 71543
        --------**---------
Number of data points in  Pre_train_S_2.csv  :  817042
0 739678
1 77364
        --------**---------
The total number of data points : 3672151
Normal :  3372743
Attack :  299408
```

Figure 4 : Nombre de messages normaux et anormaux dans chaque dataset de Preliminary -> Training

```
Preliminary , submission
Number of data points in  Pre_submit_D.csv  :  2000733
0 1799046
1 201687
         --------**---------
Number of data points in  Pre_submit_S.csv  :  1751313
0 1559164
1 192149
         --------**---------
The total number of data points : 3752046
Normal :  3358210
Attack :  393836
```

Figure 5 : Nombre de messages normaux et anormaux dans chaque dataset de Preliminary -> Submission

3.1.2 Final

'Final' est formé d'une seule dataset : Fin_host_session_submit_S.csv . Cette base de données cotient 1270310 lignes (messages) collectées qu'en mode stationnaire. La répartition des messages est la suivante :

```
Final
Number of data points in  Fin_host_session_submit_S.csv  :  1257303
0 1077305
1 179998
```

Figure 6 : Nombre de messages normaux et anormaux dans Final

3.2 Les attributs :

Les messages utilisés dans les bases de données sont décrits par 4 attributs (Timestamp , Arbitration_ID , DLC et Data). Et la nature du message est décrite à travers deux colonnes : Class (Normal ou Attack) et SubClass (Normal si le message est normal, et le type de l'attaque s'il s'agit d'une tentative d'attaque).

3.2.1 Les types des messages :

Le type de message est décrit dans la base de données à travers deux colonnes ; la première colonne prend deux valeurs : **Normal** et **Attack** et la deuxième colonne est plus spécifique, elle prend 5 valeurs : Normal si le message est normal et le type d'attaque sinon.

4 types d'attaque sont détectés :**Flooding, Spoofing, Replay** et **Fuzzing**.

Flooding :

L'objectif de cette attaque est d'empêcher le bon fonctionnement d'un service, serveur ou une infrastructure, en le rendant indisponible.

Le principe est le suivant :
L'attaquant envoie un volume assez élevé de trafic à un système afin qu'il ne puisse pas atteindre les demandes légitimes et ne puisse pas gérer la charge. Par exemple, une attaque ICMP Flooding se produit lorsqu'un système reçoit plusieurs ping ICMP et doit utiliser toutes ses ressources pour répondre.

Spoofing :

ou attaque par usurpation d'identité. Il s'agit de masquer sa propre identité et se faire passer pour quelqu'un d'autre. les attaques spoofing les plus connues sont :

- IP address spoofing : consiste à envoyer des packets qui ont pour IP source une adresse qui n'est pas attribuée à la machine qui les a émis.
- Email spoofing : envoie d'un virus par email à partir d'une adresse mail qui existe et après ouverture du mail, l'attaquant peut extraire les données qu'il souhaite avoir.

Replay :

Replay attack, ou attaque par rejeu est une attaque consistant à intercepter les paquets de données et à les rejouer (les retransmettre sans aucune modification) au destinataire.
Ce type d'attaque peut être utilisé pour accéder aux informations sur un réseau protégé, ou pour dupliquer les transactions …

Fuzzing :

Fuzzing attack est un processus utilisé généralement pour trouver les vulnérabilités dans des applications. Il s'agit d'injecter une très grande quantité de données aléatoires dans le code source.
C'est une méthode rapide utilisée même pour tester la sécurité des logiciels, elle ne demande aucune connaissance du fonctionnement du système pour l'écriture du code.

4 Les méthodes de machine learning utilisées

4.1 K-means

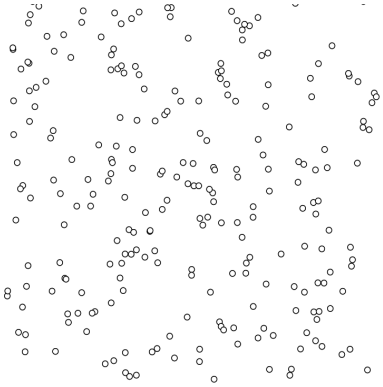
C'est un algorithme d'apprentissage non supervisé qui vise à découvrir une structure dans une base de données, c'est une méthode de partitionnement de données. À partir d'un ensemble de points et d'un entier k (le nombre d'ensemble qu'on veut en sortie), l'algorithme donne en sortie une division en k ensembles des points de data, appelés clusters, en minimisant une certaine fonction : la somme des distances entre les points d'un cluster et le centre de ce cluster (défini par la moyenne des points).

Mathématiquement parlant, le problème se traduit par : on a n points $(x_1, x_2, ..., x_n)$ qu'on veut diviser en k ensembles $(S_1, ..., S_k)$ on minimisant la fonction suivante :

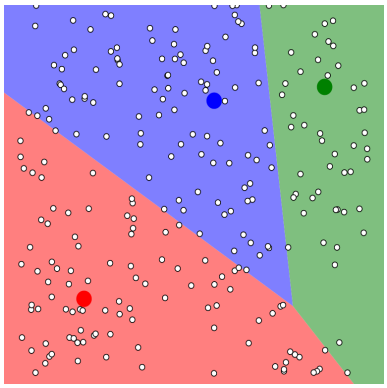
$$\arg \min_{(S_1,...,S_n)} \sum_{i=1}^{k} \sum_{j \in S_i} ||x_j - \mu_j||^2$$

On commence par initialiser k centroids au hasard (les barycentres des clusters), généralement parmi les points de data et on calcule la distance des points aux centroids, puis on attribue chaque point au centroid qui lui est proche, on actualise le centroid et on refait la même chose jusqu'à ce que cela ne varie pas, ou un nombre d'itérations fixé au début est épuisé.

Par exemple, on veut partitionner l'ensemble des points suivants en 3 clusters :



On commence par initialiser les centroids au hasard, généralement on les prend parmi les points de la data, comme indiqué ci-dessous :



Puis on attribut chaque points au cluster du centroid qui lui est proche et on calcule le nouveau centroid qui correspond au barycentre des points du cluster :

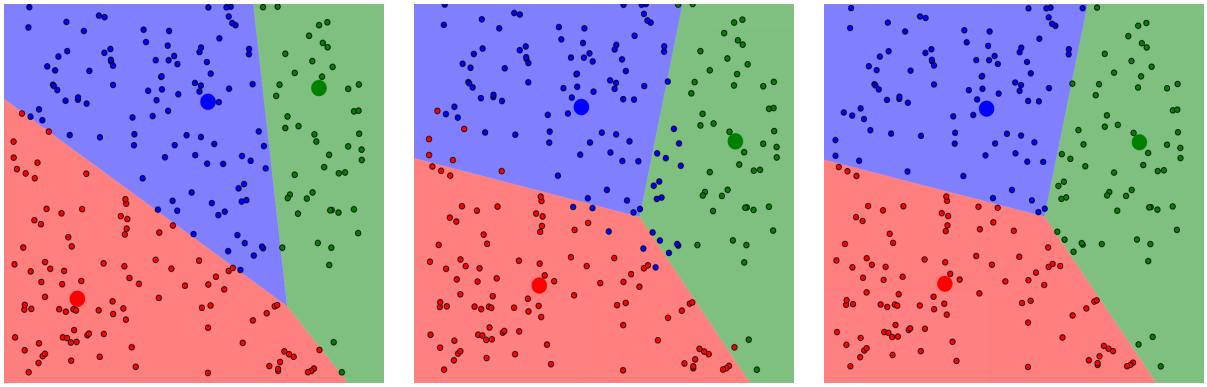


FIGURE 6 – Mise à jour dans l'algorithme K-means

On refait la même chose jusqu'à ce que les barycentres ne changent plus, le résultat obtenu est le suivant :

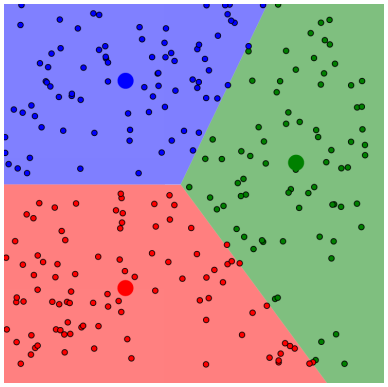


FIGURE 7 – Résultat final de K-means avec K=3

4.2 Arbre de décision

Arbre de décision fait partie des algorithmes supervisés, il permet de prédire une valeur réelle (régression) ou une classe (classification).

Ses avantages :

- Simple à interpréter et à comprendre
- La relation non linéaire entre les données n'affecte pas la performance du modèle
- N'est pas très coûteux en terme de temps de calcul.

Ses Inconvénients :

- Risque de sur-apprentissage
- Faible performance : cas de petite variance

Comment construire un arbre de décision :

L'algirthme d'arbre de décision prend tous les individus et cherche la variable qui sépare le mieux les données. son choix de cette variable est basé sur des critères d'impureté.

Cross-entropy:

$$H(\mathcal{S}) = -\sum_{k=1}^{C} p_k(\mathcal{S}) \log p_k(\mathcal{S})$$

Gini index

$$H(\mathcal{S}) = 1 - \sum_{k=1}^{C} p_k(\mathcal{S})^2$$

FIGURE 8 – Les critères d'impureté les plus utilisés

Dans cette exemple de classification on va se baser sur le critère de Gini index.

On prédire s'il une personne aime la glace selon son age, s'il aime ou pas popcorn et soda.

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

FIGURE 9 – Data

L'algorithme va évaluer tous les variables

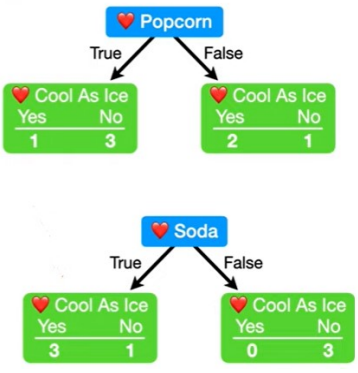


FIGURE 10 – Evaluation des variables

Mais le processus est différent pour variable age, puisque c'est une variable non binaire, l'algorithme va calculer âge moyen entre les personnes voisines.

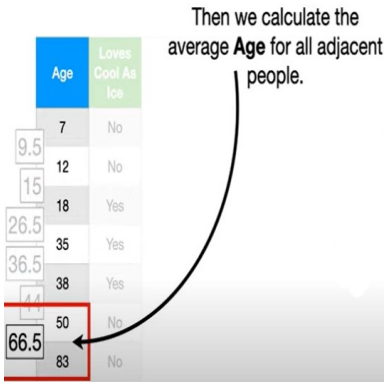


FIGURE 11 – Calcule des moyens

Après algorithme va calculer gini index de chaque age, celui qui a le petite index est le mielleur.

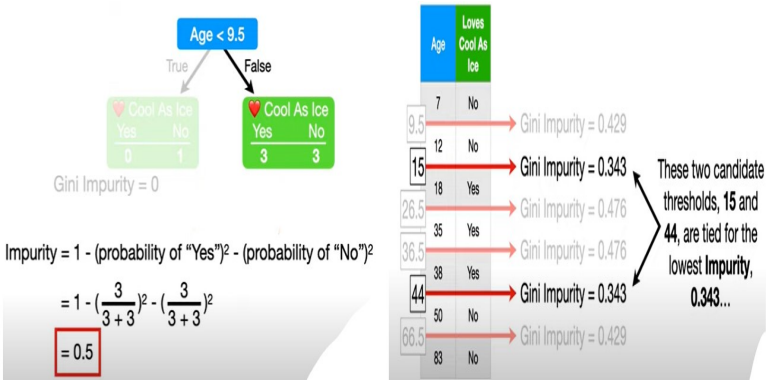


FIGURE 12 – Calcule Gini index

Ensuite, il va comparer Gini index entre tous les variables, et celui qui a index le plus petite sera dans la feuille racine.
Dans notre cas la variable soda qui a le plus petite index.

FIGURE 13 – Choix d'une variable

On se concentre sur la feuille à gauche car, les données ne sont pas encore bien classés.

L'algorithme va alors répéter meme processus d'évaluation pour choisir deuxième meilleur index.

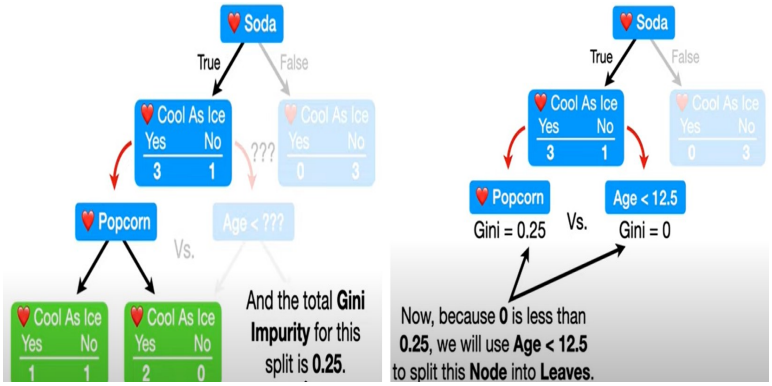


FIGURE 14 – Choix d'une deuxième variable

En fin, la décision sera basée selon la classe majoritaire de chaque feuille.

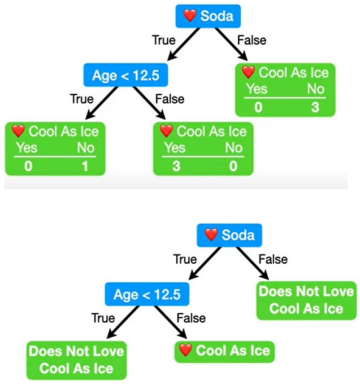


FIGURE 15 – Prédiction des classes

Comment classifier un nouveau data :

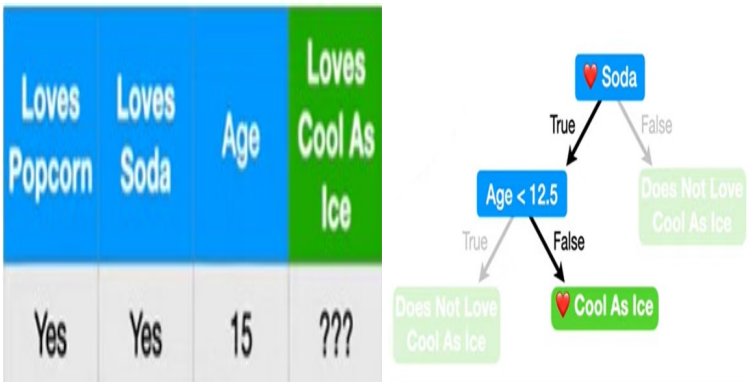


FIGURE 16 – Classifier nouveau data

4.3 K plus proches voisins

k-nearest neighbors ou K-NN est un algorithme d'apprentissage supervisé. On dispose d'une base de données de N entrées avec leurs sorties. Étant donné une nouvelle valeur d'entrée x qu'on veut labelliser, on ne prend pas en compte que les k points dont l'entrée est la plus proche de x . On attribue à l'entrée x la classe la plus représentée parmi les k proches voisins dans un problème de classification.

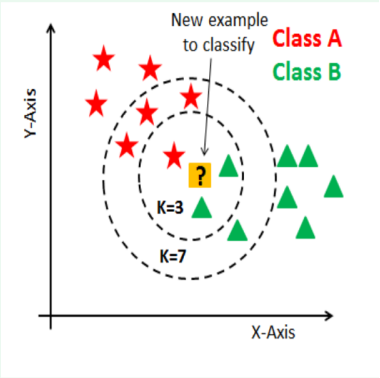


FIGURE 17 – K-NN pour classifier un nouveau point

On peut voir dans la figure ci-dessus que le choix de K est très important, pour K=3, on va classifier la nouvelle valeur d'entrée avec les triangles, mais si on prend K=7 , c'est plutôt classifier parmi les étoiles. Plus la valeur de K est petite, plus le bruit dans la data aura un effet majeur, et une valeur de K grande donne un résultat plus précis, mais très couteux en terme de calcul.
Une attention très particulière doit être accordée au choix du paramètre K.

On peut résoudre ce problème en appliquant l'algorithme avec différentes valeurs de K et en traçant la courbe de l'erreur en fonction de K et puis on choisit la valeur de K qui minimise l'erreur.

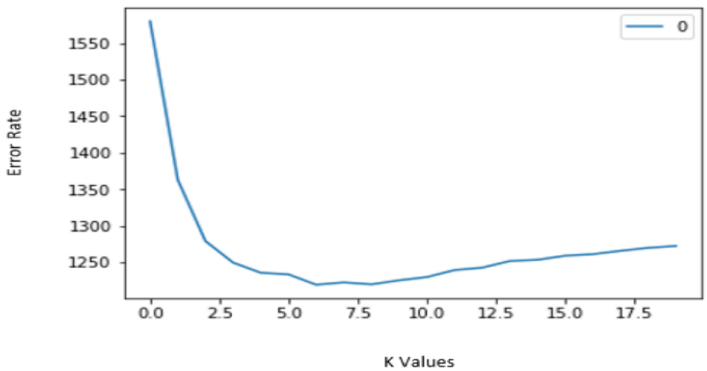


FIGURE 18 – L'erreur en fonction de la valeur de K

Les avantages majeures de K-NN sont :

- Très facile de l'implémenter et de le comprendre.
- Pas de phase d'apprentissage.
- Très approprié pour un petit ensemble de données.

Mais cette méthode a aussi plusieurs inconvénients :

- L'algorithme ne marche pas quand les données ne sont pas normalisées.
- Difficile de choisir la valeur de K.
- Il est sensible aux valeurs aberrantes et aux valeurs manquantes.

5 Résultats

5.1 Preprocessing des données

Avant d'appliquer les algorithmes de machine learning pour classifier les données, on commence d'abord par le pré-traitement des données.

Pour cela, on crée deux fonctions différentes, qui à partir du nom du fichier excel renvoient une matrice X, identique dans les deux fonctions et qui est une transformation des données décrivant le message (on transforme data et Arbitration_ID de hexadécimal en entier) , et y qui contient les labels (class pour la première fonction et subclass pour la seconde).

5.2 Cas de classification en classe normale et anormale (tous types d'attaques confondus)

Pour pouvoir comparer les scores des différents algorithmes, on va entraîner K-NN et Decision Trees sur la combinaison des bases de données Pre_train_D_1.csv , Pre_train_S_1.csv , Pre_train_D_2.csv et Pre_train_S_2.csv . On va aussi appliquer k-means et on va tester ces deux modèles sur la combinaison de Pre_submit_S.csv , Pre_submit_D.csv et Fin_host_session_submit_S.csv.

La distribution des messages entre normaux (label = 0) et anormaux (label = 1) dans ces données de train et test est la suivante :

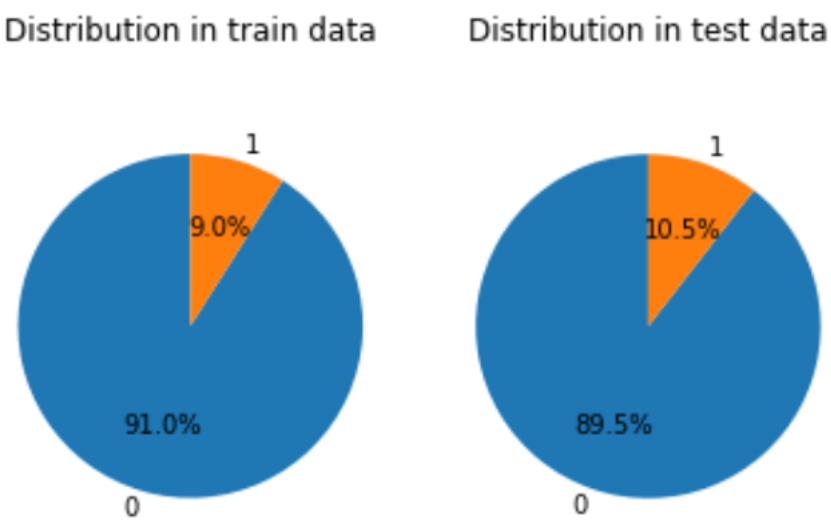


FIGURE 19 – Distribution des messages dans train et test datasets

Comme affiché ci-dessus, la classe normale est très majoritaire dans les deux datasets. Pour résoudre ce problème, on fait un undersampling pour avoir le même nombres de samples pour les deux classes :

```
undersample = RandomUnderSampler(sampling_strategy=1)
X_test, y_test = undersample.fit_resample(X_test, y_test)

X_train, y_train = undersample.fit_resample(X_train, y_train)
```

FIGURE 20 – Undersampling

Et pour un meilleur fonctionnnement des algorithmes, on fait le rescaling des données.

5.2.1 K-means

On va appliquer K-means sur les données de test et calculer le score et f1.

Avant d'appliquer l'algorithme, on applique le PCA (Principal Component Analysis) pour pouvoir visualiser le résultat de K-means.

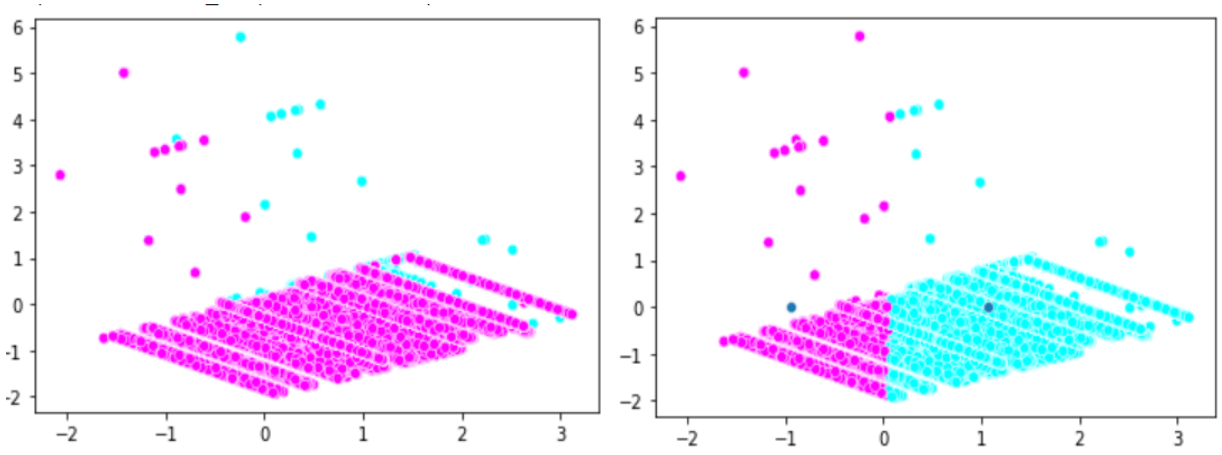


FIGURE 21 – test Data à gauche et le résultat de K-means sur cette dataset à droite

5.2.2 K-NN et Decision Trees

Pour ces deux modèles, on va les entraîner sur train data (Pre_train_D_1.csv , Pre_train_S_1.csv , Pre_train_D_2.csv et Pre_train_S_2.csv) et puis on va les tester sur test data pour pouvoir comparer les résultats.

Pour l'algorithme K-NN, il faut déterminer la valeur la plus adéquate de K qui donne les meilleurs résultats.
Pour cela, on a procédé par le Grid Search : on a d'abord calculé le score et la matrice de confusion pour différentes valeurs de k (3, 5, 7 et 9), les résultats sont affichés ci-dessous :

Valeur de k	Score	f1 score
$k = 3$	0.752	0.678
$k = 5$	0.751	0.677
$k = 7$	0.751	0.676
$k = 9$	0.750	0.675

La valeur de k qui donne le meilleur score aussi bien que la meilleure valeur de f1 score est $k = 3$, c'est la valeur qu'on va utiliser par la suite.

5.2.3 Comparaison des résultats

Pour pouvoir bien comparer les résultats des différentes méthodes, on a tracé les courbes représentant l'accuracy et f1-score pour les différents modèles. Le graphe est donné ci-dessous :

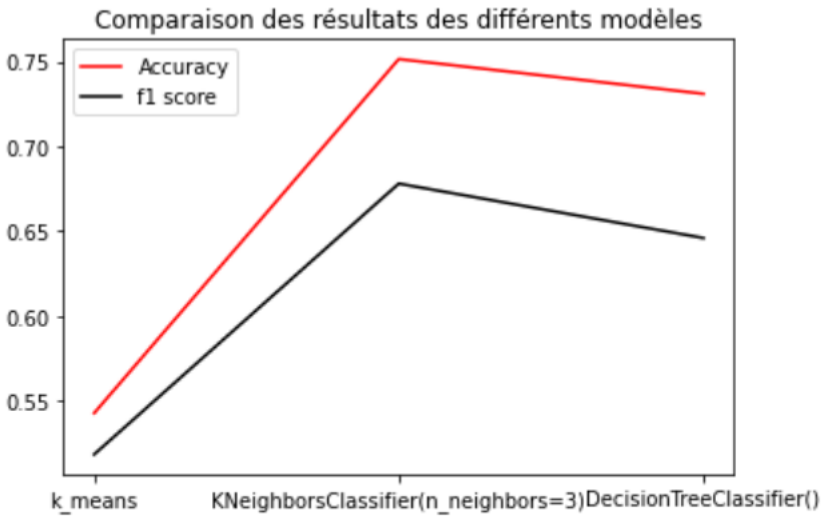


Figure 22 – Comparaison des résultats

D'après la courbe ci-dessus, on remarque bien que K-NN donne le meilleur résultat, en terme de score et de f1 aussi (grandeur très importante dans ce cas puisqu'on ne veut pas trop tolérer les erreurs sur les attaques)

Decision Tree Classifier reste aussi acceptable. Mais pour K-means, les résultats sont un peu faibles, le score dépasse à peine 0.5 , ce qui veut dire que c'est légérement mieux qu'un random classifier.

5.3 Cas de classification en 5 classes : normal, Flooding, Spoofing, Replay et Fuzzing

On utilise les mêmes datasets de train et de test que dans le cas de deux classes.

Le pourcentage de la classe normale est le même, mais la distribution des attaques dans la classe anormale est la suivante :

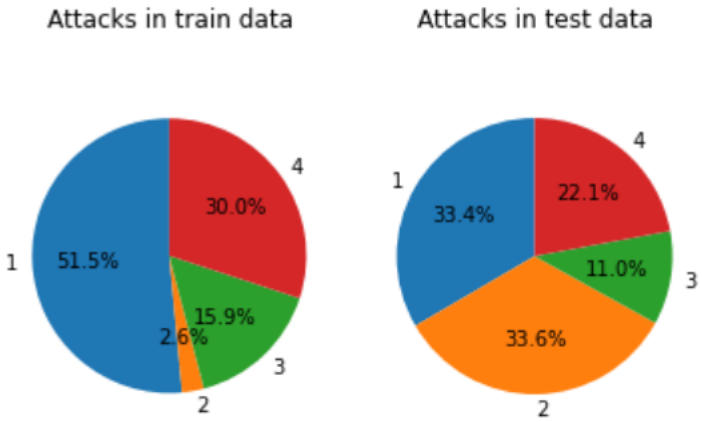


Figure 23 – Distribution des attaques

Où :

- 1 = Flooding
- 2 = Spoofing
- 3 = Replay
- 4 = Fuzzing

5.3.1 K-means

On applique K-means comme dans le cas précédent mais avec un nombre de clusters égal à 5.

Les résultats sont affichés ci-dessous :

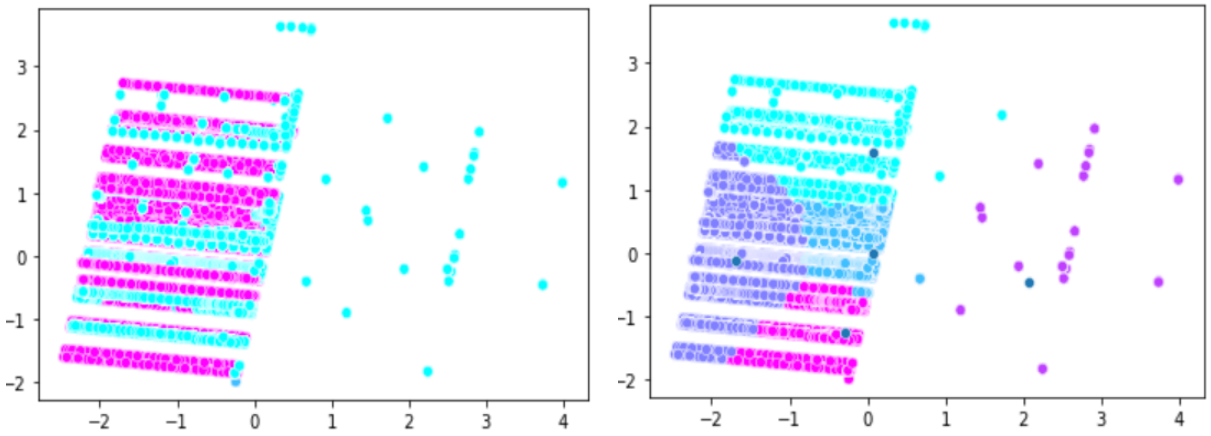


FIGURE 24 – test Data à gauche et le résultat de K-means sur cette dataset à droite

On remarque déjà que dans la représentation de dataset à gauche, les types d'attaque minoritaires ne sont pas représentés (dissimulés par les types majoritaires), donc à première vue, ce modèle ne permet pas d'avoir une bonne classification des messages en des 5 classes.

Pour avoir une équivalence entre les labels dans les deux classifications, on a attribué à chaque cluster le label de la classe majoritaire des éléments qui lui appartiennent.

5.3.2 K-NN et Decision Trees

On a commencé comme dans le cas de deux classes par déterminer la meilleure valeur de K pour le modèle K-NN. La valeur de K optimale est toujours $K = 3$.

5.3.3 Comparaison des résultats

Pour pouvoir comparer les résultats, ci-dessous le tracé des différentes valeurs de accuracy et f1 score pour les différentes classes :

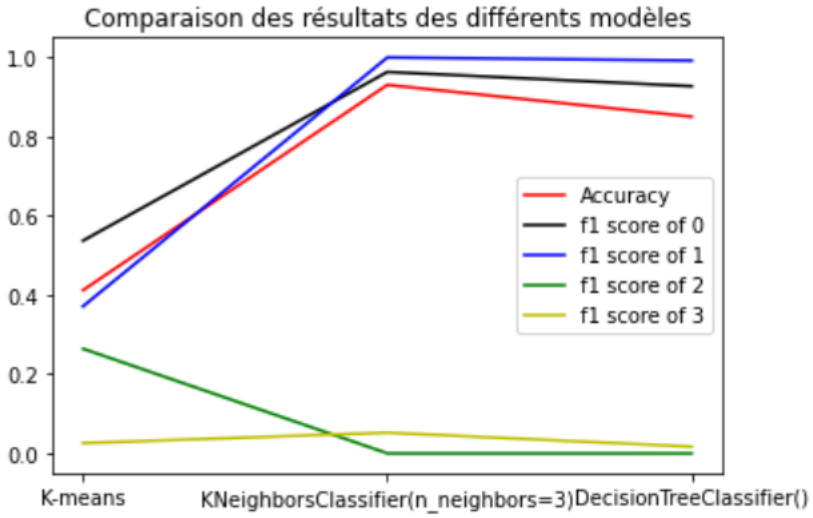


FIGURE 25 – Comparaison des résultats pour le cas de 5 classes

Pour le score, c'est toujours K-NN qui l'emporte avec un score de 0.93. Mais ce n'est pas suffisant. Pour les valeurs de f1-score, on remarque bien que pour la classe normale et la classe labellisée 1, la valeurs de f1-score est élevée pour K-NN et Decision Tree Classifier. Mais pour les classes 2 et 3, les valeurs sont très petites. Cela revient au fait que le nombre de samples qu'on a pour ces classes est très minoritaire, ce qui fait qu'on n'arrive pas à bien les détecter.

5.4 Conlusion :

Dans ce projet, nous avons appliqué différents modèles de machine Learning pour détecter les intrusions sur bus CAN.
Ce projet nous a permis de découvrir le protocole bus CAN : l'architecture, la trame CAN, les attaques spoofing, flooding, fuzzing et replay.

Cependant, nous avons rencontré certaines difficultés :

- La taille des données est très grande, ce qui fait que l'exécution de certains modèles de machine Learning sont assez coûteux en termes de temps d'exécution.
- La sélection des modèles : on a essayé au début avec plusieurs modèles, après on en a sélectionné que 3 modèles. Pour faire le meilleur choix, il ne s'agit pas juste de prendre les modèles donnant un score élevé, mais il faut voir d'autres paramètres (f1, confusion_matrix ...).
- Le problème de classe majoritaire : ce qui produit une accuracy biaisée, donc il faut prendre en compte d'autres critères pour bien évaluer le modèlele, comme F1 score.

Les résultats montrent que les outils de Machine Learning ne sont pas assez fiables pour détecter les intrusions dans le bus CAN puisque les scores et les valeurs de F1 ne sont pas assez élevés pour une meilleure détection de tentative d'attaque.

Bibliographie :

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9216166
https://odr.chalmers.se/bitstream/20.500.12380/302485/1/CSE%2021-29%20Aryan%20S%C3%B6derberg.pdf
https://iot.bzh/download/public/2020/IDS_CAN.pdf
Article Towards a Lightweight Intrusion Detection Framework for In-Vehicle Networks de Dheeraj Basavaraj and Shahab Tayeb
https://www.naftaliharris.com/blog/visualizing-k-means-clustering/
https://www.analyticsvidhya.com/blog/2022/01/introduction-to-knn-algor