Entrenament d'un arbre de decisió per a la detecció d'atacs distribuïts de denegació de servei (DDoS). Una aproximació.

Gerard Farràs Ballabriga gfarrasb@uoc.edu
Director: Enric Hernández Jiménez ehernandezj@uoc.edu

MISTIC UOC

December 29, 2019

Abstract

In the information and knowledge society of the 21st century, the information of each organization is highly valuable. The possibility of uninterrupted and secure use of information systems becomes an essential task for public and private organizations around the world. Within this context, the operational continuity of all information systems must be guaranteed. Distributed denial of service attacks (DDoS) are attacks that, using multiple techniques, impair the continued use of information systems. It is necessary, therefore, to find tools that mitigate these types of attacks. There are currently software-based mechanisms that ensure information systems, but work must be done to implement them with low-cost computational techniques and also ensure that they are updated day by day. A field currently in search and with a lot of future projection involves using artificial intelligence and machine learning techniques for real-time detection of what an attack could be. In this work, an implementation of a decision tree is carried out using machine learning techniques on a set of network traffic data (dataset) with DDoS attacks.

Keywords: Cyber Security dataset, DDoS, Machine Learning, Decision trees.

Document escrit en LaTeX

Contents

1 Introducció

La ciberseguretat consisteix en el conjunt de prevencions de seguretat que proveeixen confidencialitat, integritat i disponibilitat de les dades. La ciberseguretat és una àrea de recerca contemporània i és que totes les operacions realitzades des dels governs, empreses de caire comercial, financer o tecnològic empren ingents volums de dades emmagatzemades en sistemes d'informació digitals.

L'increment en el nombre i també la sofisticació de ciberatacs ha causat que entitats de recerca i companyies privades cerquin noves formes de detecció d'aquests ciberatacs. Un dels camps de recerca actuals consisteix en l'aplicació de tècniques d'intel·ligència artificial i machine learning per a la detecció d'aquests atacs.

1.1 Objectius del treball

L'objectiu principal d'aquest treball final de Màster consistia en estudiar la implementació d'un arbre de decisió (decision-tree) a partir d'un conjunt de dades (dataset) relacionades amb la seguretat informàtica i obtenir un model per a la posterior identificació i predicció d'atacs similars.

La idea consisteix en entrenar un algorisme per tal que pugui reconèixer en base a patrons una bateria preestablerta d'atacs coneguts. La posterior identificació, predicció i mitigació de nous atacs queda fora de l'objectiu d'aquest treball.

Enumeració d'objectius específics:

- L'aprenentatge d'algorismes relacionats amb la intel·ligència artificial i machine learning i la posterior implementació d'un algorisme de machine learning sobre un dataset existent. La creació i obtenció d'un nou dataset no serà objectiu d'aquest treball, així que se n'emprarà un de disponible.
- Estudiar com un algorisme de machine learning pot ser emprat per a la detecció (i posterior prevenció) d'atacs informàtics i seguretat informàtica en general.

- Estudiar com avaluar un model predictiu elaborat amb les tècniques anteriors.

Llistat de tasques a realitzar:

- Realitzar un breu estudi sobre diferents algorismes de machine learning.
- Escollir una base de dades (dataset) amb dades relacionades amb la seguretat informàtica.
- Escollir un dels algorismes de machine learning per a aplicar-lo sobre les dades del punt anterior.
- Obtenir un model predictiu emprant l'algorisme escollit sobre el dataset escollit.
- Estudiar com analitzar el grau d'efectivitat d'aquest model.
- Avaluar el model obtingut. Comparar el grau d'efectivitat del model obtingut amb algun altre model obtingut sobre les mateixes dades (si és que aquest està disponible).

1.2 Planificació temporal

La planificació temporal programada en la PAC 1 fou la següent:

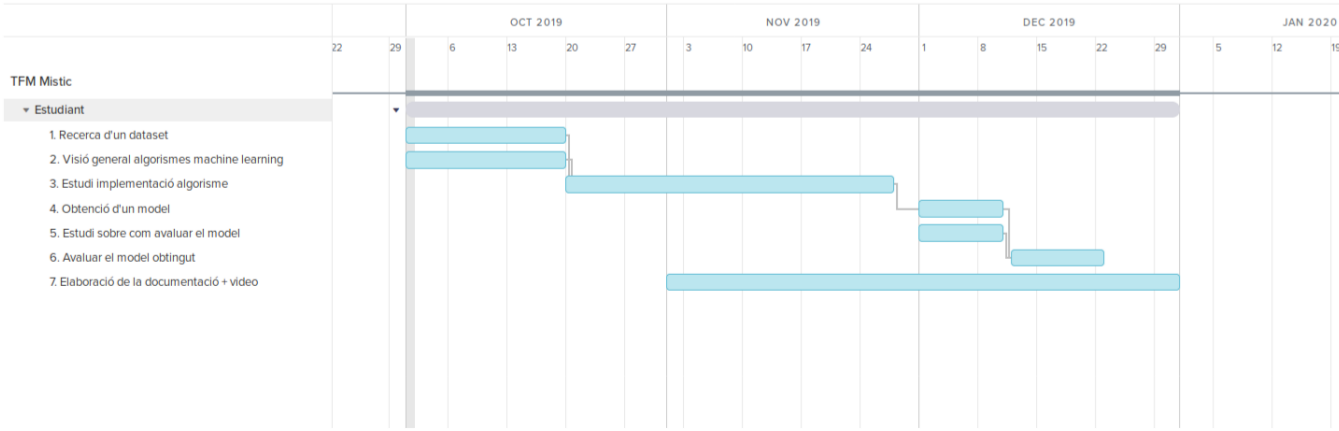
1. Cerca d'un dataset amb dades relacionades amb la seguretat informàtica. Escollir-ne un en base a interessos de l'estudiant, que sigui modern i actual, que disposi d'informació suficient, etc... Dates: 1.10 al 19.10.
2. Visió general dels algorismes relacionats machine learning. Iniciarem la tasca centrant-nos amb els decision-trees. Tasca que podem realitzar en paral·lel amb la primera. Dates: 1.10.19 al 19.10.19.
3. Estudi sobre com implementar l'algorisme estudiat en la tasca 2 sobre el dataset de la tasca 1. Dates: 20.10.19 al 27.11.19.
4. Obtenció d'un model. Dates: 1.12.19 al 10.12.19.
5. Estudiar com avaluar el model obtingut. Dates: 1.12.19 al 10.12.19.

6. Avaluar el model obtingut amb la tasca anterior. Dates: 12.12.19 al 22.12.19.

7. Elaboració de la documentació, que podrem anar escrivint en paral·lel. Dates: 1.11.19 al 31.12.19.

Adjuntem a continuació un diagrama de Gantt amb la proposta temporal:

Figure 1: Diagrama amb la planificació del projecte (PAC1)



1.3 Revisió de les taques programades

En la primera revisió de les tasques programades es va escriure que: "Penso que serà possible disposar d'un model abans del previst i potser repensar les tasques 5 i 6 agregant assumptes com:

- Recerca de literatura científica (o d'altres projectes) on es faci servir aquest mateix dataset.

- Veure si es pot comparar el model obtingut amb d'altres projectes de machine learning sobre el mateix dataset.

- Seria adequat posar-se en contacte de nou amb la institució del dataset i informar sobre el treball. Estudiar possibles sinergies o col·laboracions. Per la resta, es manté la planificació tal com està."

Respecte als dos primers punts, a causa del fet de tractar-se d'un dataset força recent, no vàrem trobar cap referència en la literatura científica on aparegui el dataset. Es va cercar, en concret, a ISI Web of Science[1] i a Scopus[2], sempre partint de la biblioteca digital de la UOC. En cap dels casos, va aparèixer cap resultat.

Figure 2: Recerca del dataset a "ISI Web of Science"

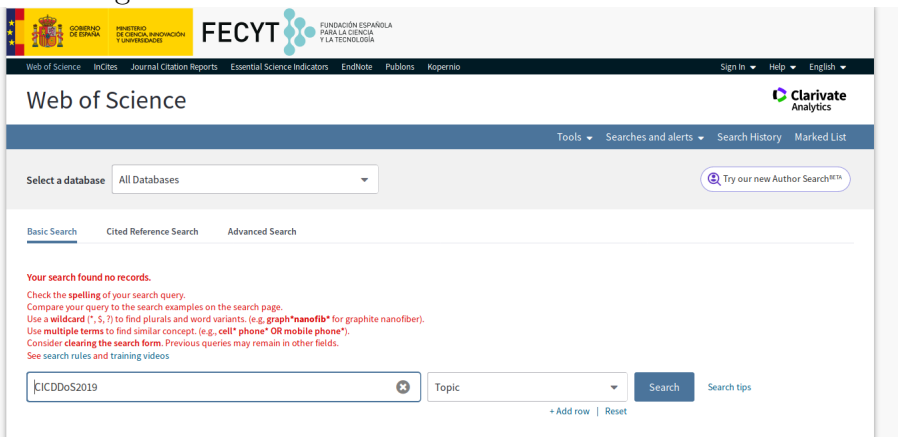
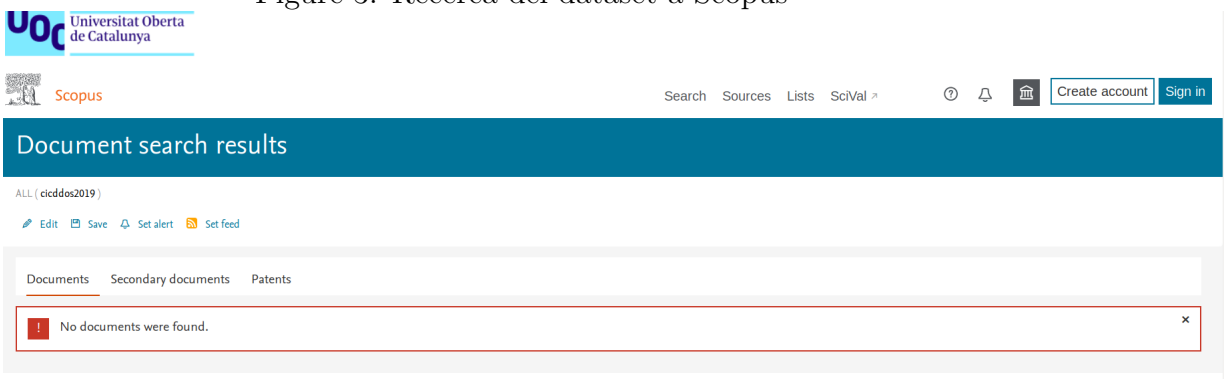


Figure 3: Recerca del dataset a Scopus



Respecte a posar-se en contacte amb la institució del dataset, sí que es va realitzar. En PhD Arash Habibi Lashkari[3] ens va indicar en quin article

[1] http://biblioteca.uoc.edu/ca/recursos/recurs/isi-web-science
[2] https://www.scopus.com/home.uri
[3] https://www.linkedin.com/in/arash-habibi-lashkari-8917461b/

podríem trobar més informació sobre el dataset generat i aquest ha estat un dels articles de referència de partida d'aquest TFM[1].

2 Datasets de ciberseguretat

Per tal d'elaborar aquest treball, calia cercar un dataset amb informació relacionada amb la seguretat informàtica. Existeixen datasets amb informació diversa. Per exemple, detecció d'atacs de DDoS, detecció de botnets, atacs sobre aplicacions web, atacs sobre HTTP, seguretat en el núvol, detecció d'intrusions a la xarxa (NIDS[4]), detecció d'exploits, worms, shellcodes, entre d'altres. Inicialment es varen avaluar els datasets següents:

- Datasets que apareixen en l'article [2] "A Review on Cyber Security Datasets for Machine Learning Algorithms".
- Datasets ubicats en el compte de github següent:

  `https://github.com/shramos/Awesome-Cybersecurity-Datasets`.

  Aquest lloc diu es defineix ell mateix com a "Awesome-Cybersecurity-Datasets. A curated list of amazingly awesome Cybersecurity datasets."

A continuació es descriuen millor alguns d'aquests datasets:

AIDA "DDoS Attack 2007" dataset[5] conté una hora de traces de tràfic anonimitzat d'un atac de DDoS del 4 d'agost del 2007. Aquest tipus d'atac de servei va consistir en blocar l'accés al servidor a través de la sobrecàrrega dels recursos computacionals del servidor i també en consumir tota la seva amplada de banda.

El dataset CTU-13[6] conté tràfic de botnets capturats per la CTU University[7] (República Txeca), l'any 2011[3]. L'objectiu d'aquest dataset fou realitzar una captura llarga de tràfic de botnet conjuntament amb tràfic normal i tràfic en segon pla (background traffic). Aquest dataset conté 13 captures

[4] Acrònim de Network Intrusion Detection Systems

[5] Veure: `https://www.caida.org/data/passive/ddos-20070804_dataset.xml`

[6] `https://mcfp.weebly.com/the-ctu-13-dataset-a-labeled-dataset-with-botnet-normal-and-backgrou`
`html`

[7] `https://www.cvut.cz/en`

(anonenades scenarios) amb diferents tipus de botnets. En cada scenario hi ha l'execució d'un malware específic, que utilitza diferents protocols i realitza diferents accions. Cada escenari es captura en un fitxer PCAP[8] que conté tots els paquets amb els diferents tipus de tràfic.

Finalment però, a través de noves cerques per la xarxa, vàrem trobar un dataset titulat CICDDoS2019[9], generat per la Universitat de New Brunswick[10]. Aquest dataset, també generat en un entorn acadèmic, és molt més actual que els anteriors (generat aquest mateix any 2019) així que es considera disposaria d'informació actualitzada. Aquest dataset, accessible públicament, conté un nou dataset amb atacs distribuïts de denegació de serveis (DDoS). S'escull com a objecte d'anàlisis per aquest treball.

2.1 Descripció de CICDDoS2019

El dataset CICDDoS2019 conté un llistat d'atacs de DDoS actuals, compilats a partir de tràfic real a través de fitxers PCAP. Aquest mateix dataset inclou també els resultats de l'anàlisi del tràfic de xarxa emprant el programari CICFlowMeter-V3[11], que etiqueta el tràfic en funció de l'hora, les adreces ip d'origen i destí, els ports d'origen i destí, així com els protocols i l'atac en qüestió.

Els atacs distribuïts de denegació de servei (DDoS) són una amenaça a la seguretat de la xarxa que tracten d'exhaurir la xarxes objectiu amb tràfic maliciós. Un detector en temps real amb poca demanda de computació seria un producte de molt interès. D'altra banda, l'avaluació de nous algorismes de detecció radica fortament sobre l'existència de datasets ben dissenyats.

2.2 Taxonomia dels atacs

En aquest apartat es resumeix breument la taxonomia d'atacs de DDoS que es proposen en l'article "Developing Realistic Distributed Denial of Service

[8] https://www.tcpdump.org/pcap.html

[9] `https://www.unb.ca/cic/datasets/ddos-2019.html`

[10] `https://www.unb.ca/`

[11] Veure: `https://github.com/ahlashkari/CICFlowMeter`. Extret literalment del lloc web "ICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyzer"

(DDoS) Attack Dataset and Taxonomy". Malgrat es comenta que existeixen diversos estudis relatius a proposar taxonomies per a atacs distribuïts de Denegació de Servei (DDoS), es tracta d'un camp que genera canvis ràpids i proposa una taxonomia actualitzada. L'article proposa dos tipus d'atacs: uns basats en la reflexió i d'altres en l'explotació.

Atacs Distribuïts de Denegació de Servei (DDoS) basats en la reflexió: es tracta d'aquells atacs on la identitat de l'atacant es manté oculta pel fet d'utilitzar components legítims basats en terceres parts. Els paquets s'envien cap a servidors reflectors per part dels atacants on l'adreça ip d'origen és la ip de la víctima i que té com a objectiu saturar la víctima amb paquets de resposta. Aquests atacs es realitzen a través de protocols amb capes transport tant UDP com TCP o amb una combinació dels dos. En aquesta categoria s'estableixen els atacs basats en TCP com MSSSQL, SSDP i també basats en UDP com els atacs CharGen, NTP i TFTP. Els que poden emprar els dos protocols són atacs basats en DNS, LDAP, NETBIOS i SNMP.

Atacs Distribuïts de Denegació de Servei (DDoS) basats en l'explotació[12] Es tracta d'aquells atacs en la qual l'atacant intenta explotar el servei remot directament. Aquests atacs es poden dur a terme a través de protocols de capa de transport tipus TCP, UDP o els dos. Basats en TCP, per exemple, hi ha l'atac d'inundació per SYN i, basats en UDP, els d'inundació via UDP i UDP-Lab. En l'atac d'inundació de paquets UDP, l'atacant envia al servidor remot un alt nombre de paquets UDP. Aquests paquets es trameten a ports aleatoris de la màquina destinatària a alta velocitat. Com a resultat, l'amplada de banda disponible de la xarxa es veu exhausta, el sistema cau i el rendiment es degrada. D'altra banda, en els atacs de Syn Flood, l'objectiu consisteix en consumir de nou els resursos de la víctima explotant el TCP-three-way handshake. Aquest atac s'inicia trametent paquets de tipus SYN cap a la màquina objectiu fins que la màquina no respon.

En aquest dataset hi ha diferents atacs moderns distribuïts de denegació de servei com són: PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, SYN, NTP, DNS, and SNMP.

[12]Respecte a aquesta classificació, hi ha certa confusió. En el lloc web del DDoS Evaluation Dataset CICDDoS2019 es descriuen els dos atacs exactament igual. L'autor d'aquest TFM considera que es tracta d'un error així que es reescriu aquí aquesta taxonomia amb criteris propis.

El període de captura d'aquest dataset va consistir en dues sessions: un "training day", que fou el dia 12 de gener de 2019, començant a les 10:30 i fins les 17:15 i, un "testing day", que fou el dia 11 de març i on els atacs van començar a les 09:40 fins les 17:35. Es van executar 12 atacs en el training day (en concret, NTP, DNS, LDAP, MSSQL, NetBIOS, SNMP, SSDP, UDP, UDP-Lab, WebDDoS, Syn i TFTP) i 7 atacs en el training day (PortScan, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag i SYN).

2.3 CICFlowMeter i característiques del tràfic

CICFlowMeter[13] és un programari que genera anàlisis del tràfic de xarxa i que mostra 84 característiques d'aquest tràfic. El programari llegeix fitxers de tipus PCAP i genera informes amb l'extracció d'aquestes característiques. També genera fitxers en format CSV (que seran els que s'empren en aquest TFM). Es tracta d'una aplicació de programari lliure escrita en llenguatge de programació Java i disponible a través de GitHub[14].

CICFlowMeter genera flow bidireccionals (Biflow), on el primer paquet determina la direcció d'anada (des de l'adreça origen a la de destí) i també la direcció de tornada (backward) (de la destinació a l'origen) i, a partir d'aquí, 84 característiques estadístiques més com la durada, el nombre de paquets, el nombre de bytes, la longitud dels paquets, etc.. que es calculen de manera separada entre la direcció d'anada i la de tornada. En el fitxer CSV resultant hi ha, entre d'altres, les sis columnes següents: un FlowID (que actua com a identificador del flow), una adreça ip d'origen (source ip), una adreça ip de destí (destination ip), un port d'origen (SourcePort), un port de destí (DestinationPort) i el protocol emprat. Cal recordar que els flows en TCP finalitzen normalment amb un paquet FIN mentre que els flows en UDP finalitzen via un timeout.

Els camps que conté aquest dataset són els següents:

```
Flow ID: identificador amb el flux de la connexió.
Source IP: adreça ip origen del flux.
```

[13] Veure: `http://www.netflowmeter.ca`

[14] Codi font de l'aplicació disponible en aquest compte de GitHub: `https://github.com/ahlashkari/CICFlowMeter`

Source Port: port del protocol de transmissió emprat en el flux.
Destination IP: adreça ip de destí del flux.
Destination Port: port del protocol de transmissió emprat en el flux.
Protocol : protocol utilitzat
Timestamp : segell del temps.
Flow Duration: Duració del flux en microsegons
Total Fwd Packets: Nombre total de paquets en la direcció forward.
Total Backward Packets : Nombre total de paquets en la direcció backward.
Total Length of Fwd Packets: Mida total dels paquets en la direcció forward.
Total Length of Bwd Packets : Mida total dels paquets en la direcció backward.
Fwd Packet Length Max : Mida màxima del paquet en la direcció forward.
Fwd Packet Length Min : Mida mínima del paquet en la direcció forward.
Fwd Packet Length Mean : Mitjana de la mida dels paquets en la direcció forward.
Fwd Packet Length Std: Desviació estàndard de la mida dels paquets en forward.
Bwd Packet Length Max : Mida màxima dels paquets en direcció forward.
Bwd Packet Length Min: Mida mínima del paquet en la direcció de backward.
Bwd Packet Length Mean : Mida mitjana del paquet en la direcció de backward.
Bwd Packet Length Std : Desviació estàndard del paquet en backward.
Flow Bytes/s : Nombre de paquets del flux per segon.
Flow Packets/s : Nombre de bytes del paquet per segon.
Flow IAT Mean : Temps de mitjana entre dos paquets enviats en el flux.
Flow IAT Std : Desviació estàndard de temps entre dos paquets tramesos
en el mateix flux.
Flow IAT Max : Temps màxim entre dos paquets tramesos en el mateix flux.
Flow IAT Min : Temps mínim entre dos paquets tramesos en el mateix flux.
Fwd IAT Total : Temps total entre dos paquets tramesos en forward.
Fwd IAT Mean : Temps mitjà entre dos paquets tramesos en forward.
Fwd IAT Std : Desviació estàndard de temps entre dos paquets tramesos
 en el mateix flux.
Fwd IAT Max : Temps màxim entre dos paquets tramesos en forward.
Fwd IAT Min : Temps mínim entre dos paquets tramesos en forward.
Bwd IAT Total : Temps total entre dos paquets tramesos en backward.
Bwd IAT Mean : Temps mitja entre dos paquets tramesos en backward.
Bwd IAT Std : Desviació estàndard del temps entre dos paquets
 tramesos en backward.
Bwd IAT Max : Temps màxim entre dos paquets tramesos en backward.
Bwd IAT Min : Temps mínim entre dos paquets tramesos en backward.
Fwd PSH Flags : Vegades que el flag PSH fou establert en els paquets

tramesos en forward (0 en el cas de UDP).
Bwd PSH Flags: Vegades que el flag PSH fou establert en els paquets tramesos en backward (0 en UDP).
Fwd URG Flags: Vegades que el flag URG fou establert en els paquets tramesos en forward (0 en UDP).
Bwd URG Flags: Vegades que el flag PSH fou establert en els paquets tramesos en backward (0 en UDP).
Fwd Header Length : Nombre total de bytes usats en les capçaleres en la direcció de forward.
Bwd Header Length : Nombre total de bytes usats en les capçaleres en la direcció de backward.
Fwd Packets/s : Nombre de paquets en la direcció de forward per segon.
Bwd Packets/s : Nombre de paquets en la direcció de backward per segon.
Min Packet Length : Longitud mínima d'un paquet.
Max Packet Length : Longitud màxima d'un paquet.
Packet Length Mean : Mitjana de la longitud d'un paquet.
Packet Length Std : Desviació estàndard de la longitud d'un paquet.
Packet Length Variance : Variança de la longitud d'un paquet.
FIN Flag Count : Nombre de paquets amb un FIN.
SYN Flag Count : Nombre de paquets amb un SYN.
RST Flag Count : Nombre de paquets amb un RST.
PSH Flag Count : Nombre de paquets amb un PSH.
ACK Flag Count : Nombre de paquets amb un ACK.
URG Flag Count : Nombre de paquets amb un URG.
CWE Flag Count : Nombre de paquets amb un CWE.
ECE Flag Count : Nombre de paquets amb un ECE.
Down/Up Ratio : Ràtio de descàrrega i de pujada.
Average Packet Size : Mida mitjana dels paquets.
Avg Fwd Segment Size : Mida mitjana observada en la direcció de forward.
Avg Bwd Segment Size : Mida mitjana observada en la direcció de Backward.
Fwd Header Length.1 : Longitud de la capçalera en el paquet de forward.
Fwd Avg Bytes/Bulk : Nombre mitjà de bytes de bulk rate en la direcció de forward.
Fwd Avg Packets/Bulk : Nombre mitjà de paquets de bulk rate en forward.
Fwd Avg Bulk Rate : Nombre mitjà de bulk rate en forward.
Bwd Avg Bytes/Bulk : Nombre mitjà en bytes de bulk rate en forward.
Bwd Avg Packets/Bulk : Nombre mitjà de bulk rate en la direcció de backward.
Bwd Avg Bulk Rate : Nombre mitjà de bulk rate en la direcció de backward.
Subflow Fwd Packets : Nombre mitjà de paquets en un subflow en forward.

Subflow Fwd Bytes : Nombre mitjà de bytes en un subflow en forward.
Subflow Bwd Packets : Nombre mitjà de paquets en un subflow en backward.
Subflow Bwd Bytes : Nombre mitjà de paquets en un subflow en backward.
Init_Win_bytes_forward : Nombre total de bytes enviats en la
finestra inicial en forward.
Init_Win_bytes_backward : El nombre total de bytes enviats en la
finestra inicial en backward.
act_data_pkt_fwd : Recompte de paquets amb almenys 1 byte de
payload de dades TCP en forward.
min_seg_size_forward : Mida mínima de segment observada en forward.
Active Mean : Temps mitjà del flux sent actiu abans de convertir-se en inactiu.
Active Std : Desviació estàndard del temps del flux sent actiu
abans de convertir-se en inactiu.
Active Max : Màxim temps del flux sent actiu
abans de convertir-se en inactiu.
Active Min : Mínim temps del flux sent actiu
abans de convertir-se en inactiu.
Idle Mean : Mitjana de temps d'un flux estant
com a inactiu i convertint-se en actiu.
Idle Std : Desviació estàndard de temps d'un flux estant
com a inactiu i convertint-se en actiu.
Idle Max : Temps màxim d'un flux estant com a inactiu i
convertint-se en actiu.
Idle Min : Temps mínim d'un flux estant com a inactiu i
convertint-se en actiu.
Label: etiqueta que classifica l'atac.

Diversos fitxers CSV amb columnes etiquetades amb les característiques esmentades aquí actuaran com a dataset per a l'entrenament dels arbres de decisió. Més endavant s'explicaran més detalls.

3 Introducció a machine learning

Andrew Ng[15], professor adjunt de Computer Science a Stanford University[16], definia machine learning com "the science of getting computers to act without being explicitly programmed". Un dels objectius del machine learning consisteix en la construcció de models tals que, a partir de dades d'entrada i l'ús d'anàlisis estadístics puguin efectuar prediccions posteriors.

Els algorismes de machine learning sovint es classifiquen entre els supervisats, els nosupervisats i l'aprenentatge de reforç (Reinforcement Learning).

Respecte els algorismes supervisats es poden agrupar entre els de regressió i classificació. Alguns dels algorismes més utilitzats són els següents: la regressió lineal, regressió logística, support vector machines (SVMs), Naive Bayes, K-nearest neighbors vote (KNN), K-Means, entre d'altres. També els arbres de decisió (decision-trees), que fou l'algorisme planificat a l'origen d'aquest treball i, per extensió a aquests, els random-forests.

3.1 Introducció als arbres de decisió

Entre els diversos tipus d'aprenentatge automàtic distingibles, aquest treball es basa en els que utilitzen un aprenentatge inductiu: a partir de mostres discretes reals N

$$e_1, ..., e_N$$

es realitza una recerca per a obtenir un patró general.

Cada mostra es defineix a partir d'un conjunt d'atributs i una etiqueta amb la qual és possible classificar la mostra.

La tasca de l'aprenentatge inductiu consisteix en induir de les dades discretes un mecanisme que permeti inferir les classificacions de cada un dels exemples a partir de les seves propietats. En el cas que s'aconseguís generar un model adequat, es podria emprar aquest mateix mecanisme per tal de deduir posteriorment la classificació de nous exemples a través de l'observació de les seves propietats.

[15] `https://www.linkedin.com/in/andrewyng/`
[16] `https://www.stanford.edu/`

Cal tenir en compte que els resultats que s'obtenen depenen molt de la qualitat dels exemples introduïts (que n'hi hagi una quantitat suficient, que representin fidelment el comportament de la població no observada, que no mostrin contradiccions internes, etc..). Entre tots els possibles mecanismes per tal d'obtenir aquestes prediccions, una de les tècniques destacades són els **arbres de decisió**, que proporcionen un conjunt de regles que s'apliquen sobre els exemples nous per tal de decidir quina és la classificació més adequada en funció dels seus atributs.

Un arbre de decisió està format per un conjunt de nodes de decisió (interiors) i de nodes de resposta (fulles):

- Un **node decisió** està associat a un dels atributs i té dues o més rames que surten d'ell, cadascuna d'elles representant els possibles valors que pot prendre l'atribut associat. Un node decisió actua com una pregunta que se li fa a l'exemple analitzat i, en funció de la resposta, agafarà un flux en una de les rames sortints. En el cas d'aquest treball, es tractarà de nodes de decisió binaris, amb solament dues rames.

- Un **node de resposta**, en canvi, està associat directament a la classificació que es desitja proporcionar i retorna la decisió de l'arbre respecte l'exemple d'entrada.

No sempre serà possible aconseguir un arbre de decisió capaç de preveure els exemples amb una fiabilitat del 100% però, com millor sigui la bateria d'exemples que es disposi (per exemple, sense contradiccions en les classificacions) , millor es comportarà l'arbre construït.
Alguns dels avantatges dels arbres de decisió són els següents:[17]

- Són simples de comprendre i interpretar. A més, es poden visualitzar fàcilment si són petits.

- Requereixen de poca preparació de les dades. D'altres tècniques sovint requereixen la normalització sobre les dades (no cal, per exemple, *feature scaling*[18], no s'han de suprimir variables amb valors en blanc, entre d'altres tasques).

[17]Extret d'aquí: `https://scikit-learn.org/stable/modules/tree.html`
[18]Veure: `https://en.wikipedia.org/wiki/Feature_scaling`

- Són capaços de gestionar dades tant numèriques com categòriques. D'altres tècniques són útils solament quan analitzen dades amb un sol tipus de variable. En aquest treball però, s'ha optat per no emprar dades categòriques.
- És possible de validar el model emprant testos estadístics.

També tenen alguns desavantatges:

- Els arbres generats poder ser tant complexos que poden no generalitzar les dades correctament (**overfitting**). Mètodes per a estalviar-se aquest problema poden ser: requerir en els nodes de resposta un nombre mínim d'exemples o establir el nombre màxim de profunditat de l'abre.
- Els arbres de decisió poder ser inestables en el sentit que una petita variació en les dades pugui resultar en un arbre completament diferent del generat.
- Alguns conceptes són difícils d'aprendre i és que els abres no els expressen directament (XOR, paritat o multiplexació).
- En el cas que hi hagi classes dominants, és possible que els arbres generats estiguin esbiaixats. És recomanable per tant balancejar els testos d'entrada abans de realizar el procés d'entrenament dels arbres.

El criteri que s'empra per mesurar la qualitat de cada partició recau en el concepte d'**information gain** que acostuma a emprar un dels dos criteris següents:

- Criteri basat en l'**entropia** i en el guany d'informació.
- Criteri de **Gini**, basat en la impuresa del node.

El guany d'informació s'utilitza per a determinar quina característica o atribut ofereix el màxim d'informació respecte una classe. Recordem la fórmula per al càlcul de l'entropia en un conjunt de dades:

$$E(S) = \sum_{i=1}^{c}(-1) * p_i * log_2(p_i)$$

On 'p' denota la probabilitat que una mostra formi part d'una categoria concreta i E(S) denota l'entropia del conjunt de dades.

En sistemes computacionals el càlcul del logaritme es relativament costós, així que s'acostuma a emprar l'**índex de Gini**.

L'índex de Gini, o la impuresa de Gini, mesura el grau o la probabilitat d'una variable particular de ser classificada incorrectament quan s'escull de manera aleatòria. En el cas que tots els elements pertanyin a una mateixa classe, es pot dir que aquesta és pura.

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

on p_i és la probabilitat d'un objecte de ser classificat segons una classe particular. Quan es construeix un arbre de decisió, es prefereix utilitzar la característica amb l'índex de Gini més baix per al node arrel.

3.2 Complexitat computacional

El cost per a construir un decision tree balancejat binari basat en CART (Classification and Regression Tree) és de [4] :

$$O(n_{samples} * n_{features} * log_2(n_{samples})$$

En el cas que ens ocupa, si es desitgés construir un arbre emprant tots els registres disponibles en el training day i emprant totes les característiques (més endavant es donen més detalls sobre els fitxers en concret):

```
$ ls
DrDoS_DNS.csv  DrDoS_LDAP.csv  DrDoS_MSSQL.csv  DrDoS_NetBIOS.csv  DrDoS_NTP.csv
DrDoS_SNMP.csv  DrDoS_SSDP.csv  DrDoS_UDP.csv  Syn.csv  TFTP.csv  UDPLag.csv
$ cat *.csv | wc -l
50063123
```

que fa referència al nombre de nSamples. Pel que fa al nombre de característiques:

```
$ head -n1 TFTP.csv  | grep -o "," | wc -l
87
```

Per tant, solament en el training day:

$$O(50063123 * 87 * log_2(50063123)$$

3.3 Tecnologies per la implementació d'arbres de decisió

Per a la creació dels arbres de decisió vàrem avaluar diverses tecnologies. Concretament, les següents: Python, R i una plataforma d'intel·ligència artificial i machine learning d'Amazon.

Amazon[19] fou descartat ràpidament i és que va semblar que no hi hauria un control prou exhaustiu de tot el procés i que caldria pagar. Les alternatives foren R i Python.

Python és un llenguatge de programació d'alt nivell i de propòsit general àmpliament utilitzat. Ideal per a principiants. Creat l'any 1991 permet un codi llegible i flexible on, amb poques línies de codi, es poden realitzar moltes tasques.

Python disposa d'una llibreria, anomenada Scikit-learn[20], que està disponible amb llicència de programari lliure i que implementa diversos algorimes de machine learning. Conté algorimes per a la classificació, regressió, clustering amb suport per a vector machines, random forests, gradient boosting, k-means, entre d'altres. Empra les llibreries de caire científic NumPy[21] i SciPy[22].

R[23] és un llenguatge de programació àmpliament emprat per al desenvolupament de programari estadístic i anàlisi de dades i s'està plasmant com un estàndard per al desenvolupament de nou programari estadístic. Creat

[19] "Amazon SageMaker is a fully-managed platform that enables developers and data scientists to quickly and easily build, train, and deploy machine learning models at any scale". Extret de manera literal del seu lloc web `https://aws.amazon.com/sagemaker/`

[20] "Scikit-learn is a free software learning library for the Python programming language". Extret de `https://scikit-learn.org/stable/`

[21] `https://numpy.org/`

[22] `https://www.scipy.org/`

[23] `https://www.r-project.org/`

originalment des de la Universitat d'Auckland a Nova Zelanda, està desenvolupat actualment per un Equip de Desenvolupament de R. Tot el codi de R està disponible de manera oberta i gratuïtament sota llicència GNU.

Després de realitzar algunes proves es va decidir emprar Python i és que permet la creació de decision-trees amb molta facilitat.

D'altra banda, la llibreria scikit-learn incorpora diferents algorismes per a crear decision-trees[5] ID3 (Iterative Dichotomiser 3), C4.5 (el successor de ID3), C5.0 i CART (Classification and Regression Trees), molt similar al C4.5. Scikit-learn utilitza una versió optimitzada de l'algorisme de CART encara que, actualment, no suporta variables categòriques.

4 Procés d'entrenament

Es descriu en aquest apartat tot el detall tècnic de la implementació dels algorismes. Per a la implementació s'ha emprat una màquina amb les característiques següents:

```
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 18.04.2 LTS
Release: 18.04
Codename: bionic

$  uname -a
Linux Zeus 4.15.0-54-generic #58-Ubuntu SMP Mon Jun 24 10:55:24 UTC 2019 x86_64
x86_64 x86_64 GNU/Linux

$ free -h
              total        used        free      shared  buff/cache   available
Mem:            15G        807M         12G        8,5M        2,1G         14G
Swap:           15G        1,1G         14G

$ grep -c processor /proc/cpuinfo
12
```

```
$ grep  "model name" /proc/cpuinfo | head -1
model name : Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz
```

Una màquina amb 12 CPU's, 16G de memòria RAM amb Ubuntu 18.04.

La versió de Python emprada és la següent:

```
$ python3 --version
Python 3.6.8
```

4.1 Descàrrega dels fitxers del dataset

Els fitxers del dataset es poden descarregar de manera gratuïta previ un registre a través del lloc web següent:

```
http://205.174.165.80/CICDataset/CICDDoS2019/
```

El dataset consisteix en dos fitxers .zip (un fa referència al tràfic del training day i un altre al del testing day). És adequat la verificació del MD5 en cada cas. Segons les adreces:

```
http://205.174.165.80/CICDataset/CICDDoS2019/Dataset/CSVs/CSV-01-12.
md5
http://205.174.165.80/CICDataset/CICDDoS2019/Dataset/CSVs/CSV-03-11.
md5
```

S'observa que el hash md5 haura de ser:

```
b86b3553b1c5086222b27ea17a27e07a  CSV-01-12.zip
d4cfa92e3eb4fc30b9100f5d09b66efe  CSV-03-11.zip
```

que es corresponen certament amb els següents:

```
$ md5sum *.zip
b86b3553b1c5086222b27ea17a27e07a  CSV-01-12.zip
d4cfa92e3eb4fc30b9100f5d09b66efe  CSV-03-11.zip
```

4.2 Descripció dels fitxers del dataset

El dataset escollit conté dos fitxers .zip que disposen de diversos fitxers en format CSV (Comma Separated Values). En concret:

```
$ ls -lh *.zip
-rw-rw-r-- 1 gerard gerard 2,2G de no 10 15:19 CSV-01-12.zip
-rw-rw-r-- 1 gerard gerard 1,2G de no 10 15:16 CSV-03-11.zip
```

El fitxer del 12 de gener (CSV-01-12.zip) conté les dades del training day i, el del 11 de març (CSV-03-11.zip) el del testing day. Cal descomprimir ambdós fitxers. En la carpeta 01-12/:

```
$ ls -lh *.csv
-rwxrwxrwx 1 gerard gerard  2,0G de ma  7  2019 DrDoS_DNS.csv
-rwxrwxrwx 1 gerard gerard  875M de ma  1  2019 DrDoS_LDAP.csv
-rwxrwxrwx 1 gerard gerard  1,8G de ma  1  2019 DrDoS_MSSQL.csv
-rwxrwxrwx 1 gerard gerard  1,6G de ma  1  2019 DrDoS_NetBIOS.csv
-rwxrwxrwx 1 gerard gerard  616M de ma  7  2019 DrDoS_NTP.csv
-rwxrwxrwx 1 gerard gerard  2,1G de ma  1  2019 DrDoS_SNMP.csv
-rwxrwxrwx 1 gerard gerard  1,2G de ma  1  2019 DrDoS_SSDP.csv
-rwxrwxrwx 1 gerard gerard  1,5G de ma  1  2019 DrDoS_UDP.csv
-rwxrwxrwx 1 gerard gerard  608M de ma  1  2019 Syn.csv
-rwxrwxrwx 1 gerard gerard  8,7G de ma  1  2019 TFTP.csv
-rwxrwxrwx 1 gerard gerard  151M de ma  1  2019 UDPLag.csv
```

I, en la carpeta 03-11/:

```
$ls -lh
total 6,2G
-rwxrwxrwx 1 gerard gerard 832M de ma  6  2019 LDAP.csv
-rwxrwxrwx 1 gerard gerard 2,3G de ma  6  2019 MSSQL.csv
-rwxrwxrwx 1 gerard gerard 1,4G de ma  6  2019 NetBIOS.csv
-rwxrwxrwx 1 gerard gerard  75M de ma  6  2019 Portmap.csv
-rwxrwxrwx 1 gerard gerard 1,8G de ma  6  2019 Syn.csv
-rwxrwxrwx 1 gerard gerard 1,7G de ma  6  2019 UDP.csv
-rwxrwxrwx 1 gerard gerard 305M de ma  6  2019 UDPLag.csv
```

En cadascun d'aquests fitxers hi ha la mateixa capçalera, que indica l'atribut de cada columna. Per exemple:

```
$ head -1 LDAP.csv
Unnamed: 0,Flow ID, Source IP, Source Port, Destination IP, Destination Port,
Protocol, Timestamp, 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, Bwd Packet Length Min, Bwd Packet Length Mean, Bwd
Packet Length Std,Flow Bytes/s, Flow Packets/s, Flow IAT Mean, Flow IAT Std,
Flow IAT Max, Flow IAT Min,Fwd IAT Total, Fwd IAT Mean, Fwd IAT Std, Fwd IAT Max,
Fwd IAT Min,Bwd IAT Total, Bwd IAT Mean, Bwd IAT Std, Bwd IAT Max, Bwd IAT Min,
Fwd PSH Flags, Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, Fwd Header Length,
Bwd Header Length,Fwd Packets/s, Bwd Packets/s, Min Packet Length,
Max Packet Length,
Packet Length Mean, Packet Length Std, Packet Length Variance,FIN Flag Count,
SYN Flag Count, RST Flag Count, PSH Flag Count, ACK Flag Count, URG Flag Count,
CWE Flag Count, ECE Flag Count, Down/Up Ratio, Average Packet Size,
Avg Fwd Segment Size,
Avg Bwd Segment Size, Fwd Header Length.1,Fwd Avg Bytes/Bulk,
Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk,
Bwd Avg Packets/Bulk,Bwd Avg Bulk Rate,Subflow Fwd Packets,
Subflow Fwd Bytes, Subflow Bwd Packets, Subflow Bwd Bytes,
Init_Win_bytes_forward, Init_Win_bytes_backward, act_data_pkt_fwd,
min_seg_size_forward,Active Mean, Active Std, Active Max,
Active Min,Idle Mean, Idle Std, Idle Max, Idle Min,SimillarHTTP,
Inbound, Label
```

El significat de cadascun d'aquests camps ha estat escrit anteriorment. És possible recomptar també la quantitat de registres que conté cada fitxer. Primer per al training day i després pels del testing day:

```
$ wc -l *
    5074414 DrDoS_DNS.csv
    2181543 DrDoS_LDAP.csv
    4524499 DrDoS_MSSQL.csv
    4094987 DrDoS_NetBIOS.csv
    1217008 DrDoS_NTP.csv
    5161378 DrDoS_SNMP.csv
    2611375 DrDoS_SSDP.csv
    3136803 DrDoS_UDP.csv
```

```
    1582682 Syn.csv
   20107828 TFTP.csv
     370606 UDPLag.csv
   50063123 total

$ wc -l *.csv
   2113235 LDAP.csv
   5775787 MSSQL.csv
   3455900 NetBIOS.csv
    191695 Portmap.csv
   4320542 Syn.csv
   4643127 UDP.csv
    838562 UDPLag.csv
  21338848 total
```

Es mostra a continuació un parell de línies d'algun d'aquests fitxers, solament a tall d'exemple:

```
$ tail -2 Syn.csv
46401,172.16.0.5-192.168.50.1-43302-40846-6,172.16.0.5,43302,192.168.50.1,
40846,6,2018-12-01 13:34:27.403142,1,2,0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,
2000000.0,1.0,0.0,1.0,1.0,1.0,1.0,0.0,1.0,1.0,0.0,0.0,0.0,0.0,0.0,
0,0,0,0,40,0,2000000.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,0,0,0,1,0,0,0,0.0,
0.0,0.0,0.0,40,0,0,0,0,0,0,2,0,0,0,5840,-1,0,20,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,0.0,0,1,Syn
791758,172.16.0.5-192.168.50.1-43303-51800-6,172.16.0.5,43303,
192.168.50.1,51800,6,2018-12-01 13:34:27.403143,49,2,0,0.0,0.0,0.0,0.0,0.0,
0.0,0.0,
0.0,0.0,0.0,0.0,40816.32653061225,49.0,0.0,49.0,49.0,49.0,
49.0,0.0,49.0,49.0,
0.0,0.0,0.0,0.0,0.0,0,0,0,0,40,0,40816.32653061225,0.0,0.0,0.0,0.0,0.0,0.0,
0.0,0,0,0,0,1,0,0,0,0.0,0.0,0.0,0.0,40,0,0,0,0,0,0,2,0,0,0,5840,-1,
0,20,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0,1,Syn
```

S'observa que, en cada fitxer, hi ha registres amb atacs però també registres amb tràfic benigne:

```
$ cat Syn.csv | grep Syn | wc -l
1582289
$ cat Syn.csv | grep BENIGN | wc -l
392
$ cat Syn.csv | wc -l
1582682
```

1582289 + 392 + 1 registre amb la capçalera = 1582682
S'observen les etiquetes dels diferents atacs en la carpeta del training day:

```
$ cat *.csv | cut -f 88 -d "," | sort | uniq
BENIGN
DrDoS_DNS
DrDoS_LDAP
DrDoS_MSSQL
DrDoS_NetBIOS
DrDoS_NTP
DrDoS_SNMP
DrDoS_SSDP
DrDoS_UDP
Syn
TFTP
UDP-lag
WebDDoS
```

I ara en els fitxers del testing day:

```
$ cat *.csv | cut -f 88 -d "," | sort | uniq
BENIGN
LDAP
MSSQL
NetBIOS
Portmap
Syn
```

Els registres que coincideixen en els dos dies són els següents:

- Tràfic benigne tant en els fitxers del training day com en el del testing day apareix amb l'etiqueta BENIGN.

- Atacs de LDAP que, en un cas, es mostren amb l'etiqueta DrDoS_LDAP i en l'altre solament amb LDAP. Es pressuposa que es tracta del mateix atac.
- Atacs de MSSQL que, en un cas, es mostren amb l'etiqueta DrDoS_MSSQL i en l'altre solament amb etiqueta MSSQL. Es pressuposa de nou que es tracta del mateix atac.
- Atacs de MSSQL que, en un cas, es mostren amb l'etiqueta DrDoS_NetBIOS i en l'altre solament amb l'etiqueta NETBIOS. Es pressuposa de nou que es tracta del mateix atac.
- Atacs de Syn, que en ambdós fitxers es mostren amb l'etiqueta Syn.

Els atacs que estan solament en el training day són els següents: DrDoS_DNS, DrDoS_NTP, DrDoS_SNMP, DrDoS_SSDP, DrDoS_UDP, TFTP, UDP-lag, WebDDoS.

Per tal d'unificar les etiquetes (labels) entre els training i testing days, en un dels dos fitxers vàrem realitzar substitucions massives emprant l'eina sed[24]:

```
$ cat DrDoS_LDAP.csv | sed 's/DrDoS_LDAP/LDAP/g' > DrDoS_LDAP-training.csv
$ wc -l DrDoS_LDAP.csv
2181543 DrDoS_LDAP.csv
$ wc -l DrDoS_LDAP-training.csv
2181543 DrDoS_LDAP-training.csv
```

Les dues darreres línies es realitzen solament per a assegurar-se que la substitució no ha corromput els fitxers.

4.3 Descripció estadística de les característiques

És possible descriure estadísticament les característiques emprant el mateix codi en Python. Concretament, una vegada carregades les dades en un objecte Pandas[25], cal cridar solament la funció `describe`. Per exemple:

[24] Sed (stream editor). A non-interactive command-line text editor. Veure: `https://www.gnu.org/software/sed/`

[25] Veure: `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html`

```
pima = pd.read_csv('../../01-12/DrDoS_DNS.csv', header=None, names=col_names,
dtype={'Flow Bytes/s' : 'object', 'Flow Packets/s' : 'object',
 'SimillarHTTP' : 'object', 'Inbound' : 'object'})

with pd.option_context('display.max_columns', 80):
    print(pima.describe(percentiles=None, include=[np.number]))
```

El resultat és quelcom similar al següent (s'obvien aquí totes les columnes i se'n mostren solament dues a tall d'exemple):

```
       Total Length of Fwd Packets  Total Length of Bwd Packets  \
count                 5.074413e+06                 5.074413e+06
mean                  2.952438e+03                 6.768139e+00
std                   9.122037e+03                 6.837585e+03
min                   0.000000e+00                 0.000000e+00
25%                   2.928000e+03                 0.000000e+00
50%                   2.944000e+03                 0.000000e+00
75%                   2.944000e+03                 0.000000e+00
max                   1.526642e+07                 1.099376e+07
```

Aquesta descripció pot servir també per a veure columnes buides i que es podrien obviar per a la selecció de les característiques. En aquest cas concret, les columnes següents són sempre amb valor zero:

```
Bwd PSH Flags Fwd URG Flags  Bwd URG Flags,  PSH Flag Count
ECE Flag Count Fwd Avg Bytes/Bulk Fwd Avg Packets/Bulk
Fwd Avg Bulk Rate  Bwd Avg Bytes/Bulk   Bwd Avg Packets/Bulk
Bwd Avg Bulk Rate
```

A continuació es realitza una descripció estadística de totes les dades de tots els fitxers del training day. No s'ha pogut fer emprant el mateix mètode anterior i és que el sistema mata el procés de Python quan aquest intenta processar tot el volum de dades (per excés de recursos). Es comentarà aquest assumpte computacional més endavant, en els testos. Solament comentar aquí que la màquina utilitzada no disposa de recursos suficients per a carregar tot el volum de dades. Per a realitzar el mateix càlcul però a través d'una alternativa s'emprarà l'eina Datamash[26]. El codi en Bash que s'ha emprat per a obtenir les dades següents és aquest:

[26]Datamash consisteix en una eina GNU de terminal que permet realitzar tasques numèriques, textuals i operacions estadístiques. Veure: `https://www.gnu.org/software/datamash/`

```
$ more cal.sh
#!/bin/bash
echo " min max mean var"
for i in {9..85}
do
camp=‘cut -d "," -f $i header.txt‘
echo -n "$camp: "
cut -d "," -f $i all0112.csv|sed 's/\./,/g' | datamash min 1 max 1 mean 1 svar 1
done
```

Els resultats mostren, per a cada columna de dades, el valor mínim, el màxim, la mitjana i la desviació estàndard:

```
$ ./cal.sh
min max mean var
 Flow Duration: 0 120000000 1123535,3994078 30327044617363
 Total Fwd Packets: 1 100148 4,9867302895593 59744,11976529
 Total Backward Packets: 0 4602 0,022010557394035 2,2407867380174
Total Length of Fwd Packets: 0 15266416 2394,5418589839 44702193,051568
 Total Length of Bwd Packets: 0 10993758 5,9608306810811 15927156,090198
 Fwd Packet Length Max: 0 32120 679,32755153136 208258,6339225
 Fwd Packet Length Min: 0 2021 673,7759255757 211892,82271589
 Fwd Packet Length Mean: 0 3015,290539206 677,11545442265 209471,06552625
 Fwd Packet Length Std: 0 2221,5560553174 2,4211587671749 132,59793020207
Bwd Packet Length Max: 0 37960 0,44202967246623 1530,6356855447
 Bwd Packet Length Min: 0 1460 0,045031079969619 7,3061199914952
 Bwd Packet Length Mean: 0 5011,0601589103 0,13888658922914 84,868581438742
 Bwd Packet Length Std: 0 7045,0952595505 0,12699243650819 102,67054852335
Flow Bytes/s: datamash: invalid input: field 1 requested, line 2340 has only 0 fie
 Flow Packets/s: 0,030517510604263 inf inf -nan
 Flow IAT Mean: 0 65536145 247865,61048866 385642157565,48
 Flow IAT Std: 0 68082925,445839 401486,38996608 918666434189,44
 Flow IAT Max: 0 119954412 737570,9702919 4339417151971,9
 Flow IAT Min: 0 65536145 36,165331192356 614675565,833
Fwd IAT Total: 0 120000000 1123033,3550065 30319101718867
 Fwd IAT Mean: 0 65536145 250413,89903271 418181557914,48
 Fwd IAT Std: 0 67368953,895632 402751,99148492 945867451862,54
 Fwd IAT Max: 0 119954412 737255,00758786 4335406739482,6
```

```
 Fwd IAT Min: 0 65536145 35,109491075984 618104541,0124
Bwd IAT Total: 0 119999989 27897,272436979 2028790920528,8
 Bwd IAT Mean: 0 58961594 5965,6002252506 102652343846,51
 Bwd IAT Std: 0 83384281,651295 10570,486749887 311737933411,42
 Bwd IAT Max: 0 119744177 21088,157340439 1155015407762,2
 Bwd IAT Min: 0 258 0,027812414058479 1,045164825298
Fwd PSH Flags: 0 1 0,00016247491765993 0,0001624485228059
 Bwd PSH Flags: 0 0 0 0
 Fwd URG Flags: 0 0 0 0
 Bwd URG Flags: 0 0 0 0
 Fwd Header Length: -252927116050 155938 -108978779,43389 2,2632702194255e+18
 Bwd Header Length: -2125437950 147280 -7323,0515725431 15543068450923
Fwd Packets/s: 0 4000000 1065711,891289 838591010217
 Bwd Packets/s: 0 2000000 105,39920779762 11776213,656292
 Min Packet Length: 0 1472 673,76967758217 211890,2383069
 Max Packet Length: 0 37960 679,6044297446 209262,2058688
 Packet Length Mean: 0 4023,9453551913 676,92951478461 209585,0948506
 Packet Length Std: 0 6616,562066007 2,3953826299225 181,52717139152
 Packet Length Variance: 0 43778893,573322 187,2650257092 339117234,36399
FIN Flag Count: 0 0 0 0
 SYN Flag Count: 0 1 1,6619022804655e-05 1,6618746944687e-05
 RST Flag Count: 0 1 0,00016247491765993 0,0001624485228059
 PSH Flag Count: 0 0 0 0
 ACK Flag Count: 0 1 0,041247395887016 0,039546049009475
 URG Flag Count: 0 1 0,00069837847874898 0,00069789076018974
 CWE Flag Count: 0 1 0,00040668666382545 0,00040652127790305
 ECE Flag Count: 0 0 0 0
 Down/Up Ratio: 0 23 0,0052497735258647 0,0065463821160064
 Average Packet Size: 0 4025,7785876993 975,31977281816 498917,78966329
 Avg Fwd Segment Size: 0 3015,290539206 677,11545442265 209471,06552625
 Avg Bwd Segment Size: 0 5011,0601589103 0,13888658922914 84,868581438742
 Fwd Header Length.1: -252927116050 155938 -108978779,43389 2,2632702194255e+18
Fwd Avg Bytes/Bulk: 0 0 0 0
 Fwd Avg Packets/Bulk: 0 0 0 0
 Fwd Avg Bulk Rate: 0 0 0 0
 Bwd Avg Bytes/Bulk: 0 0 0 0
 Bwd Avg Packets/Bulk: 0 0 0 0
Bwd Avg Bulk Rate: 0 0 0 0
```

```
Subflow Fwd Packets: 1 100148 4,9867302895593 59744,11976529
 Subflow Fwd Bytes: 0 15266416 2394,5418589839 44702193,051568
 Subflow Bwd Packets: 0 4602 0,022010557394035 2,2407867380174
 Subflow Bwd Bytes: 0 10993758 5,9608306810811 15927156,090198
Init_Win_bytes_forward: -1 65535 252,41338397022 2048607,70744
 Init_Win_bytes_backward: -1 65535 3,6470371638104 163837,9394615
 act_data_pkt_fwd: 0 5043 3,2690725458697 197,32990424709
 min_seg_size_forward: -1408237563 1480 -44832637,799091 4,5636394914208e+16
Active Mean: 0 61512893 613,23694593492 3881965631,4625
 Active Std: 0 48680467,951726 221,13649465056 593391745,67294
 Active Max: 0 72868426 955,45975144334 6524043531,8107
 Active Min: 0 61512893 491,00939729835 3446693501,6125
Idle Mean: 0 119219447 78286,542945442 1624504349465,3
 Idle Std: 0 66002917,440113 20509,107295958 176595890321,61
 Idle Max: 0 119219447 105849,27208648 2988065055026,4
 Idle Min: 0 119219447 58754,985696195 1041593394925,3
```

S'observen novament algunes característiques que prenen sempre valor 0. En concret:

`Bwd PSH Flags, Fwd URG Flags, Bwd URG Flags, FIN Flag Count, PSH Flag Count, ECE Flag Count, Fwd Avg Bytes/Bulk, Fwd Avg Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg Bytes/Bulk, Bwd Avg Packets/Bulk, Bwd Avg Bulk Rate.`

S'observen també errors en els camps següents:

`Flow Bytes/s Flow Packets/s`

(fet que apareixerà novament més endavant).

És possible calcular el rang de la matriu de les dades associada i veure així si hi ha variables linealment independents. És possible veure-ho amb Python amb el codi següent:

```
pima = pd.read_csv('b.txt', header=None, names=col_names ,
dtype={'Flow Bytes/s' : 'object', 'Flow Packets/s' : 'object',
 'SimillarHTTP' : 'object', 'Inbound' : 'object'})
print("# columns: " , len(columns))
numpy_matrix = pima.as_matrix(columns)
print ("Matrix shape: ", numpy_matrix.shape )
print("Matrix rank: " , np.linalg.matrix_rank(numpy_matrix))
```

On b.txt conté en aquest cas 50000 registres:

```
$ wc -l b.txt
50000 b.txt
```

I el resultat de l'execució és el següent:

```
$ python3 rank.py
# columns:  75
Matrix shape:  (50000, 75)
Matrix rank:  49
```

S'observa per tant que, del dataset inicial, que conté 75 columnes, el rang associat a la matriu passa a 49, fet que significa que 26 columnes són linealment dependents d'altres. Algunes d'aquestes columnes seran les que contenen valors en blanc. Malauradament, no s'implementa aquesta característica i és que no es calcula el rang associat amb totes les dades.

Un altre assumpte d'interès en l'apartat de la descripció estadística de les dades és l'elaboració de gràfiques, per exemple emprant boxplots. A continuació, es mostra un boxplot de la característica "Total Backward Packets".

Per a l'elaboració del gràfic, s'ha realitzat de la manera següent. Primer s'extreu la característica amb tots els valors:

```
$ cut -d "," -f 11 ../../originalData/01-12/all0112.csv
--output-delimiter=" " > data.txt
```

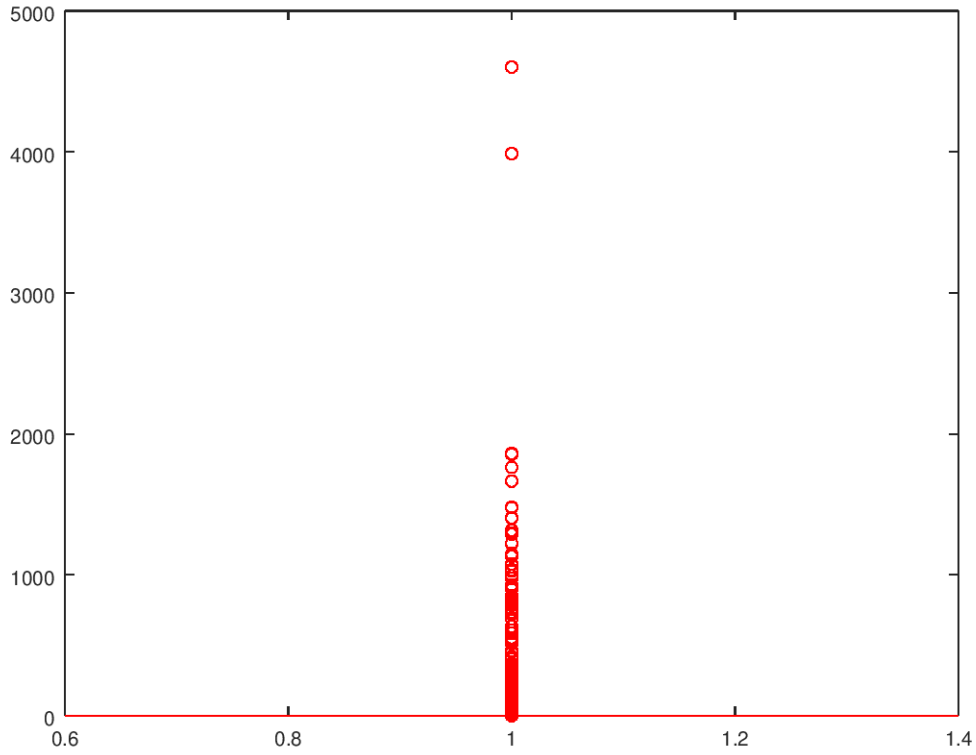
A continuació es pot generar la gràfica amb el programari de càlcul simbòlic Octave[27]:

```
octave:1> pkg load statistics
octave:2> data=load('data.txt')
octave:3> boxplot(data)
```

S'observa que la caixa del boxplot queda completament aixafada en l'eix de coordenades i és que normalment aquest valor deu ser zero. Es pot comprovar:

[27] GNU Octave, a Scientific Programming Language. `https://www.gnu.org/software/octave/`

Figure 4: Boxplot de la característica "Total backward Packets" de tots els fitxers del training day.




```
$ wc -l nou.txt
50063112 nou.txt

$ cat nou.txt | grep -x "0" | wc -l
49726420
```

Efectivament, en aquest cas, un 99% dels casos aquest valor és 0. De manera similar ocorre amb d'altres camps. No hem graficat tots els camps sistemàticament.

Finalment, comentar que la cerca d'"outliers" (registres amb valors anòmals) podria haver estat objecte d'estudi també en aquest apartat. Per exemple, a

través del càlcul de Z score[28], i suprimir-los del dataset d'entrenament. No s'ha realitzat en aquest TFM i és que no disposem de prou informació per a veure què pot ésser un valor anòmal.

4.4 Entrenament dels arbres de decisió

Implementar un arbre de decisió amb Python és relativament senzill. S'explica a continuació l'esquema general per a fer-ho. S'agreguen comentaris per a llegir el codi fàcilment[7]:

```
#Cal definir el nom de les columnes dels fitxers origen.
#La informació es pot extreure de la capçalera de cada fitxer.
col_names=['id', 'Flow ID','Source IP','Source Port','Destination IP',
'Destination Port','Protocol','Timestamp','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',
'Bwd Packet Length Min','Bwd Packet Length Mean','Bwd Packet Length Std',
'Flow Bytes/s','Flow Packets/s','Flow IAT Mean','Flow IAT Std','Flow IAT Max',
'Flow IAT Min','Fwd IAT Total','Fwd IAT Mean','Fwd IAT Std','Fwd IAT Max',
'Fwd IAT Min','Bwd IAT Total','Bwd IAT Mean','Bwd IAT Std','Bwd IAT Max',
'Bwd IAT Min','Fwd PSH Flags','Bwd PSH Flags','Fwd URG Flags','Bwd URG Flags',
'Fwd Header Length','Bwd Header Length','Fwd Packets/s','Bwd Packets/s',
'Min Packet Length','Max Packet Length','Packet Length Mean','Packet Length Std',
'Packet Length Variance','FIN Flag Count','SYN Flag Count','RST Flag Count',
'PSH Flag Count','ACK Flag Count','URG Flag Count','CWE Flag Count',
'ECE Flag Count','Down/Up Ratio','Average Packet Size','Avg Fwd Segment Size',
'Avg Bwd Segment Size','Fwd Header Length.1','Fwd Avg Bytes/Bulk',
'Fwd Avg Packets/Bulk','Fwd Avg Bulk Rate','Bwd Avg Bytes/Bulk',
'Bwd Avg Packets/Bulk','Bwd Avg Bulk Rate','Subflow Fwd Packets',
'Subflow Fwd Bytes','Subflow Bwd Packets','Subflow Bwd Bytes',
'Init_Win_bytes_forward','Init_Win_bytes_backward','act_data_pkt_fwd',
```

[28]"A z-score is the number of standard deviations away from a mean for a data point. A z-score helps point out how unusual or usual a data point is from the other values". Extret de manera literal d'aquí: `https://dfrieds.com/math/z-scores`. Aquesta xifra es pot calcular emprant Python directament. Veure `https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.zscore.html`

```
'min_seg_size_forward','Active Mean','Active Std','Active Max','Active Min',
'Idle Mean','Idle Std','Idle Max','Idle Min','SimillarHTTP','Inbound','Label']
#Es llegeix el fitxer origen en una estructura de dades de pandas.
#Pandas s'encarrega d'assignar cada columna del tipus que considera més adequat
#encara que hi ha alguns camps que és millor assignar-los com a "object"
# i és que, sinó, la lectura requereix de més temps i més memòria.
pima =     pd.read_csv('../01-12/Syn.csv', header=None, names=col_names ,
dtype={'Flow Bytes/s' : 'object', 'Flow Packets/s' : 'object',
'SimillarHTTP' : 'object', 'Inbound' : 'object'})
#s'indiquen a continuació quines seran les columnes que actuaran
# per a realitzar l'arbre
feature_cols = ['Protocol','Flow Duration','Total Fwd Packets','Total Backward Pac
'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','Bwd Packet Length Min','Bwd Packet Length Mean',
'Bwd Packet Length Std', 'Flow IAT Mean','Flow IAT Std','Flow IAT Max',
'Flow IAT Min','Fwd IAT Total','Fwd IAT Mean','Fwd IAT Std','Fwd IAT Max',
'Fwd IAT Min','Bwd IAT Total','Bwd IAT Mean','Bwd IAT Std','Bwd IAT Max',
'Bwd IAT Min','Fwd PSH Flags','Bwd PSH Flags','Fwd URG Flags','Bwd URG Flags',
'Fwd Header Length','Bwd Header Length','Fwd Packets/s','Bwd Packets/s',
'Min Packet Length','Max Packet Length','Packet Length Mean','Packet Length Std',
'Packet Length Variance','FIN Flag Count','SYN Flag Count','RST Flag Count',
'PSH Flag Count','ACK Flag Count','URG Flag Count','CWE Flag Count',
'ECE Flag Count','Down/Up Ratio','Average Packet Size','Avg Fwd Segment Size',
'Avg Bwd Segment Size','Fwd Header Length.1','Fwd Avg Bytes/Bulk',
'Fwd Avg Packets/Bulk','Fwd Avg Bulk Rate','Bwd Avg Bytes/Bulk',
'Bwd Avg Packets/Bulk','Bwd Avg Bulk Rate','Subflow Fwd Packets',
'Subflow Fwd Bytes','Subflow Bwd Packets','Subflow Bwd Bytes','Init_Win_bytes_forw
'min_seg_size_forward','Active Mean','Active Std','Active Max',
'Active Min','Idle Mean','Idle Std','Idle Max','Idle Min']

#es crea l'arbre de decisió en base a les columnes i també a l'etiqueta
# Label que conté la classificació. S'entrena amb les dades.
X = pima[feature_cols]
y = pima.Label
clf = DecisionTreeClassifier()
clf = clf.fit(X,y)
```

```
#Podem veure certa informació de l'arbre obtingut
print("Number of leaves" , clf.get_n_leaves())
print("Depth tree" , clf.get_depth())

#un cop aquí és possible escriure en mode text, generar una imatge,
# en fitxer PDF el resultat. Per a realitzar l'exportació en text
r = export_text(clf, feature_names=feature_cols)
print(r)
```

Respecte aquest algorisme DecisionTreeClassifier, observem els seus paràmetres:

```
DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None,
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0,
max_features=None, random_state=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
class_weight=None, presort=False)
```

Per defecte empra el criteri de Gini que, tal com s'ha comentat anteriorment, té un cost computacional menys costós. El procediment per a la generació dels arbres de decisió fou quelcom successiu, realitzant una sèrie de testos distints. S'indicaran més avall la sèrie de testos realitzats.

Finalment, respecte al tipus de dades que empra cada atribut, és possible veure-les amb les comandes de Python següents:

```
>>> pima.info(verbose=True)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4320541 entries, 0 to 4320540
Data columns (total 88 columns):
id                              int64
Flow ID                         object
Source IP                       object
Source Port                     int64
Destination IP                  object
Destination Port                int64
Protocol                        int64
Timestamp                       object
Flow Duration                   int64
```

```
Total Fwd Packets              int64
Total Backward Packets         int64
Total Length of Fwd Packets    float64
Total Length of Bwd Packets    float64
Fwd Packet Length Max          float64
Fwd Packet Length Min          float64
Fwd Packet Length Mean         float64
Fwd Packet Length Std          float64
Bwd Packet Length Max          float64
Bwd Packet Length Min          float64
Bwd Packet Length Mean         float64
Bwd Packet Length Std          float64
Flow Bytes/s                   object
Flow Packets/s                 object
Flow IAT Mean                  float64
Flow IAT Std                   float64
Flow IAT Max                   float64
Flow IAT Min                   float64
Fwd IAT Total                  float64
Fwd IAT Mean                   float64
Fwd IAT Std                    float64
Fwd IAT Max                    float64
Fwd IAT Min                    float64
Bwd IAT Total                  float64
Bwd IAT Mean                   float64
Bwd IAT Std                    float64
Bwd IAT Max                    float64
Bwd IAT Min                    float64
Fwd PSH Flags                  int64
Bwd PSH Flags                  int64
Fwd URG Flags                  int64
Bwd URG Flags                  int64
Fwd Header Length              int64
Bwd Header Length              int64
Fwd Packets/s                  float64
Bwd Packets/s                  float64
Min Packet Length              float64
Max Packet Length              float64
```

Packet Length Mean	float64
Packet Length Std	float64
Packet Length Variance	float64
FIN Flag Count	int64
SYN Flag Count	int64
RST Flag Count	int64
PSH Flag Count	int64
ACK Flag Count	int64
URG Flag Count	int64
CWE Flag Count	int64
ECE Flag Count	int64
Down/Up Ratio	float64
Average Packet Size	float64
Avg Fwd Segment Size	float64
Avg Bwd Segment Size	float64
Fwd Header Length.1	int64
Fwd Avg Bytes/Bulk	int64
Fwd Avg Packets/Bulk	int64
Fwd Avg Bulk Rate	int64
Bwd Avg Bytes/Bulk	int64
Bwd Avg Packets/Bulk	int64
Bwd Avg Bulk Rate	int64
Subflow Fwd Packets	int64
Subflow Fwd Bytes	int64
Subflow Bwd Packets	int64
Subflow Bwd Bytes	int64
Init_Win_bytes_forward	int64
Init_Win_bytes_backward	int64
act_data_pkt_fwd	int64
min_seg_size_forward	int64
Active Mean	float64
Active Std	float64
Active Max	float64
Active Min	float64
Idle Mean	float64
Idle Std	float64
Idle Max	float64
Idle Min	float64

```
SimillarHTTP                 object
Inbound                      object
Label                        object
dtypes: float64(43), int64(36), object(9)
memory usage: 2.8+ GB
```

4.5 Selecció de les característiques

Es comenta en aquest apartat les decisions preses per a la selecció dels atributs que es poden emprar per a l'entrenament de l'arbre de decisió. El dataset incorpora 88 columnes amb característiques diferents. El procés de selecció de les característiques a utilitzar consisteix en dues vies:

- Eliminar les característiques que, des d'un punt de vista semàntic, no aportin informació al procés. En aquest cas, s'han extret les característiques següents:

  ```
  id                          int64
  Flow ID                     object
  Source IP                   object
  Source Port                 int64
  Destination IP              object
  Destination Port            int64
  Protocol                    int64
  Timestamp                   object
  ```

  Es tracta de característiques que l'arbre no ha de tenir en compte i és que són totalment dependents del dataset i que, en un entorn diferent de detecció d'atacs, no es podrien emprar. La detecció dels atacs no hauria de dependre d'aquestes característiques.

- Poden succeir també problemes des de l'àmbit tècnic. S'observa que hi ha característiques que són de tipus "object". Aquests objectes es van intentar codificar emprant la classe LabelEncoder[29] de Python, però

[29] "LabelEncoder is a utility class to help normalize labels such that they contain only values between 0 and n_classes1. It can also be used to transform non-numerical labels (as long as they are hashable and comparable) to numerical labels". Extret de manera literal de `https://scikit-learn.org/stable/modules/preprocessing_targets.html#preprocessing-targets`

en tots els testos realitzats, han aparegut problemes constantment. Es recorda, tal com s'esmenta en 3.3 que la llibreria utilitzada no suporta variables categòriques. Caldria, potser, filtrar les dades millor i depurar possibles valors erronis. Es tracta dels camps següents:

```
Flow Bytes/s                    object
Flow Packets/s                  object
SimillarHTTP                    object
Inbound                         object
```

- Finalment, eliminar valors no significatius estadísticament parlant (per exemple, tots els valors que sempre es mostren a 0 o que són linealment dependents de d'altres columnes). No s'ha extret cap característica per aquest motiu, encara que hagués estat adequat fer-ho.

4.6 Testos i bancs de proves

Aquest treball consisteix en estudiar la idoneïtat d'entrenar arbres de decisió a partir d'un dataset concret relacionat amb la seguretat informàtica. Es realitzen amb aquest objectiu diversos bancs de proves que permeten aprofundir en el procés i l'anàlisi de resultats. A mesura que es realitzen proves s'obtenen algunes conclusions. De bones a primeres, tampoc no es coneix el cost computacional de cap dels algorismes, així que es parteix d'exemples concrets analitzats de manera individual. Així doncs, no s'intenta obtenir un arbre de decisió emprant tots els registres de tot el dataset(la màquina emprada tampoc no ho suporta, tal com s'esmentarà més endavant), sinó que es realitzen testos més breus. En concret, es realitzen les proves següents:

En el test 1 es generen arbres de decisió individuals emprant solament els registres del training day que no disposen de correspondència amb els del testing day i realitzant la verificació amb una partició de la mostra (70% dels registres es fan servir per entrenar l'arbre i, el 30% restant per a verificar-lo).

En el test 2 es genera un sol fitxer que conté 2100 registres amb totes les etiquetes presents en el testing day i es verifica l'arbre amb un dels fitxers també del mateix testing day.

En el test 3 es generen diversos arbres a partir de fitxers concrets del training

day i es comparen amb els fitxers homòlegs del testing day.

En el test 4 s'entrena un arbre de decisió emprant un dataset que contingui tots els registres comuns entre el training day i també el testing day. S'inicia el procediment creant dos fitxers, un per al training day i un altre pel testing day, amb tots els registres. L'objectiu és solucionar els problemes ocorreguts en els testos anteriors (en dos hi havia manca de registres diferents i en l'altre massa poc registres).

En el test 5 es forçarà realitzar un arbre més reduït, amb més mostres per fulla, forçant l'algorisme a disposar de més classes en cada fulla.

En el test 6 s'estudiarà la idoneïtat d'aplicar la reducció de la dimensionalitat emprant anàlisis de components principals (anomenat també PCA)[6].

En el test 7 es genera primer un sol fitxer amb més de 50 milions de registres del training day i s'intenta generar un arbre amb aquest dataset.

En el test 8 es genera un sol fitxer amb 1 milió de registres diferents en el traninig day. Es tracta d'un fitxer amb més entropia que els anteriors - disposa de registres amb les classificacions següents:

BENIGN DrDoS_DNS, DrDoS_LDAP, DrDoS_MSSQL, DrDoS_NetBIOS, DrDoS_NTP, DrDoS_SNMP, DrDoS_SSDP, DrDoS_UDP, Syn,TFTP,UDP-lag, WebDDoS. Llavors s'entrena un arbre amb el 70% dels registres i es compara amb el 30% restant.

En el test 9 es realitza de nou una altra prova emprant PCA.

En el test 10 es realitza una prova emprant Random Forests.

4.6.1 Test 1

En aquest primer test s'empren solament els fitxers del training day que no disposen de correspondència en el testing day. Es tracta dels següents:

DrDoS_DNS, DrDoS_NTP, DrDoS_SNMP, DrDoS_SSDP, DrDoS_UDP,

TFTP, UDP-lag, WebDDoS

Es generaran arbres de decisió emprant el 70% dels registres de cada fitxer i el 30% restant s'emprarà per a comparar la validesa i encert de l'arbre resultant. Per a realitzar aquesta tasca, s'empren les línies de codi similars a les següents:

```
# 70% dels registres per a training i el 30% restant per a test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1)
# S'entrena solament aquest train
clf = clf.fit(X_train,y_train)
print("Number of leaves" , clf.get_n_leaves())
print("Depth tree" , clf.get_depth())
#Es realitza la predicció emprant el test
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

Emprant el script següent:

```
$ more executaTest1.sh
#!/bin/bash
echo "DoS_DNS"
time python3 makeTest1DrDoS_DNS.py
echo "DoS_NTP"
time python3 makeTest1DrDoS_NTP.py
echo "DoS_SNMP"
time python3 makeTest1DrDoS_SNMP.py
echo "DoS_SSDP"
time python3 makeTest1DrDoS_SSDP.py
echo "DoS_UDP"
time python3 makeTest1DrDoS_UDP.py
```

S'obtenen els resultats següents:

```
DoS_DNS
Importing training test...
Spliting data...
Fitting the tree...
Number of leaves 73
```

```
Depth tree 16
Accuracy: 0.999989489754
real 3m18,446s
user 0m58,602s
sys 0m9,189s

DoS_NTP
Importing training test...
Spliting data...
Fitting the tree...
Number of leaves 97
Depth tree 14
Accuracy: 0.999895919782
real 0m21,006s
user 0m18,930s
sys 0m1,804s

DoS_SNMP
Importing training test...
Spliting data...
Fitting the tree...
Number of leaves 42
Depth tree 10
Accuracy: 0.999992895957
real 3m51,433s
user 0m58,830s
sys 0m9,279s

DoS_SSDP
Importing training test...
Spliting data...
Fitting the tree...
Number of leaves 32
Depth tree 11
Accuracy: 0.999987235341

real 0m34,579s
user 0m30,424s
```

```
sys 0m3,305s
DoS_UDP
Importing training test...
Spliting data...
Fitting the tree...
Number of leaves 43
Depth tree 11
Accuracy: 0.999988310817
real 0m44,742s
user 0m40,529s
sys 0m3,613s
```

S'observa que es generen arbres de decisió per a cada conjunt i que es realitza una predicció del 30% de les dades encertant correctament el resultat. Malgrat això, abans de treure conclusions precipitades, es calcula el percentatge que disposa cada fitxer per a cada etiqueta:

```
$ cut -d "," -f 88 DrDoS_DNS.csv | sort | uniq
BENIGN
DrDoS_DNS
$ cat DrDoS_DNS.csv | grep BENIGN | wc -l
3402
$ cat DrDoS_DNS.csv | grep DrDoS_DNS | wc -l
5071011
$ bc -l <<< '3402/(3402+5071011)'
.00067042237200637788
```

En aquest primer cas s'observa que el nombre de tràfic benigne és irrisori. Això significa que, prenent un registre de manera aleatòria i indicant que es tracta de DrDoS_DNS, la probabilitat d'encert és altíssima. Malgrat es va veure que l'arbre s'entrenava i s'obtenia un bon nivell d'exactitud, els resultats no són estadísticament significatius i és que el nombre de registres de tràfic benigne és massa baix. Realitzem a continuació el càlcul de l'entropia per aquests registres. Es realitza aquest càlcul de nou amb Octave:

```
0.99933 * log2(1/0.99933) + 0.00067042237200637788*log2(1/0.00067042237200637788)
ans =  0.0080343
```

Observem que l'entropia és tant baixa (0.0080343) que la probabilitat d'encertar fent-ho de manera completament aleatòria és altíssima. El grau de "sorpresa" és molt baix. Succeeix el mateix en els casos següents, on es calcula solament el tant per cent del tràfic benigne:

```
cut -d "," -f 88 DrDoS_NTP.csv | sort | uniq
BENIGN
DrDoS_NTP
cat DrDoS_NTP.csv | grep BENIGN | wc -l
14365
cat DrDoS_NTP.csv | grep DrDoS_NTP | wc -l
1202642
bc -l <<< '14365/(14365+1202642)'
.01180354755560157008

cut -d "," -f 88 DrDoS_SNMP.csv | sort | uniq
BENIGN
DrDoS_SNMP
$ cat DrDoS_SNMP.csv | grep BENIGN | wc -l
1507
$ cat DrDoS_SNMP.csv | grep DrDoS_SNMP | wc -l
5159870
$ bc -l <<< '1507/(1507+5159870)'
.00029197634662222891

$ cat DrDoS_SSDP.csv | grep BENIGN | wc -l
763
$ cat DrDoS_SSDP.csv | grep DrDoS_SSDP | wc -l
2610611
$ bc -l <<< '763/(763+2610611)'
.00029218334868923409

$ cut -d "," -f 88 DrDoS_UDP.csv | sort | uniq
BENIGN
DrDoS_UDP
$ cat DrDoS_UDP.csv | grep BENIGN | wc -l
2157
$ cat DrDoS_UDP.csv | grep DrDoS_UDP | wc -l
```

```
3134645
$ bc -l <<< '2157/(2157+3134645)'
.00068764301986545532

$ cut -d "," -f 88 TFTP.csv | sort | uniq
BENIGN
TFTP
$ cat TFTP.csv | grep BENIGN | wc -l
25247
$ cat TFTP.csv | grep TFTP | wc -l
20082580
$ bc -l <<< '25247/(25247+20082580)'
.00125558072485903126

$ cut -d "," -f 88 UDPLag.csv | sort | uniq
BENIGN
UDP-lag
WebDDoS
$ cat UDPLag.csv | grep BENIGN | wc -l
3705
$ cat UDPLag.csv | grep UDP-lag | wc -l
366461
$ cat UDPLag.csv | grep WebDDoS | wc -l
439
```

On el tràfic benigne amb l'etiqueta BENIGN representa solament el:

```
$ bc -l <<< '3705/(3705+366461+439)'
.00999716679483547172
```

el tràfic amb UDP-lag representa:

```
$ bc -l <<< '366461/(3705+366461+439)'
.98881828361732842244
```

i, finalment, aquest WebDoS representa:

```
$ bc -l <<< '439/(3705+366461+439)'
.00118454958783610582
```

La significació estadística dels arbres obtinguts no és adequada: caldria entrenar un arbre que partís d'un volum més alt d'indeterminació i assegurar-se que realment funciona amb un conjunt de dades també més indeterminades.

Conclusió del primer test: cal entrenar arbres a partir de dades amb més entropia (més balancejats), on hi hagi més etiquetes diferents. En aquest cas s'ha generat un arbre diferent per a cada fitxer, així que no és útil.

4.6.2 Test 2

En aquest test es genera un fitxer test100.csv que conté un centenar de registres de cada etiqueta (benigne i atacs) presents en el training day. S'entrena un arbre tal com s'ha mostrat en el procés anterior i es compara amb un dels fitxers del testing day. Per a generar el fitxer s'empra el script següent:

```
$ more generaTest.sh
#!/bin/bash
cat DrDoS_DNS.csv | grep DrDoS_DNS | head -100 >> test100.csv
cat DrDoS_DNS.csv | grep BENIGN | head -100 >> test100.csv
cat DrDoS_LDAP.csv | grep DrDoS_LDAP | head -100 >> test100.csv
cat DrDoS_LDAP.csv | grep BENIGN | head -100 >> test100.csv
cat DrDoS_MSSQL.csv | grep DrDoS_MSSQL | head -100 >> test100.csv
cat DrDoS_MSSQL.csv | grep BENIGN | head -100 >> test100.csv
cat DrDoS_NetBIOS.csv | grep DrDoS_NetBIOS | head -100 >> test100.csv
cat DrDoS_NetBIOS.csv | grep BENIGN | head -100 >> test100.csv
cat DrDoS_NTP.csv | grep DrDoS_NTP | head -100 >> test100.csv
cat DrDoS_NTP.csv | grep BENIGN | head -100 >> test100.csv
cat DrDoS_SNMP.csv | grep DrDoS_SNMP | head -100 >> test100.csv
cat DrDoS_SNMP.csv | grep BENIGN | head -100 >> test100.csv
cat DrDoS_SSDP.csv | grep DrDoS_SSDP | head -100 >> test100.csv
cat DrDoS_SSDP.csv | grep BENIGN | head -100 >> test100.csv
cat DrDoS_UDP.csv | grep DrDoS_UDP | head -100 >> test100.csv
cat DrDoS_UDP.csv | grep BENIGN | head -100 >> test100.csv
cat Syn.csv | grep Syn | head -100 >> test100.csv
cat Syn.csv | grep BENIGN | head -100 >> test100.csv
cat TFTP.csv | grep TFTP | head -100 >> test100.csv
```

```
cat TFTP.csv | grep BENIGN | head -100 >> test100.csv
cat UDPLag.csv | grep UDPLag | head -100 >> test100.csv
cat UDPLag.csv | grep BENIGN | head -100 >> test100.csv

$ wc -l test100.csv
2100 test100.csv
```

S'entrena un arbre i es compara amb el fitxer Syn.csv del mateix training day:

```
$ time python3 test2.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 85
Depth tree 19
Accuracy results: 0.141011991677
Accuracy results: 223177
real 0m14,245s
user 0m13,069s
sys 0m1,866s
```

La velocitat de càlcul és alta (14 segons), però el grau d'exactitud és massa baix. Es realitza la mateixa predicció però emprant un altre fitxer (concretament, el fitxer 01-12/DrDoS_SSDP.csv):

```
$ time python3 test2.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 85
Depth tree 19
Accuracy results: 0.0147937445958
Accuracy results: 38632
real 0m24,514s
user 0m22,336s
sys 0m2,904s
```

Resultats similars a l'anterior: càlcul ràpid però grau d'exactitud massa baix. Cal entrenar un arbre amb més registres.

Conclusió del segon test: cal entrenar arbres amb força registres, malgrat el temps de càlcul sigui alt.

4.6.3 Test 3

El tercer test consisteix en entrenar un arbre amb els registres d'un dels fitxers (per exemple, Syn.csv) presents en el training day (que, es recorda, conté registres amb tràfic benigne i també tràfic atacant). Amb l'arbre entrenat, es compara amb les dades del fitxer homòleg del testing day i es verifica el grau d'encert. A continuació es mostra part del codi per aquest test:

```
#s'importa el training data set per a l'entrenament de l'arbre
print("Importing training test....")
pima = pd.read_csv('../01-12/Syn.csv', header=None, names=col_names,
dtype={'Flow Bytes/s' : 'object', 'Flow Packets/s' : 'object',
'SimillarHTTP' : 'object', 'Inbound' : 'object'})

#s'importa el testing data set per a la posterior validació de l'arbre entrenat
print("Importing testing test....")
pimaTest = pd.read_csv('../03-11/Syn.csv', header=None, names=col_names,
dtype={'Flow Bytes/s' : 'object', 'Flow Packets/s' : 'object',
'SimillarHTTP' : 'object', 'Inbound' : 'object'})
# Selecció de les característiques i de la variable Label
# que indica la classificació de la mostra
X = pima[feature_cols]
y = pima.Label
XTest = pimaTest[feature_cols]
yTest = pimaTest.Label
# Es crea i s'entrena l'arbre de decisió
clf = DecisionTreeClassifier()
print("Fitting the decision tree...")
clf = clf.fit(X,y)
# S'obté un resultat. Es mostren dades descriptives de l'arbre
print("Number of leaves" , clf.get_n_leaves())
print("Depth tree" , clf.get_depth())
```

```
#Es prediu una resposta per al test dataset. Es mostra el grau d'accuracy
y_Test_pred = clf.predict(XTest)
print("Accuracy results:",metrics.accuracy_score(yTest, y_Test_pred))
```

S'executa aquest algorisme amb el dataset d'atacs de Syn i s'obté el resultat següent:

```
$ python3 makeSynDecisionTree.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 26
Depth tree 12
Accuracy results: 0.882556374306
```

De manera similar a l'anterior, es genera també per al NetBIOS:

```
$ python3 makeNetbiosDecisionTree.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 45
Depth tree 13
Accuracy results: 0.999911745106
```

Per a LDAP:

```
$ python3 makeLDAPDecisionTree.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 37
Depth tree 10
Accuracy results: 0.903310754985
```

I, finalment, per MSSQL:

```
$  python3 makeMSSQLDecisionTree.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 45
Depth tree 10
Accuracy results: 0.998122506616
```

Caldria però analitzar abans d'extreure conclusions precipitades, el nivell d'entropia de cada dataset. Es calcula primer el de Syn:

```
$ cat ../01-12/Syn.csv | wc -l
1582681
$ cat ../01-12/Syn.csv | grep Syn | wc -l
1582289
$ cat ../01-12/Syn.csv | grep BENIGN | wc -l
392
```

La classe BENIGN però ocupa un total de:

```
octave:2> 392 / (1582681)
ans =    2.4768e-04
```

Es calcula en el testing day:

```
$ cat ../03-11/Syn.csv | wc -l
4320541
$ cat ../03-11/Syn.csv | grep Syn | wc -l
4284751
$ cat ../03-11/Syn.csv | grep BENIGN | wc -l
35790
```

El percentatge en el testing day de registres BENIGN és solament del 0.0082837 així que, la fiabilitat del arbre no és estadísticament significativa i és que és pràcticament segur que, si l'arbre respongués que qualsevol registre se'l categoritza amb un Syn, la resposta seria correcta.

Conclusió del tercer test: es cau amb el mateix parany que en el primer test. Cal entrenar arbres amb més entropia, amb etiquetes diferents.

4.6.4 Test 4

En aquest test s'entrena un arbre de decisió emprant un dataset que contingui tots els registres comuns entre el training day i també el testing day. S'inicia el procediment creant dos fitxers, un per a cada dia, amb tots els registres (el primer fitxer s'anomenarà TraniningFour i, el segon, testingFour):

```
$  cat DrDoS_LDAP-training.csv DrDoS_MSSQL-training.csv
 DrDoS_NetBIOS-training.csv Syn.csv > TrainingFour.csv
$ wc -l TrainingFour.csv
12383707 TrainingFour.csv

$  cat LDAP.csv MSSQL.csv NetBIOS.csv Syn.csv > testingFour.csv
$ wc -l testingFour.csv
15665460 testingFour.csv
```

Es calculen a continuació quants registres hi ha de cada classe en cada fitxer així com el nivell d'entropia:

```
$ cat TrainingFour.csv | grep BENIGN | wc -l
5717
$ cat TrainingFour.csv | grep LDAP | wc -l
2179930
$ cat TrainingFour.csv | grep MSSQL | wc -l
4522492
$ cat TrainingFour.csv | grep NetBIOS | wc -l
4093279
$ cat TrainingFour.csv | grep Syn | wc -l
1582289
```

La classe BENIGN representa un total de 4.6165e-04 casos (training day). La classe LDAP representa un total de 0.17603 casos (training day). La classe MSSQL representa un total de 0.36520 casos (training day). La classe NetBIOS representa un total de 0.33054 casos (training day). La classe Syn representa un total de 0.12777 casos (training day). Certament, 0.12777 + 0.33054 + 0.36520 + 0.17603 + 4.6165e-04 = 1.

El càlcul de l'entropia en el fitxer de training day és el següent:

```
octave:2> 0.12777 * log2(1/0.12777) + 0.33054 * log2(1/0.33054) +
 0.36520*log2(1/0.36520) + 0.17603*log2(1/0.17603) +
  4.6165e-04*log2(1/4.6165e-04)
```

L'entropia per al fitxer de training day és 1.8842

Realitzem ara el mateix procediment per al fitxer de testingFour:

```
$ wc -l  testingFour.csv
15665460 testingFour.csv
$ cut -d "," -f 88 testingFour.csv | uniq | sort | uniq
BENIGN
LDAP
MSSQL
NetBIOS
Syn
$ cat testingFour.csv | grep BENIGN | wc -l
45029
$ cat testingFour.csv | grep LDAP | wc -l
1915122
$ cat testingFour.csv | grep MSSQL | wc -l
5763061
$ cat testingFour.csv | grep NetBIOS | wc -l
3657497
$ cat testingFour.csv | grep Syn | wc -l
4284751
```

La classe BENIGN representa un total de 0.0028744 casos (testing day).
La classe LDAP representa un total de 0.12225 casos (testing day).
La classe MSSQL representa un total de 0.36788 casos (testing day).
La classe NetBIOS representa un total de 0.23348 casos (testing day).
La classe Syn representa un total de 0.27352 casos (testing day)
Certament, $0.0028744 + 0.12225 + 0.36788 + 0.23348 + 0.27352 = 1$
Calculem l'entropia dels registres d'aquest fitxer:

```
octave:7> 0.0028744 * log2(1/0.0028744) + 0.12225 * log2(1/0.12225) +
 0.36788 * log2(1/0.36788) + 0.23348 * log2(1/0.23348) +
 0.27352*log2(1/0.27352)
ans =  1.9272
```

S'inicia el procés d'entrenament de l'arbre emprant els registres del training day i comparant-los amb els del testing day. En la primera execució, després de cinquanta minuts de càlcul, apareix un error de memòria:

```
$  time python3 makeFourDecisionTree.py
Importing training test....
Importing testing test....
Traceback (most recent call last):
  File "makeFourDecisionTree.py", line 19, in <module>
    pimaTest = pd.read_csv('../03-11/testingFour.csv', header=None, names=col_name
  File "/usr/lib/python3/dist-packages/pandas/io/parsers.py", line 709, in parser_
  File "/usr/lib/python3/dist-packages/pandas/io/parsers.py", line 455, in _read
  File "/usr/lib/python3/dist-packages/pandas/io/parsers.py", line 1069, in read
    ret = self._engine.read(nrows)
  File "/usr/lib/python3/dist-packages/pandas/io/parsers.py", line 1839, in read
    data = self._reader.read(nrows)
  File "pandas/_libs/parsers.pyx", line 902, in pandas._libs.parsers.TextReader.re
  File "pandas/_libs/parsers.pyx", line 952, in pandas._libs.parsers.TextReader._r
  File "pandas/_libs/parsers.pyx", line 2237, in pandas._libs.parsers._concatenate
MemoryError
real 50m22,143s
user 3m41,583s
sys 2m8,543s
```

Cal agregar més memòria d'intercanvi (swap) en el sistema:

```
$ free -h
              total        used        free      shared  buff/cache   available
Mem:            15G        923M         13G        3,9M        1,0G         14G
Swap:           15G        799M         15G
```

Es disposa en el sistema d'un fitxer de 100Gb preparat per a hostatjar més espai de swap:

```
$  ls -lh
-rw------- 1 root root 100G d'oct 27 07:12 /media/gerard/62551692-00cb-4c2c-
a96d-d54d279c07ba/swapfile

$  swapon /media/gerard/62551692-00cb-4c2c-
```

```
a96d-d54d279c07ba/swapfile

$  free -h
              total        used        free      shared  buff/cache   available
Mem:            15G        956M         13G        3,9M        1,0G          14G
Swap:          115G        798M        115G
```

I s'executa el procés de nou:

```
$ time python3 makeFourDecisionTree.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 22257
Depth tree 47
Accuracy results: 0.987110305092
real 133m38,061s
user 6m43,020s
sys 5m14,232s
```

Després de dues hores de càlcul el sistema obté un arbre amb una profunditat de 47 i un total de 22257 fulles. S'entrena l'arbre de decisió emprant el fitxer TrainingFour, que conté cinc etiquetes de tràfic diferent i un total de 12383707 registres i es valida amb el fitxer del dia de test (testingFour), que conté un total de 15665460 registres. El grau d'encert de l'arbre és de més d'un 98%.

Conclusió del quart test: aquest arbre és coherent i interessant. Caldria però reduir la complexitat en dos sentits: pel que fa a la velocitat de càlcul i de memòria (es recorda que el càlcul ha trigat més de dues hores i que ha estat necessari augmentar la quantitat de memòria d'intercanvi) i també reduir l'arbre (el cost computacional de recórrer per a posterior prediccions un arbre amb 22.257 fulles és alt).

4.6.5 Test 5

Partint dels testos anteriors, s'intenta implementar un arbre més reduït, amb menys nombre de nodes i amb menys profunditat. La pregunta però és: serà

igualment curós ? Quin canvi en el nivell d'exactitud pot produir un arbre així ?
Per tal d'implementar aquesta opció, es tracta de realitzar exactament el mateix codi que l'anterior però agregant el paràmetre següent en la definició de l'arbre:

```
clf = DecisionTreeClassifier(min_samples_leaf=1200)
```

Aquesta paràmetre forçarà l'arbre a disposar en cada fulla d'almenys 1200 registres. Es realitzen diversos testos realitzant canvis en aquest paràmetre min_samples_leaf. Els resultats són els següents:

```
#En aquest cas, min_samples_leaf=1200
$ python3 makeFourDecisionTree.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 2069
Depth tree 33
Accuracy results: 0.987909004906
```

S'observa fins i tot un grau d'exactitud major que en la prova anterior! Es recorda que les dades obtingudes anteriorment eren les següents:

```
Number of leaves 22257
Depth tree 47
Accuracy results: 0.987110305092
```

Es realitzen encara més arbres més exigents:

```
clf = DecisionTreeClassifier(min_samples_leaf=12000)
```

```
$ time python3 makeFourDecisionTree.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 346
Depth tree 21
```

```
Accuracy results: 0.985671343197
real 133m31,594s
user 7m0,777s
sys 4m58,984s
```

S'obté un arbre més reduït encara i amb un grau d'exactitud que segueix sent força bo. Es segueix amb més exigència:

```
clf = DecisionTreeClassifier(min_samples_leaf=1200000)
```

```
$ time python3 makeFourDecisionTree.py
Importing training test....
Importing testing test....
Fitting the decision tree...
Number of leaves 7
Depth tree 4
Accuracy results: 0.985737412116
```

S'obté un arbre encara més reduït i amb un grau d'exactitud que segueix essent molt bo. En concret, dels 15665460 registres presents en el testing-Four.csv, n'encerta 15442030 i en falla per tant,

```
octave:2> 15665460 -15442030
ans =  223430
```

que representa un:

```
octave:3> 223430 / 15665460
ans =  0.014263
```

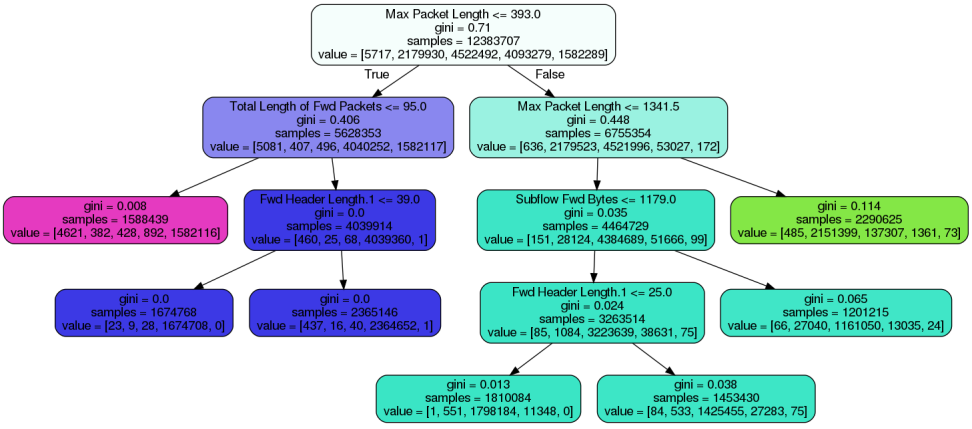
d'errors, fet que ja concorda amb el grau d'exactitud mostrat per l'algorisme (0.9857). El nombre de valors predits de manera correcta es pot mostrar amb la comanda següent:

```
print("Accuracy results:",metrics.accuracy_score(yTest,
y_Test_pred, normalize=False))
```

Es mostra a continuació una representació textual de l'arbre obtingut i també una representació gràfica:

```
Accuracy results:
|--- Max Packet Length <= 393.00
|   |--- Total Length of Fwd Packets <= 95.00
|   |   |--- class: Syn
|   |--- Total Length of Fwd Packets >  95.00
|   |   |--- Fwd Header Length.1 <= 39.00
|   |   |   |--- class: NetBIOS
|   |   |--- Fwd Header Length.1 >  39.00
|   |   |   |--- class: NetBIOS
|--- Max Packet Length >  393.00
|   |--- Max Packet Length <= 1341.50
|   |   |--- Subflow Fwd Bytes <= 1179.00
|   |   |   |--- Fwd Header Length.1 <= 25.00
|   |   |   |   |--- class: MSSQL
|   |   |   |--- Fwd Header Length.1 >  25.00
|   |   |   |   |--- class: MSSQL
|   |   |--- Subflow Fwd Bytes >  1179.00
|   |   |   |--- class: MSSQL
|   |--- Max Packet Length >  1341.50
|   |   |--- class: LDAP
```

Figure 5: Decision tree generat en aquest cinquè test



Conclusió del cinquè test: s'ha generat un arbre molt més reduït, amb el mateix cost computacional que l'anterior i obtenint nivells

d'exactitud força alts. S'ha incrementat el nivell d'entropia respecte en els testos 1 i 3, però no amb l'entropia general de tot el dataset.

4.6.6 Test 6

En aquest test s'aplica la tècnica de PCA (anàlisis de components principals) amb la idea de reduir la dimensionalitat de la mostra. Si, en comptes d'haver de tractar més de 80 característiques es pogués entrenar un arbre amb moltes menys columnes, el procés d'entrenament seria molt més ràpid des d'un punt de vista de càlcul computacional.

S'utilitzarà un dels fitxers del training day (en aquesta cas s'ha escollit el fitxer UDPLag.csv), es reduirà amb components principals, es realitzarà una partició per a la predicció (70% per a l'entrenament de l'arbres versus el 30% per a la predicció). Certament que es cau de nou amb el mateix parany que en els testos 1 i 3 (manca d'entropia en les dades per a l'entrenament de l'arbre), però cal estudiar la idoneïtat de seguir amb aquesta via o no. El codi per aquest test és similar a l'anterior encara que:

```
#Cal importar noves llibreries
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
#S'estandarditza cada característica restant-hi
# la seva mitjana i obtenint la variança.
#En concret, es realitza el següent: z = (x - u) / s,
# on u és la mitjana de la característica i s és #la desviació estàntard1
train_img = StandardScaler().fit_transform(X)
#Es desitja crear components suficients per a explicar
# el 95% de la variança
pca = PCA(.95)
principalComponents = pca.fit_transform(train_img)

#És possible veure tant el nombre de components
print ("El nombre de components és " , pca.n_components_ )
# com l'explicació de la variança:
print ("Explained variance es " , pca.explained_variance_ratio_ )
```

El resultat obtingut és:

```
$ time python3 makeDecTreeWithPCA.py
Importing training test....
explained variance es  [ 0.19930499  0.17426174  0.12122758  0.05509059
0.04711211  0.04444928 0.04034494  0.03400637  0.03005071
  0.02898279  0.02699579  0.02413169 0.02093199  0.01959971  0.01606371
  0.01559778  0.01431086  0.01400627  0.01149175  0.01069707  0.01009023]
El nombre de components és  21
Number of leaves 88
Depth tree 12
Accuracy results: 0.999712183627
real 0m7,583s
user 0m22,804s
sys 0m2,492s
```

Conclusió test 6: malgrat s'és conscient que l'arbre generat no és útil (pel mateix motius que els testos 1 i 3, on manquen valors diferents en l'arbre inicial), el test promet: de 76 columnes inicials en aquest cas es passa a 21. Es recomana explorar aquesta via.

4.6.7 Test 7

Es genera un sol fitxer que conté tots els registres del training day:

```
$ cat DrDoS_DNS.csv DrDoS_LDAP.csv DrDoS_MSSQL.csv DrDoS_NetBIOS.csv
DrDoS_NTP.csv DrDoS_SNMP.csv DrDoS_SSDP.csv DrDoS_UDP.csv Syn.csv
TFTP.csv UDPLag.csv > AllTrainingDay.csv
```

Conté exactament el nombre següent de registres:

```
$ cat ../../01-12/AllTrainingDay.csv | wc -l
50063114
```

I s'executa l'algorisme que entrena l'arbre a partir del 70% dels registres i avaluarà el seu nivell d'exactitud amb el 30% restant:

```
$ time python3 makeAllTrainingDay.py
Importing training test....
Matat
real 380m,814s
user 10m29,256s
sys 2m2,801s
```

Després de més de 6 hores de càlcul, el procés apareix "Matat", sense explicacions. Prèviament s'havia augmentat la quantitat de memòria d'intercanvi swap (més detalls sobre aquest assumpte en un test anterior, on un dels processos moria per manca de memòria):

```
$ free -h
              total        used        free      shared  buff/cache   available
Mem:            15G        954M        203M        5,0M         14G          14G
Swap:          115G        771M        115G
```

Conclusió: l'algorisme en aquesta màquina no ho suporta tot. Hi ha un llindar en el nombre de registres que es poden processar.

4.6.8 Test 8

Es genera un sol fitxer que conté 1 milió de registres de cada fitxer del training day. S'entrena un arbre a partir del 70% dels registres i es prediu el resultat partint del 30% restant. Es genera primer aquest fitxer inicial:

```
$ head -1000000 DrDoS_DNS.csv >> test8.csv
$ head -1000000 DrDoS_LDAP.csv >> test8.csv
$ head -1000000 DrDoS_MSSQL.csv >> test8.csv
$ head -1000000 DrDoS_NetBIOS.csv >> test8.csv
$ head -1000000 DrDoS_NTP.csv >> test8.csv
$ head -1000000 DrDoS_SNMP.csv >> test8.csv
$ head -1000000 DrDoS_SSDP.csv >> test8.csv
$ head -1000000 DrDoS_UDP.csv >> test8.csv
$ head -1000000 DrDoS_Syn.csv >> test8.csv
$ head -1000000 Syn.csv >> test8.csv
$ head -1000000 TFTP.csv >> test8.csv
$ head -1000000 UDPLag.csv >> test8.csvg
$ wc -l test8.csv
10370605 test8.csv
```

Observem a continuació exactament els valors que conté la mostra:

```
$ cut -d "," -f 88 test8.csv | uniq | sort | uniq
BENIGN
DrDoS_DNS
```

```
DrDoS_LDAP
DrDoS_MSSQL
DrDoS_NetBIOS
DrDoS_NTP
DrDoS_SNMP
DrDoS_SSDP
DrDoS_UDP
Syn
TFTP
UDP-lag
WebDDoS
```

i en calculem la quantitat de cadascun d'ells (amb l'objectiu de calcular després l'entropia):

```
$ cat test8.csv | grep BENIGN | wc -l
25530
$ cat test8.csv | grep DrDoS_DNS | wc -l
997994
$ cat test8.csv | grep DrDoS_LDAP | wc -l
999244
$ cat test8.csv | grep DrDoS_MSSQL | wc -l
998987
$ cat test8.csv | grep DrDoS_NetBIOS | wc -l
999146
$ cat test8.csv | grep DrDoS_NTP | wc -l
986449
$ cat test8.csv | grep DrDoS_SNMP | wc -l
999296
$ cat test8.csv | grep DrDoS_SSDP | wc -l
999700
$ cat test8.csv | grep DrDoS_UDP | wc -l
998868
$ cat test8.csv | grep Syn | wc -l
999655
$ cat test8.csv | grep TFTP | wc -l
998834
$ cat test8.csv | grep UDP-lag | wc -l
```

```
366461
$ cat test8.csv | grep WebDDoS | wc -l
439
```

Es calcula a continuació el grau d'entropia de les dades obtingudes:

```
octave:1> (25530/10370603) * log2(10370603/25530) +
(997994/10370603)*log2(10370603/9979949) +
(997994/10370603)*log2(10370603/997994) +
(998987/10370603)*log2(10370603/998987) +
(999146/10370603)*log2(10370603/999146) +
(986449/10370603)*log2(10370603/986449) +
(999296/10370603)*log2(10370603/999296) +
(999700/10370603)*log2(10370603/999700) +
(998868/10370603)*log2(10370603/998868) +
(999655/10370603)*log2(10370603/999655) +
(998834/10370603)*log2(10370603/998834) +
(366461/10370603)*log2(10370603/366461) +
(439/10370603)*log2(10370603/439)
ans =  3.1222
```

I es genera l'arbre:

```
$ time python3 makeSelectionTrainingDay.py
Importing training test...
Spliting data...
Fitting the tree...
Number of leaves 504
Depth tree 22
Accuracy: 0.732397761493
Accuracy: 2278622
real 38m47,920s
user 3m56,693s
sys 1m42,399s
```

Conclusió: en aquestes dades hi ha un considerable augment de l'entropia. L'arbre generat disminueix en el seu grau d'exactitud (baixa fins al 0.73). Amb un augment de l'entropia, disminueix el grau d'exactitud.

4.6.9 Test 9

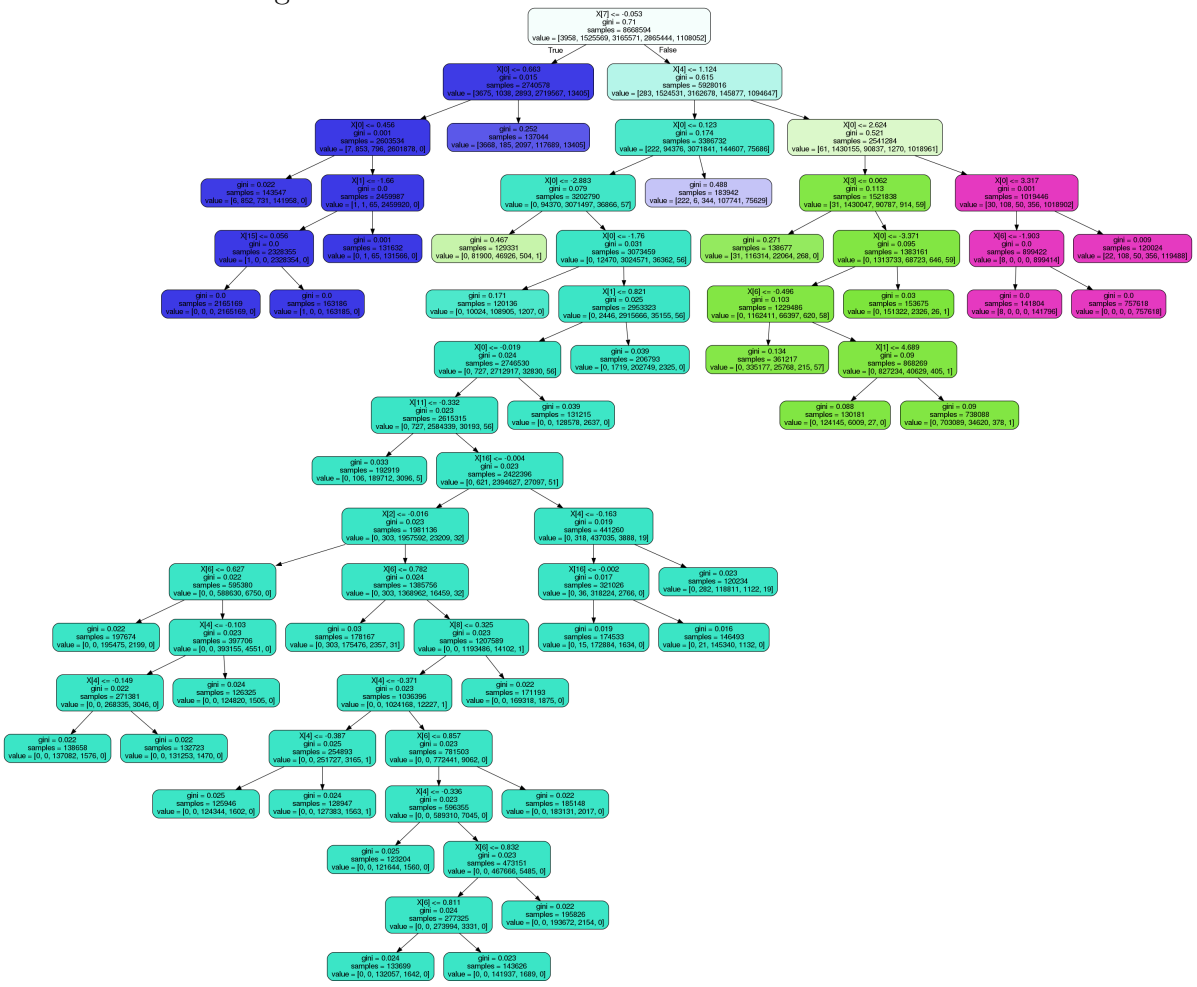
Aquest test és similar al test 4 i test 5 (parteix del mateix fitxer d'entrada) però aplicant-hi PCA. L'algorisme per tant, descompon les dades amb un nombre de components suficients com per explicar la variança en un 95% i realitza després l'entrenament de l'abre. Aquest és el resultat:

```
$ time python3 makeFourDecisionTree.py
Importing training test....
Explained variance es  [ 0.21165701  0.15262119  0.10872012  0.05616054
  0.04961431  0.04547995 0.04343958  0.03909086  0.03642278  0.03127678
  0.03096788  0.02491278 0.02185696  0.01859207  0.01682296  0.01566907
  0.01456773  0.01397399 0.01368517  0.01335444]
 El nombre de components és  20
Fitting the decision tree...
Number of leaves 225
Depth tree 31
Accuracy results: 0.980853879815
Accuracy results: 3643983
```

Es mostra a continuació part de l'arbre. Fixem-nos que ara els nodes de decisió parteixen de features (els components) i no de característiques originals:

```
|--- feature_7 <= -0.05
|   |--- feature_0 <= 1.08
|   |   |--- feature_16 <= -0.02
|   |   |   |--- class: NetBIOS
|   |   |--- feature_16 >  -0.02
|   |   |   |--- feature_2 <= 0.05
|   |   |   |   |--- feature_15 <= -0.02
|   |   |   |   |   |--- class: NetBIOS
|   |   |   |   |--- feature_15 >  -0.02
|   |   |   |   |   |--- class: NetBIOS
|   |   |   |--- feature_2 >  0.05
|   |   |   |   |--- class: NetBIOS
|   |--- feature_0 >  1.08
|   |   |--- class: Syn
|--- feature_7 >  -0.05
```

Figure 6: Arbre de decisió del novè test.



Conclusió del novè test: emprar PCA és una bona manera de reduir la dimensionalitat i, mantenir, en canvi, uns bons resultats. L'arbre però queda més il·legible, en el sentit que no empra les característiques originals sinó uns components calculats.

4.6.10 Test 10

Aquest darrer test és diferent que els anteriors i és que s'empra el concepte de Random Forests. Aquests consisteixen en un conjunt de decision trees similars als que s'han desenvolupat en els testos anteriors però més simples i

és que utilitzen menys atributs (seleccionats de manera aleatòria). Es mostra a continuació el constructor de l'algorisme:

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100,
criterion='gini', max_depth=None, min_samples_split=2,
 min_samples_leaf=1, min_weight_fraction_leaf=0.0,
 max_features='auto', max_leaf_nodes=None,
  min_impurity_decrease=0.0, min_impurity_split=None,
   bootstrap=True, oob_score=False, n_jobs=None,
   random_state=None, verbose=0, warm_start=False,
    class_weight=None, ccp_alpha=0.0, max_samples=None)
```

Escriure un algorisme de Random Forest[30] en Python és similar a com s'ha realitzat anteriorment, així que es comenta aquí solament les línies característiques:

```
#Cal importar la nova llibreria
from sklearn.ensemble import RandomForestClassifier
# 70% training and 30% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
 random_state=1)

rf = RandomForestClassifier(n_estimators=100, verbose=1)
rf.fit(X_train, y_train);
y_pred = rf.predict(X_test)
```

El resultat de l'execució emprant 100 decision trees es mostra a continuació:

```
$ time python3 randomforest.py
Importing training test...
Spliting data...
Fitting the random forest...
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed: 31.4min finished
Predicting
```

[30] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:  2.3min finished
Accuracy: 0.745723610146
Accuracy: 3093442


real 69m14,617s
user 30m22,877s
sys 2m36,560s
```

Es pot observar que el resultat és lleugerament millor al test 8 que emprava el mateix fitxer origen encara que també ha augmentat el temps de càlcul.

Emprar 1000 (en comptes de 100) arbres, no incrementa massa el resultat en aquest cas:

```
$ time python3 randomforest.py
Importing training test...
Spliting data...
Fitting the random forest...
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed: 284.4min finished
Predicting
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 1000 out of 1000 | elapsed: 113.2min finished
Accuracy: 0.745906820287
Accuracy: 3094202

real 434m27,470s
user 293m40,601s
sys 10m33,338s
```

Potser hagués estat adequat incrementar el nombre de "concurrent workers" emprant còmput en paral·lel amb l'objectiu de reduir el temps de càlcul (que ha estat de més de 7 hores).

Conclusió del desè test: Els Random Forests són una generalització dels arbres de decisió, basats en arbres més reduïts (pel que

fa a la selecció de les característiques) així que es tracta d'un bon mètode de càlcul.

5 Conclusions personals i treball futur

Després de la implementació de tots els tests és possible afirmar la idoneïtat de generar arbres de decisió per a la detecció d'atacs de DDoS, malgrat cal recordar que els arbres entrenats són dependents del dataset emprat i que, en el cas d'atacs nous, el model entrenat seria inútil.

En aquest punt desitjaria realitzar alguns comentaris respecte el mètode de treball utilitzat: em vaig preocupar massa inicialment d'implementar l'algorisme en concret i vaig deixar de banda altres assumptes que eren importants. Em refereixo sobretot a la descripció estadística i matemàtica del dataset d'entrada. Aquesta descripció hauria de ser una primera fase imprescindible abans d'afrontar un projecte de machine learning d'aquestes característiques. Una comprensió i depuració correcta de les dades permetria eliminar columnes innecessàries (pel fet simplement que no contenen dades o que presenten dependències lineals). Caldria haver calculat també de bones a primeres l'entropia de les dades i veure quins experiments podrien ser útils i quins no. Aquest punt caldria haver-se introduït en la planificació temporal del projecte (veure apartat 1.2).

Un altre assumpte a comentar és el cost computacional. L'execució d'alguns dels algorismes ha trigat hores, saturant complementament els recursos de l'ordinador, fent-lo inservible durant aquestes estones i, en alguns casos abortant per excés d'utilització de recursos. Vaig estar cercant per la xarxa quin seria el llindar de registres en el maquinari però no vaig arribar a cap conclusió. Si ho hagués sabut abans, potser hagués emprat noves infraestructures de càlcul.[31]

[31] Per exemple, Amazon EC2 (`https://aws.amazon.com/ec2/`. Extret de manera literal del lloc web: "Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers. Amazon EC2's simple web service interface allows you to obtain and configure capacity with minimal friction. It provides you with complete control of your computing resources and lets you run on Amazon's proven computing environment."

Amb l'ànim d'explicar a futurs estudiants del Màster Universitari de Seguretat de les TIC en aquesta línia de treball, comentar que:

- Cal realitzar abans de res una descripció estadística acurada del dataset d'entrada. Això permetria eliminar atributs inútils o variables dependents.
- Caldria elaborar una planificació dels testos a realitzar. En el meu cas, desconeixia completament tant els límits del meu maquinari com el temps de processament, així que la majoria de tasques s'han realitzat a les palpentes de manera empírica: potser hagués estat adequat elaborar una estimació pel que fa al nombre màxim de registres d'entrada. També, per a treballs que s'acostumen a realitzar en entorns domèstics, és aconsellable emprar infraestructures forànies (tipus Amazon EC2, encara que calgui pagar-les), marcant un pressupost previ.
- El dataset d'entrada ha de mantenir un balanceig adequat pel que fa als registres (veure conclusions tests 1, 3, 8).
- El nombre de registres del dataset d'entrada ha de disposar d'un nombre de registres suficients, malgrat és possible que el temps de càlcul sigui alt (que caldria acotar també) (veure conclusions dels tests 2 i 7).
- Per tal de prevenir l'overfitting dels arbres, és adequat forçar l'algorisme a disposar d'un mínim de registres (per exemple, un tant per cent del dataset (veure conclusions tests 4 i 5).
- La reducció per PCA és adequada, no cal comprendre per a usar-la tots els detalls matemàtics subjacents i ajuda a reduir la dimensionalitat. La desavantatge rau en el fet que l'arbre resultant no és tant llegible (veure tests 6 i 9).
- Els Random Forests són una generalització dels decision trees, que es poden paral·lelitzar amb treballadors concurrents (veure test 10).
- Una darrera recomanació, aquesta no tècnica sinó formal i és emprar LaTeX. Malgrat l'ús inicial és complicat, abstreu l'estudiant d'un munt de tasques tedioses i que el programari computa de manera automàtica.

Un treball futur consistiria en implementar totes les millores comentades i processar tot el dataset amb maquinari adequat.

List of Figures

References

[1] I. Sharafaldin, A. H. Lashkari, S. Hakak and A. A. Ghorbani *Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy*. 2019 International Carnahan Conference on Security Technology (ICCST), CHENNAI, India, 2019, pp. 1-8.

[2] Yavanoglu, Ozlem and Aydos, Murat *A Review on Cyber Security Datasets for Machine Learning Algorithms*. 2017 10.1109/BigData.2017.8258167

[3] Sebastian Garcia, Martin Grill, Honza Stiborek and Alejandro Zunino *An empirical comparison of botnet detection methods*. Computers and Security Journal, Elsevier. 2014. Vol 45, pp 100-123. http://dx.doi.org/10.1016/j.cose.2014.05.011

[4] Sani H.M., Lei C., Neagu D. (2018) *Computational Complexity Analysis of Decision Tree Algorithms*. In: Bramer M., Petridis M. (eds) Artificial Intelligence XXXV. SGAI 2018. Lecture Notes in Computer Science, vol 11311. Springer, Cham

[5] Pedregosa et al. *Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.*

[6] Peña, Daniel (2002)*Análisis de Datos Multivariantes*.

[7] Raschka S., Mirjalili, V. (2017)*Python Machine Learning*.