Anàlisi de intrusions a la xarxa en un entorn de dades massives amb Apache Spark

Eric Tormo Herrera

Resum- Actualment, a la nostra societat i en tot el món, es generen grans volums de dades cada dia. L'emmagatzemament i anàlisi d'aquestes dades permet extreure coneixement valuós per al desenvolupament i gestió de les diverses entitats. Big Data permet a aquestes entitats, classificar i detectar patrons, entre altres utilitats, per poder extreure el màxim coneixement de grans volums de dades. En aquest projecte s'utilitzarà el framework Apache Spark per a generar un model predictiu capaç de classificar les connexions a una xarxa de comunicacions, segons si es tracta d' una connexió normal o una intrusió a la xarxa.

Paraules clau- Dades massives, Mineria de dades, Apache Spark, intrusió, xarxa, arbre de decisió, atac, KDD Cup Data, temps real.

Abstract- Nowadays, in our society and the whole world, huge volumes of data are generated every day. The storage and analysis of this data allow extracting valuable knowledge for the development and management of the various entities. Big Data allows to these entities, classify and detect patterns, among other utilities, in order to extract the maximum knowledge of huge volumes of data. In this project is used the Apache Spark framework to generate a predictive model capable of classifying the connexions to a comunications network, depending on whether it is a normal connection or a network intrusion.

Keywords- Big Data, Data Mining, Apache Spark, intrusion, network, decision tree, attack, KDD Cup Data, real time.

1 INTRODUCCIÓ

Actualment el trànsit de dades que circula per les diverses xarxes té una gran importància per les diferents empreses o sectors de la nostra societat. Les dades, han esdevingut un dels recursos mes valuosos per al desenvolupament de projectes, tant tecnològics com financers. Son moltes les empreses i entitats que intenten extreure coneixement de les dades que acumulen diariament, però no és una tasca senzilla. El volum de dades que es gestiona diariament és de grans dimensions i els sistemes informàtics no incorporen les eines necessàries per defecte, per poder tractar de forma adequada i senzilla aquestes dades.

És aqui on s'introdueix el terme "Big Data" o dades massives, fent referència al conjunt de dades i procediments que degut al seu volum, varietat o velocitat, no poden ser processades amb els sistemes informatics habituals. Proporciona les eines necessaries a l'usuari, per al tractament i gestió de dades massives.

Big Data també es pot definir per les "4 V":

- **Volum:** quantitat de dades que es vol analitzar. Actualment, s'estima que el volum de dades està per sobre del zettabyte[1], pel que cal considerar la reestructuració d'alguns sistemes d'emmagatzematge en un entorn de dades massives.

- **Velocitat:** ritme al que es reben les dades i s'aplica alguna acció per al seu tractament. La informació té més valor si les dades són analitzades en temps real i perden valor quan aquestes deixen de reflectir la realitat.

  Existeixen dos tipus de velocitats que composen la

• E-mail de contacte: erictormo@gmail.com

• Menció realitzada: Tecnologies de la Informació

• Treball tutoritzat per: Jordi Casas Roma (Enginyeria de la Informació i de les Comunicacions)

• Curs 2017/18

[1] 1 Zettabyte = 10^{21} Bytes

velocitat final: la **velocitat de càrrega** (velocitat a la que les dades són preparades, interpretades i integrades amb la resta de dades) i la **velocitat de processament** (velocitat a la que s'apliquen funcions estadístiques o d'aprenentatge computacional).

- **Varietat:** no es tracta un únic format de dades, sinó que aquest és heterogeni, com textos, videos, etc. Segons el nivell d'estructuració de les dades es poden classificar en: **dades estructurades** (informació representada per un conjunt de dades simples amb una estructura fixe), **dades semiestructurades** (el conjunt de dades és simple pero no tenen una estructura fixe) i **dades no estructurades** (la informació no apareix representada per dades simples, sinó per una composició d'unitats estructurals de nivell superior).

- **Veracitat:** no totes les dades son fiables, pel que es necessiten accions o procediments pels quals validar la fiabilitat d'aquestes dades.

El procés de *big data* es basa en el següent: primer, es realitza la captura de les dades, extretes de fonts de dades o *datasets*. A continuació cal aplicar les transformacions necessàries a les dades per que totes es disposin en un mateix format. Un cop transformades les dades, s'emmagatzemaran en bases de dades NoSQL, ja que aquestes permeten manipular grans quantitas de informació de forma més ràpida i flexible. Quan ja es disposen de les dades en una base de dades NoSQL, s'apliquen tècniques de mineria de dades, per tal de realitzar l'anàlisi d'aquestes i poder extreure coneixement, com pot ser un model predictiu. Finalment, es presenten les dades a través de informes i gràfiques, entre d'altres.
Al llarg d'aquest treball, definirem quin conjunt de dades analitzarem i quines tecniques de *data mining* utilitzarem per al seu tractament.

2 OBJECTIUS I MOTIVACIÓ

Tal i com s'ha esmentat anteriorment, la tecnología Big Data està resultant ser una de les tecnologies emergents amb més rellevància en el sector TI, ja que gràcies als seus procediments, es pot obtenir més coneixement de la informació. És el principal motiu del desenvolupament d'aquest projecte, poder aprofunditzar en el mon del Big Data, conjuntament amb el *data mining*, per tal d'aprendre diverses tècniques amb les que poder extreure el màxim coneixement possible d'un gran volum de dades.
Els objectius d'aquest treball són els següents:

1. Netejar i filtrar les dades per tal de poder realitzar l'anàlisi de forma correcte.

2. Analitzar en primera instància les dades abans de ser tractades, per tal de compendre quins camps poden ser determinants en el resultat.

3. Realitzar un model predictiu que davant d'una connexió entrant, permeti detectar si es tracta d'una intrusió no permesa a la xarxa o d'una connexió normal.

4. Entrenar el model predictiu amb diferents algorismes d'aprenentatge computacional, per comparar resultats i poder determinar quin algorisme ens és més favorable.

3 ESTAT DE L'ART

Pel que fa a les tecnologies més utilitzades en l'àmbit del *big data* actualment, es destaquen dos per sobre de la resta. Es tracta de dos frameworks: *Apache Hadoop* i *Apache Spark*.

Apache Hadoop[2]: És un framework que permet el processament distribuït de grans conjunts de dades a través de diversos *clusters* mitjançant models de programació bàsics. Ha estat dissenyat per ser fàcilment escalable en servidors individuals, fins a milers de màquines, oferint cadascuna còmput i emmagatzematge local. Detecta i controla errors a la capa d'aplicació, oferint servei de disponibilitat en un cluster d'ordinadors. Aquest *framework* esta composat de diferents móduls, com el Hadoop Distributed File System (HDFS), Hadoop Yarn y Hadoop MapReduce.

Apache Spark[3]: És un *framework* de processament ràpid de dades en memòria, que permet executar de forma eficient tasques d'aprenentage computacional o càrregues de treball SQL en temps real, que requereixen un accés iteratiu al conjunt de dades. Spark requereix un administrador de *clusters*, podent suportar *Hadoop YARN* o *Apache Mesos* i un sistema d'emmagatzamament distribuît, com per exemple *HDFS, MapR File System o Amazon S3*, entre d'altres.

3.1 Introducció a Apache Spark

Tal i com hem mencionat anteriorment, Spark és un *framework* d'Apache, que permet realitzar anàlisi de dades en temps real i proporciona eines per dur a terme tasques de mineria de dades.
El motor de Spark, utilitza els RDD com a tipus de dada més bàsic. El RDD agrega dades i particions a través d'un clúster de servidor, on poden realitzar calculs, traslladar les dades a un magatzem diferent o executar-se mitjançant un model analític. Amaga la complexitat computacional a l'usuari, ja que aquest no ha de preocupar-se en definir on s'envien cadascun dels fitxers específics, ni quins recursos ha d'utilitzar per distribuir o recuperar els fitxers.
El RDD són tipus de dades inmutables, és a dir, no es poden realitzar operacions de transformació sobre ells. En el nostre projecte, si es volen realitzar operacions d'aquest tipus, caldrà aplicar les transformacions necessàries a les dades abans de que aquestes siguin declarades com a RDD.

Spark està format per diferents mòduls i llibreries que permeten analitzar i gestionar les dades de forma senzilla. Algunes d'aquestes llibreries que s'utilitzaran en el nostre

[2] http://hadoop.apache.org/
[3] https://databricks.com/spark/about

projecte són les següents:

- **Spark SQL**: Permet als usuaris poder consultar dades emmagatzemades en diferents aplicacions utilitzant el llenguatge SQL comú.

- **MLlib**: Es tracta d'una llibreria que presenta eines per realitzar tasques d'aprenentatge computacional i permet als usuaris utilitzar operacions estadístiques avançades. Aquesta llibreria serà una de les més utilitzades per poder aplicar els algorismes de mineria de dades.

A nivell de llenguatge de programació per desenvolupar les tasques necessàries, Apache Spark suporta diferents llenguatges com Java, Scala i Python. En el cas del nostre projecte, utilitzarem el llenguatge Python, a través del mòdul *Pyspark*[4].

3.2 Diferències entre Hadoop i Spark

Tant Hadoop com Spark són frameworks d'Apache per tractar grans volums de dades. A continuació es presenten les principals diferencies entre els dos frameworks.

Avantatges de Spark respecte Hadoop

- **Velocitat**: Spark executa les aplicacions més ràpid que Hadoop, fins a 100 vegades a memòria i 10 vegades a disc.

- **Dificultat en programació**: Spark presenta operadors d'alt nivell amb els que és fàcil programar. Hadoop en canvi, requereix que els desenvolupadors codifiquin totes les operacions a realitzar.

- **Facilitat d'ús**: Amb la instal·lació de Spark no es necessita afegir cap component o mòdul més, mentre que amb Hadoop es requereix la utilització de diferents motors per realitzar segons quines funcionalitats, com Storm, Giraph, Impala, entre d'altres.

- **Anàlisi en temps real**: Spark permet analitzar dades generades per events de *streaming* en temps real, com poden ser els missatges de Twitter i Facebook, mentre que Hadoop analitza grans quantitats de dades, però un cop aquestes estan emmagatzemades.

Inconvenients de Spark respecte Hadoop

El principal inconvenient que presenta Spark envers Hadoop és la seguretat. Spark només suporta l'autenticació per contrasenya secreta compartida (*shared secret password authentication*), mentre que Hadoop és més segur, suportant Llistes de Control d'Accés (ACLs[5]).

[4]Documentació de Pyspark: http://spark.apache.org/docs/latest/api/python/

[5]ACLs: https://en.wikipedia.org/wiki/Accesscontrollist

4 METODOLOGIA

Com a metodologia de treball per dur a terme aquest projecte, s'ha utilitzat la metodologia en *Cascada*[6]. Per al desenvolupament del projecte, és necessari realitzar un conjunt d'etapes o fases, les quals han de ser realitzades de forma secuencial i en un cert ordre. Fins que no es disposa de la realització completa d'una d'elles, no es pot continuar a la realització de la següent, per falta o incompletesa de les dades.
A la figura 1, es poden observar les fases d'implementació que tindrà el projecte, explicades a la secció 5.

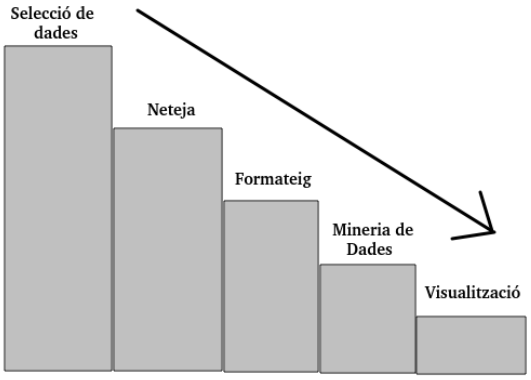

Fig. 1: Metodologia en Cascada

5 FASES DEL PROJECTE

Per al desenvolupament del treball, s'utilitzarà el framework *d'Apache Spark*. Aquest framework serà instal·lat a una màquina virtual amb Sistema Operatiu Linux, on es descarregarà i tractarà el fitxer del *KDD Cup 1999 Data*[7].

Per tal d'aplicar les tècniques necessàries per elaborar un anàlisi de les dades, es seguirà l'estructura que es descriu a continuació:

1. **Selecció de dades**: Consisteix en realitzar una recerca en diverses fonts de dades obertes (*Open Data*), per tal de descarregar les dades que es volen analitzar.

2. **Neteja**: Es realitza l'eliminació de inconsistències i dades no fiables, per tal de preparar un conjunt de dades el més fiable i verificades possible.

3. **Formatejat de dades**: Es trien les dades que es volen utilitzar del conjunt anterior i s'apliquen transformacions, per tal de donar un format específic a les dades que facilitin les tècniques de mineria de dades.

[6]Cascada: https://www.smartsheet.com/agile-vs-scrum-vs-waterfall-vs-kanban

[7]KDD Cup 199 Data: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

4. **Mineria de dades**: S'apliquen tècniques i algorismes d'aprenentage computacional per trobar patrons que ajudin a elaborar el model predictiu.

5. **Visualització**: A través de mètodes de representació, com gràfiques, es presenten els resultats obtinguts d'extreure el coneixement de les dades.

Al llarg del treball, es desenvoluparan dos models predictius, basats ens dos algorismes d'aprenentatge computacional diferents. Un cop s'hagin creat aquests models, es compararan els resultats d'encert de tots dos, per veure quin proporciona millors resultats per al nostre conjunt de dades.

6 IMPLEMENTACIÓ

6.1 FASE 1. DADES A ANALITZAR

Els arxius que s'han utilitzat per dur a terme el projecte, són els proporcionats per *KDD Cup 1999 Data*, *kddcup.data.gz (742.6 MB)* com a conjunt d'entrenament del model predictiu i *corrected.gz (47,3 MB)* com a conjunt de validació. Les dades que s'analitzaran en aquest projecte per tal de poder determinar si un accés a la xarxa es tracta d'una connexió normal o és un atac, son les diferents components que composen una connexió en el conjunt d'entrenament. Aquests camps es recullen i descriuen a les Taules 1,2 i 3.

6.2 FASE 2. NETEJA

La web del *KDD Cup 1999 Data* també proporciona un arixu que conté el 10% del conjunt d'entrenament que hem mencionat a l'apartat anterior. Analitzant aquest arxiu, podem observar que cada fila conté com a ultim camp el resultat de la connexió, finalitzat amb un ".". Aquest caràcter s'elimina per evitar problemes en mètodes de filtratge o classificació de l'algorisme.

A través d'un script propi, utilitzant la comanda *distinct()* de *pyspark*, obtenim tots els valors diferents per cadascun dels camps que composen la connexió i podem eliminar aquelles files de l'arxiu, que continguin valors incongruents.

6.3 FASE 3. FORMATEIG DE LES DADES

Un cop han estat seleccionades y netejades les dades, es procedeix a dur a terme el formateig d'aquestes. Tal i com hem pogut observar a la secció 6.1, la gran majoria de les dades s'expressen de forma numèrica, excepte tres camps: protocol, servei i flags.

Per poder realitzar operacions amb totes les dades conjuntament, cal que totes comparteixin el mateix tipus de dades, pel que transformarem aquests camps a dades numèriques. Per realitzar aquesta categorització, s'ha creat una funció que indexa cadascun dels valors de text, proporcionant un valor numèric, com si es tractés d'un identificador.

6.4 FASE 4. MINERIA DE DADES

L'etapa de mineria de dades consisteix en aplicar algoritmes que ajudin a detectar patrons, per tal de poder elaborar el model predictiu final. Els algorismes de mineria de dades permeten analitzar un volum de dades, amb l'objectiu de trobar patrons o relacions entre les diferents dades del conjunt. En aquest projecte utilitzarem dos tècniques basades en algoritmes de classificació: els *Arbres de Decisió* i *Random Forest*.

6.4.1 Arbres de Decisió

Construeixen models de classificació o regressió en estructura d'arbre. Aquest model està constituit per nodes, vectors i fletxes. Cada node representa el moment en que s'ha de triar una acció d'entre diferents possibles.
Els vectors son la representació de la decisió final, que comprén desde la primera decisió presa al primer node fins arribar a l'últim node. Les fletxes són la unió entre diferents nodes i representen cadascuna de les accions/decisions que es poden triar.
Els arbres de decisió es construeixen desde el node arrel (primer node i millor predictor de l'arbre), fins a les fulles (últims nodes de l'arbre). Per a la construcció de l'arbre, s'utilitza l'algorisme *ID3* [8], que requereix el càlcul de l'Entropia i el Guany de informació.
El càlcul de l'Entropia es realitza en cadascun dels nodes de l'arbre. Es basa en calcular, com d'homogeni és el conjunt de dades que s'està evaluant. Si el conjunt de dades és totalment homogeni, l'entropia serà 0, en canvi, si aquest conjunt conté diverses dades, dividides a parts iguals entre elles, és a dir, les dades es repeteixen en la mostra el mateix nombre de vegades cadascuna, l'entropia serà 1.
Per obtenir el guany de informació es calcula les diferències de les entropies per a cadascun dels atributs del conjunt de dades, aquell atribut que disposi de més guany serà el primer node classificador, i així succesivament fins arribar als nodes fulla.
Un cop s'ha construit l'arbre amb una part del conjunt de dades d'entrenament, es calcularà l'encert promig utilitzant l'altre conjunt de dades restant, les dades de validació. A la figura 2, és pot observar un exemple d'un arbre de decisió.

Punts forts de l'algorisme

- **Fàcil de comprendre**: Els arbres de decisió presenten un model gràfic fàcil d'entendre un cop han estat explicats els seus elements i el seu funcionament.

- **Poc requeriment de formatejat de dades**: Tot i que en el nostre projecte hem formatejat les dades per evitar posibles errors o incongruencies, els arbres de decisió no necesiten la creació de noves variables o variables fictíces per poder treballar.

- **Bon funcionament davant d'un gran volum**: Aquest algorisme permet analitzar un conjunt de dades d'un

[8] Algorisme ID3: https://www.cise.ufl.edu/ ddd/cap6635/Fall-97/Short-papers/2.htm

Component	Descripció
Duration	Temps en el que ha estat establerta la connexió
Protocol	Protocol de la capa de transport que s'ha utilitzat (tcp o udp)
Service	Servei utilizat a la capa d'interconnexió de xarxes (http,ftp, ssh...)
Flag	Part d'un segment TCP, que identifica un tipus de connexió (SYN, ACK...)
Source Bytes	Quantitat de dade en bytes enviades de l'emisor al receptor
Destination Bytes	Quantitat de dade en bytes enviades del receptor a l'emisor
Land	Identifica si la connexió s'ha realitzat desde o cap al mateix host/port
Wrong fragment	Nombre de fragments identificats com a "wrong"
Urgent	Nombre de paquets marcats com a ürgent"

TAULA 1: CARACTERÍSTIQUES BÀSIQUES D'UNA CONNEXIÓ TCP INDIVIDUAL.

Component	Descripció
Hot	Nombre de identificadors "hot"
Number Failed Logins	Numero de intents de login erronis
Logged in	Descriu si el login s'ha realitzar amb èxit o no
Number Compromised	Nombre de condicions çompromised"
Root Shell	Descriu si s'ha pogut obtenir una "root shell"amb exit
Su Attempted	Identifica si s'ha realitzat la comanda "su root"
Number Root	Nombre d'accesos com a root
Number File Creations	Nombre d'operacions de creació d'arxius realitzades
Number Shells	Nombre de shells prompts
Number Acces Files	Numero d'operacions a arxius de control d'accés
Number Outbound Files	Nombre de comandes d'anada en una sessió ftp
Is Host Login	Identifica si el loggin prové d'una "hot list"
Is Guest Login	Identifica si el login prové d'una sessió de convidat

TAULA 2: FUNCIONS DE CONTINGUT D'UNA CONNEXIÓ SUGGERIDA PEL CONEIXEMENT DEL DOMINI

Component	Descripció
Count	Nombre de connexions amb el mateix host que la connexió actual en els ultims dos segons
Serror Rate	Percentatge de connexions al mateix host que tenen "SYN"errors
Rerror Rate	Percentatge de connexions al mateix host que tenen "REJ"errors
Same SRV Rate	Percentatge de connexions al mateix host i al mateix servei
Diff SRV Rate	Percentatge de connexions al mateix host i diferents serveis
SRV Count	Nombre de connexions al mateix servei que la connexió actual en els últims dos segons
SRV Serror Rate	Percentatge de connexions al mateix servei que tenen "SYN errors"
SRV Rerror Rate	Percentatge de connexions al mateix servei que tenen "REJ errors"
SRV Diff Host Rate	Percentatge de connexions a diferents hosts pero mateix servei"

TAULA 3: FUNCIONS DE TRÀNSIT CALCULADES UTILITZANT UNA FINESTRA DE TEMPS DE DOS SEGONS

gran volum en un temps raonable. Al llarg del projecte es podrà observar i comparar el temps de càlcul requerit en cada cas.

Limitacions

- **Dependència del conjunt d'entrenament**: Depenent del percentatge de dades que s'utilitzin com a conjunt d'entrenament i quina distribució de variables hi hagi en aquest conjunt, l'arbre generat, generalitzarà més unes variables que d'altres, podent ser aquestes millor classificadores per a l'arbre.
- **Conceptes difícils d'aprendre**: Hi han conceptes que per a l'algorisme són difícils d'aprendre, com problemes de multiplexor. Aquest fet provoca que l'arbre es construeixi amb molta profunditat. En el nostre projecte però, aquestes profunditats d'arbre estaran controlades.

Resultats

En aquest apartat realitzarem diverses proves, modificant els paràmetres del codi que genera un arbre de decisió amb

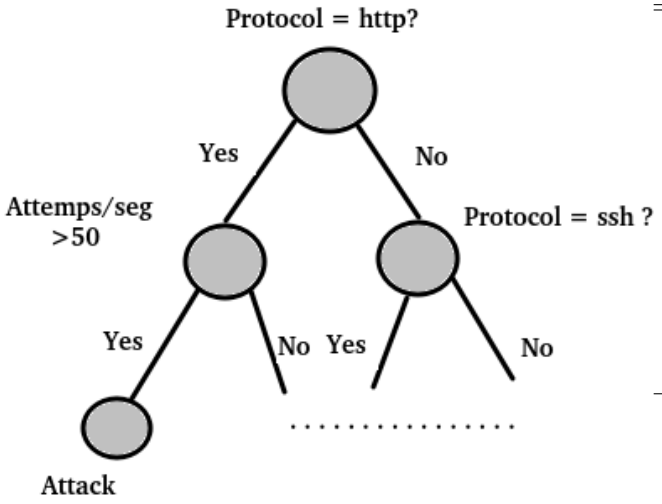

Fig. 2: Exemple d'arbre de decisió (no referent al resultat del projecte)

el modul *pyspark* d'Apache Spark. Per tal de generar un arbre de decisió els paràmetres requerits són els següents:

- **numClasses**: número de classes en que es vol fer la classificació. Per defecte totes les proves seran realitzades amb un valor de 2 per aquest paràmetre, diferenciant entre si la connexió és un atac o no.
- **input**: conjunt de dades per generar l'arbre. Camp inmutable.
- **categoricalFeaturesInfo**: Especifica quins camps de dades son categòrics, és a dir, quins camps de dades són tractats diferents a la resta, i quins valors comprén cadascun dels camps especificats. Camp inmutable.
- **inpurity**: criteri utilitzat per calcular la impuresa dels nodes. Els valors per aquest paràmetre poden ser *gini* o *entropy* [9]
- **maxDepth**: màxima profunditat de l'arbre, és a dir, maxima quantitat de decisions que caldrà prendre per arribar a la solució. La màxima profunditat amb la que es pot generar un arbre és de 30.
- **maxBins**: nombre màxim de diferents decisions (fletxes) que es poden prendre en cada node.

A continuació es presenten diferents resultats obtinguts de realitzar arbres de decisió, modificant els valors dels camps *inpurity* i *maxDepth*.El camp *Temps*, fa referència al temps que ha trigat l'algorisme en entrenar l'arbre.

Tot i ser una breu diferència, els càlculs de la impuresa realitzats amb el valor *entropy*, mostren millors resultats tant com en temps d'execució per a la generació de l'arbre com a l'encert del model predictiu.

[9] Càlcul de impuresa dels nodes: https://spark.apache.org/docs/2.2.0/mllib-decision-tree.html

Inpurity	MaxDepth	Temps (seg)	Encert (%)
gini	4	175,792	91,83
	5	178,769	92,17
	7	188,643	92,28
	10	214,219	92,30
	12	231,786	92,69
	15	251,499	92,71
	17	266,273	92,71
	20	280,003	92,95
	22	311,111	93,10
	25	312,399	93,16
	27	325,661	93,16
	30	327,669	93,16
entropy	4	167,577	92,24
	5	172,270	92,26
	7	187,438	92,38
	10	208,830	92,69
	12	225,362	93,53
	15	238,930	93,80
	17	257,069	93,73
	20	263,227	93,73
	22	265,334	93,73
	25	266,182	93,73
	27	266,617	93,73
	30	267,359	93,73

TAULA 4: RESULTATS D'ENCERT DELS ARBRES DE DECISIÓ SEGONS LA INPURESA I LA MÀXIMA PROFUNDITAT.

Visualització de resultats

A les figures 3 i 4, es pot observar la diferencia que existeix entre aplicar el paràmetre *gini* i *entropy*, tant en temps d'execució de l'algorisme, com en encert a la predicció.

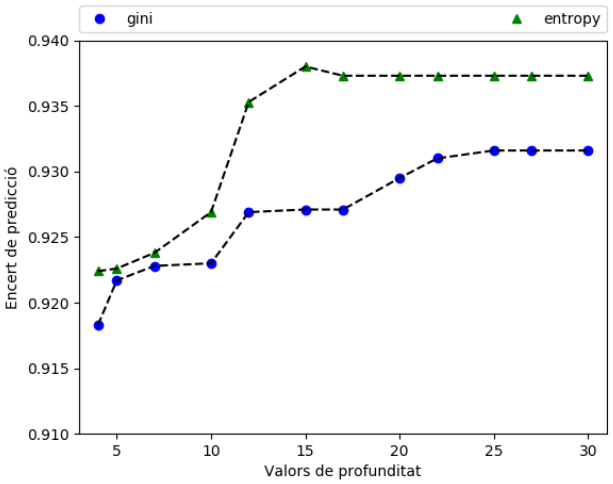

Fig. 3: Comparació entre els resultats de predicció dels arbres generats

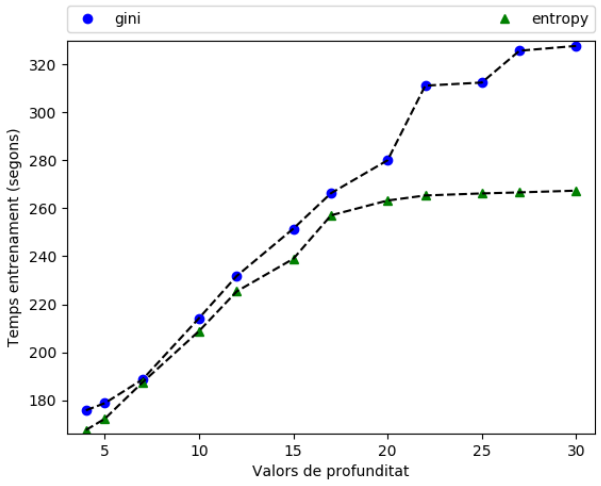

Fig. 4: Comparació entre els temps de execució per generar els arbres

Tal i com es pot observar a la Taula 4, els valors generats amb el paràmetre *entropy* són millors. Mentre que els arbres amb paràmetre de impuresa *gini* necessiten una profunditat d'arbre de 25 nodes per assolir el màxim encert de predicció, els arbres amb el paràmetre de impuresa *entropy* només necessiten una profunditat màxima d'arbre de 15 nodes per assolir l'encert de predicció màxim, fent que tant el temps de construcció de l'abre de decisió, com el temps de predicció sigui menor. Per tant, es pot concloure que aquest arbre és l'òptim de tots dos quan es volen realitzar prediccions mitjançant arbres de decisió per a aquest conjunt de dades.

6.4.2 Random Forest

Consisteix en la construcció de multiples arbres de decisió, generats cadascun amb diferents particions del conjunt d'entrenament, per tant, cadascun dels arbres de decisió que es generen, són diferents l'un de l'altre i promitgen un valor de predicció diferent. Es recullen tots els valors de predicció de cadascun dels arbres generats i es calcula el promig de tots ells, proporcionant un valor d'encert general.
Aquest algorisme de *bagging* [10], permet reduir les variances que es produeixen en l'encert d'un arbre de decisió, que es creat amb diferent conjunt d'entrenament que altre. Mentre que en algorismes *CART*[11], com els arbres de decisió, es permet fer recerca de totes les variables, en tot el conjunt d'entrenament, per trobar quina és la millor variable classificadora. Amb Random Forest es canvia aquest procediment, limitant a un conjunt aleatori d'atributs en els que fer la cerca de millor atribut classificador, ja que s'utilitzen diferents particions del conjunt de dades.
Com que per a la construcció de cadascun dels arbres es selecciona un subconjunt de variables aleatòries, s'afavoreix el poder donar opcions de selecció com a atribut classificador a aquells atributs que quedarian exclosos o eclipsats per altres variables que tinguesin més rellevància. Amb aquest procediment doncs, es permet poder mesurar la importància relativa de cada variable del conjunt d'entrenament, estimant l'error comés per cadascun dels classificadors generats per l'algorisme.

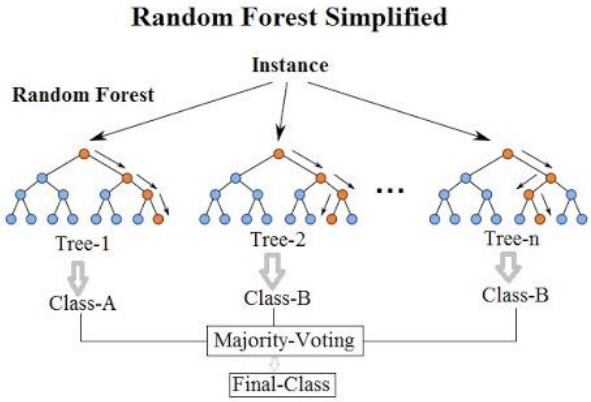

Fig. 5: Exemple de Random Forest

Visualització de resultats

En aquest apartat, es presenten els resultats dels encerts que s'han obtingut en realitzar l'algorisme de *Random Forest*. Com aquest algoritme es basa en la generació de diferents arbres de decisió, els paràmetres que utilitza són els mateixos que els d'un algoritme que genera un arbre de decisió, afegint dos paràmetres més, *numTrees* i *featureSubsetStrategy*:

- **numTrees**: Indica el nombre d'arbres que es volen generar amb l'algoritme.

- **featureSubsetStrategy**: Nombre de característiques per utilitzar com a candidates de decisió als nodes. En el nostre cas, el valor es generarà de forma automàtica.

Tal i com s'ha procedit amb els arbres de decisió a l'apartat anterior, es generaran resultats mantenint el valor del paràmetre *maxDepth* que millors resultats ha aportat, en aquest cas 30 i es modificaran el paràmetres *inpurity* i *numTrees*.

Un cop s'han recullit els valors de les prediccions realitzades amb aquest algorisme utilitzant els diferents paràmetres per a la impuresa, podem generar les figures 6 i 7 per visualitzar les diferències que es presenten.
Analitzant els resultats de la Taula 5 i les figures 6 i 7, es pot observar que la diferència en valors tant de predicció com de temps d'execució, no són tant distants com en el cas de només un arbre de decisió.
Pel que fa referència als valors d'encert a la predicció, en comparació entre els paràmetres *gini* i *entropy*, en utilitzar l'algorisme *Random Forest*, és el paràmetre *gini*, el que presenta millor encert. Observant la gràfica d'encerts en la predicció, tots dos parametres gairebé segueixen la mateixa tendència d'encert. El numéro d'arbres a generar per obtenir un resultat d'encert òptim i fiable és de 20.

[10] Bagging: Procediment per reduir les variances en algorismes que presenten una alta variança
[11] Algorismes CART: https://www.ibm.com/support/knowledgecenter/en/SSLVMB22.0.0/com.ibm.spss.statistics.algorithms/algtree-cart.htm

Inpurity	NumTrees	Temp (seg)	Encert (%)
entropy	2	349,677	92,57
	5	457,035	92,52
	10	510,104	92,54
	12	610,480	92,68
	15	710,512	92,26
	17	714,519	92,51
	20	893,641	92,66
	25	915,317	92,42
	50	1672.010	92,32
	100	3526,714	92,44
gini	2	337,893	92,93
	5	483,725	92,74
	10	659.949	92,64
	12	694,655	92,54
	15	844,002	92,49
	17	844,592	92,52
	20	919.803	92,79
	25	1142,669	92,51
	50	2167.880	92,47
	100	5073,169	92,59

TAULA 5: RESULTATS D'ENCERT MIG DELS ARBRES DE DECISIÓ SEGONS LA INPURESA, LA MÀXIMA PROFUNDITAT I EL NÚMERO D'ARBRES GENERAT

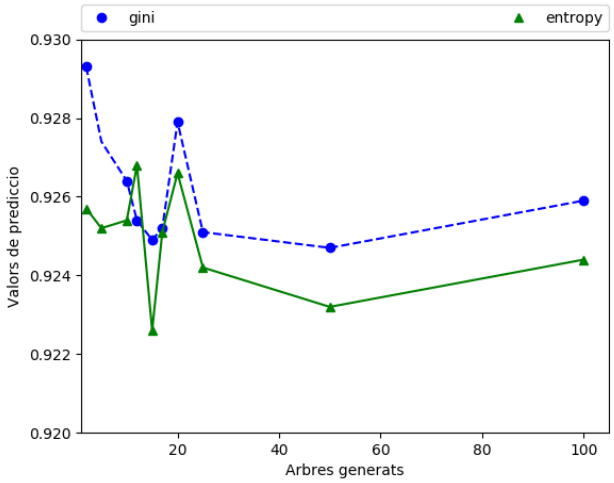

Fig. 6: Comparació de valors de predicció amb Random Forest

És cert que el millor resultat es presenta quan els arbres generats són 2, però es possible que aquests dos arbres s'hagin entrenat amb un conjunt de dades molt semblant, i per tant tinguin un encert molt similar amb poca variança, pel que no és pot considerar del tot fiable, ja que el conjunt de validació també es sempre el mateix.

6.4.3 Resultats de Random Forest amb diverses profunditats

Després d'analitzar els resultats dels dos algorismes, hem pogut concloure quins són els paràmetres que proporcionen resultats òptims en cadascun d'ells. Pel que fa als Arbres de Decisió, el càlcul mitjançant la impuresa *entropy* i

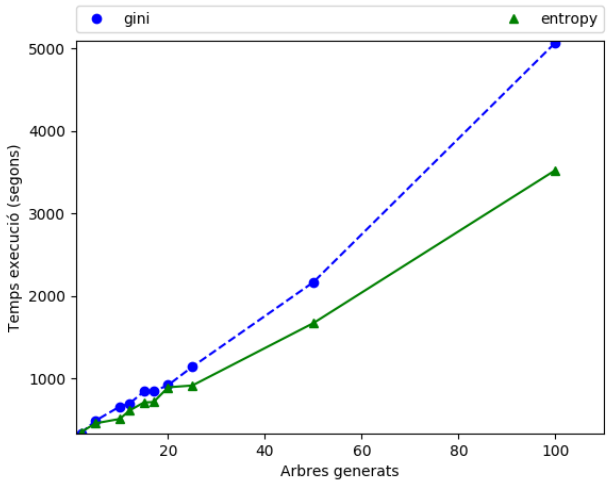

Fig. 7: Comparació entre els temps de execució per generar tots els arbres

una profunditat de 30 nodes, proporcionaba els millors resultats mentre que a Random Forest, els proporcionaba la impuresa *gini* amb la mateixa profunditat i un nombre d'arbres generats igual a 20.
En aquest apartat, analitzarem si és possible obtenir millors resultats variant les profunditats dels arbres, amb l'algorisme *Random Forest* i mantenint el nombre d'arbres a generar que millors resultats ens ha proporcionat a l'apartat anterior.

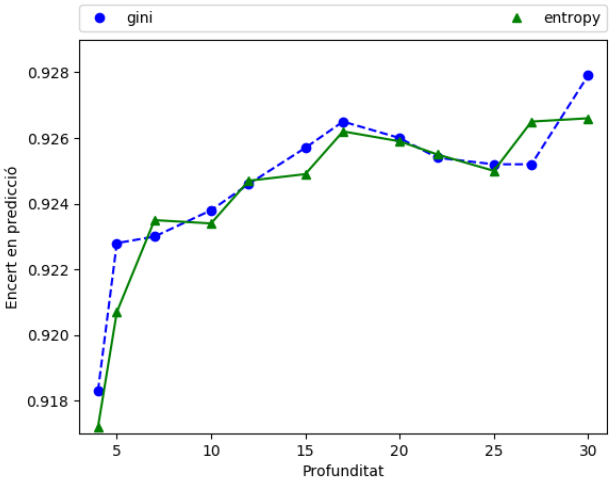

Fig. 8: Encert amb Random Forest segons la profunditat, generant 20 arbres de decisió

Si comparem els resultats d'encert d'aquest algorisme amb aquests paràmetres, amb el millor resultat d'encert del *Random Forest* realitzat a la secció 6.4.2, obtenim que el millor resultat és el mateix, utilitzant el paràmetre *gini* amb una profunditat d'arbre de 30 nodes. Tot i que no aconsegueix arribar a l'encert màxim d'un sol arbre de decisió amb profunditat màxima de 30 nodes, l'encert que presenta en la predicció s'hi apropa força, pel que es pot dir que aquest algorisme amb aquests paràmetres, és una solució consistent.

Inpurity	MaxDepth	Temps (seg)	Encert(%)
entropy	4	218,199	91,72
	5	244,622	92,07
	7	295,050	92,35
	10	367,305	92,34
	12	433,730	92,47
	15	503,370	92,49
	17	612,000	92,62
	20	702,152	92,59
	22	754,061	92,55
	25	778,580	92,50
	27	761,798	92,65
	30	893,641	92,66
gini	4	211,468	91,83
	5	263,231	92,28
	7	326,839	92,30
	10	368,531	92,38
	12	424,373	92,46
	15	519,132	92,57
	17	601,156	92,65
	20	681,729	92,60
	22	755,515	92,54
	25	866,631	92,52
	27	916,129	92,52
	30	919.803	92,79

TAULA 6: RESULTATS D'ENCERT MIG DELS ARBRES DE DECISIÓ SEGONS LA INPURESA I LA MÀXIMA PROFUNDITAT CREANT 20 ARBRES

7 PROBLEMES EN EL DESENVOLUPAMENT

Al llarg del projecte, han sorgit problemes que han dificultat el desenvolupament d'aquest. Els problemes principals han estat la memòria de la màquina amb la que es treballa i el seu sistema operatiu. Degut a les facilitats tant de instal·lació d'Apache Spark i mòduls o llibreries requerides, com d'utilització d'eines a posteriori, per tal de generar gràfiques o càlculs, es va decidir utilitzar el sistema operatiu *Ubuntu 16.04*. Degut a que es disposava d'un sistema operatiu *Windows 10*, el projecte es va iniciar amb Màquines Virtuals (*VirtualBox*) amb el sistema operatiu desitjat. Després de realitzar diferents configuracions de memòria de la màquina virtual, no es va aconseguir finalitzar amb èxit les proves d'execució del script que executa Spark amb els algorismes necessaris per realitzar els càlculs, degut a fallades de memòria.
Davant d'això es va optar per instal·lar el sistema operatiu en substitució de l'actual (*Windows 10*), dotant de suficient memòria per evitar aquests problemes.
Finalment, l'últim problema que es va trobar va ser en alguns fitxers de configuració de Spark, per tal de poder executar l'script. Els fitxers de configuració del directori *"/usr/lib/spark/bin/"* (aquest directori pot variar segons la instal·lació de cada usuari), requerien l'actualització de la ubicació d'alguns mòduls d'Spark, segons el directori on havien estat instal·lats.

8 LINIES DE FUTUR

En cas de continuar amb el desenvolupament d'aquest projecte, s'estudiaria la realització d'un model predictiu més específic. L'arxiu de dades que proporciona el *KDD CUP Data 1999* especifica en cadascuna de les connexions, si es tracta d'una connexió normal o bé si es tracta d'un atac DoS, Buffer Overflow, entre d'altres. En el nostre projecte, hem substituit aquests valors d'atac, per un valor genèric "atac", per realitzar una classificació binaria. Per tant, l'objectiu seria desenvolupar un classificador multiclasse, capaç de predir si la connexió és normal, o un atac específic, entre tots els que presenta l'arxiu de dades.
Altre implementació d'aquest projecte seria analitzar les connexions que es realitzen a una xarxa concreta en temps real, ja que, tal i com hem vist, el framework Apache Spark presenta eines que faciliten l'anàlisi de dades en temps real.

9 CONCLUSIONS

Avui dia, *big data*, presenta un procés per a l'anàlisi de dades de gran volum, que resulta de gran profit per a tots els sectors que tracten amb dades d'aquestes característiques. En aquest projecte, hem posat en pràctica aquest procés, conjuntament amb l'utilització de la mineria de dades, per analitzar un fitxer de gran volum, en el que es recullen diferents connexions realitzades, categoritzades com a una connexió normal, o un atac.
Per poder desenvolupar l'anàlisi d'aquestes dades, s'ha utilitzat el framework *Apache Spark*, que presenta diferents llibreries per poder utilitzar algorismes de mineria de dades. Tot i que aquest framework és característic per poder realitzar anàlisi de dades en temps real, el projecte s'ha realitzat llegint les dades ja emmagatzemades. Els algorismes que s'han utilitzat han estat els *Arbres de Decisió* i *Random Forest*. Després de realitzar diferents provés modificant els paràmetres de cada algorisme, s'ha pogut crear un model predictiu, basant en l'algorisme *Random Forest*, amb un encert de predicció molt elevat.

10 AGRAÏMENTS

Agrair especialment al tutor Jordi Casas Roma, per l'ajuda aportada alhora de fer recerca d'arxius de volums suficientment grans en diversos repositoris *Open Data*, per tal de poder començar el projecte, per l'orientació sobre quins algorismes d'aprenentatge computacional podrien donar bons resultats i a la vegada ser fàcils de comprendre i per la pàciencia davant dels diversos problemes que s'han produït al llarg del desenvolupament del treball.

REFERÈNCIES

[1] ¿Qué es Big Data? (2012, June 18). https://www.ibm.com/developerworks/ssa/local/im/que-es-big-data. Accedit el 21/09/2017

[2] Oracle Big Data. https://www.oracle.com/es/big-data/index.html. Accedit el 22/09/2017

[3] Bagging and Random Forest Ensemble Algorithms for Machine Learning. (2016, September 21). https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/. Accedit el 28/12/2017

[4] Viktor Mayer-Schönberger (2014) Big Data: A Revolution That Will Transform How We Live, Work, and Think.

[5] Welcome to Apache™ Hadoop®!. http://hadoop.apache.org/. Accedit el 1/10/2017

[6] Apache Spark™ - Lightning-Fast Cluster Computing. https://spark.apache.org. Accedit el 1/10/2017

[7] Decision Tree. http://www.saedsayad.com/decisiontree.htm. Accedit el 15/12/2017

[8] KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accedit el 7/09/2017

[9] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. (1996). From Data Mining to Knowledge Discovery in Databases. American Association for Artificial Intelligence

[10] David J. Hand. (1998). Data Mining: Statistics and More?. The American Statistician

[11] What is the difference between Hadoop and Spark?. https://www.quora.com/What-is-the-difference-between-Hadoop-and-Spark. Accedit el 3/1/2018