Università
Ca'Foscari
Venezia

Corso di Laurea

in informatica

ordinamento Data science

Tesi di Laurea

Studio ed analisi di alcuni algoritmi nell'ambito dell'Intrusion Detection System

Relatore
Ch. Prof. Riccardo Focardi

Laureando
Marco Pastrello
Matricola
881679

Anno Accademico
2021 / 2022

Università
Ca'Foscari
Venezia

Indice

Introduzione

Obiettivi

L'obiettivo di questo lavoro di Tesi è studiare l'utilizzo di algoritmi di machine learning solitamente applicati per la realizzazione di Intrusion Detection System all'interno il contesto fornito dal Dataset UNSW-NB15.

Università
Ca'Foscari
Venezia

L'Intrusion Detection System (IDS) è un dispositivo software o hardware o entrambi utilizzato per monitorare una rete e identificare attività malevole o violazioni di policy. Queste attività vengono riportate ai Security Information and Event Management (SIEM) che ricevendo numerosi output utilizza un filtro per decidere come gestirle.

IDS possono essere classificate in due diverse categorie in base a dove avviene il rilevamento e in altre due in base al metodo di rilevamento.

Network o Host Intrusion Detection System

Network Intrusion Detection System

I Network Intrusion Detection System (NIDS) sono posizionati in maniera tale da poter monitorare l'intera sottorete, in questo modo è possibile il controllo del traffico e l'identificazione di possibili attacchi attraverso algoritmi, intelligenza artificiale o similarità con attacchi già registrati.

Host Intrusion Detection System

I Host Intrusion Detection System (HIDS) sono posizionati sui singoli host della rete e agiscono solo sui pacchetti inviati o ricevuti da quest'ultimi, comunicando all'utente possibili attività sospette, per farlo solitamente controlla i file all'interno del dispositivo e controlla se le zone critiche sono state modificate o cancellate.

Signature-based o Anomaly-based Intrusion Detection System

Signature-based Intrusion Detection System

I Signature-based Intrusion Detection System si caratterizzano per l'identificazione degli attacchi tramite una ricerca di pattern conosciuti. Questo permette ottimi risultati nell'identificare attacchi conosciuti ma non con nuovi attacchi.

Anomaly-based Intrusion Detection System

I Anomaly-based Intrusion Detection System sono stati creati per identificare nuove tipologie di attacchi con la veloce evoluzione di quest'ultimi. L'approccio consiste nell'utilizzo di algoritmi di machine learning per creare un modello sull'attività sicura della rete e applicarlo successivamente su tutto il traffico. Questo approccio potrebbe generare un rischio maggiore di false positive e un deterioramento delle prestazioni con il tempo.

I Network Traffic Analysis (NTA) sono un altro tipo di Anomaly-based IDS che considera attacchi esterni e insider dannosi che mirano al compromettere un account o una macchina.

IDS comparati ai firewalls

Gli IDS differenziano dai firewalls perché i firewalls utilizzano regole statiche per permettere o bloccare connessioni tra la sottorete e un'altra rete esterna, non segnalando attacchi interni. Gli IDS, invece, segnalano un possibile attacco senza prendere provvedimenti diretti.

Il miglior posizionamento di un IDS può variare a seconda della rete, solitamente il posto migliore e subito dopo il firewall nella direzione di una connessione da rete a sottorete, questo perché una volta applicate le regole statiche, il ruolo dell'IDS e provare a identificare i possibili attacchi che bypassano quest'ultime.

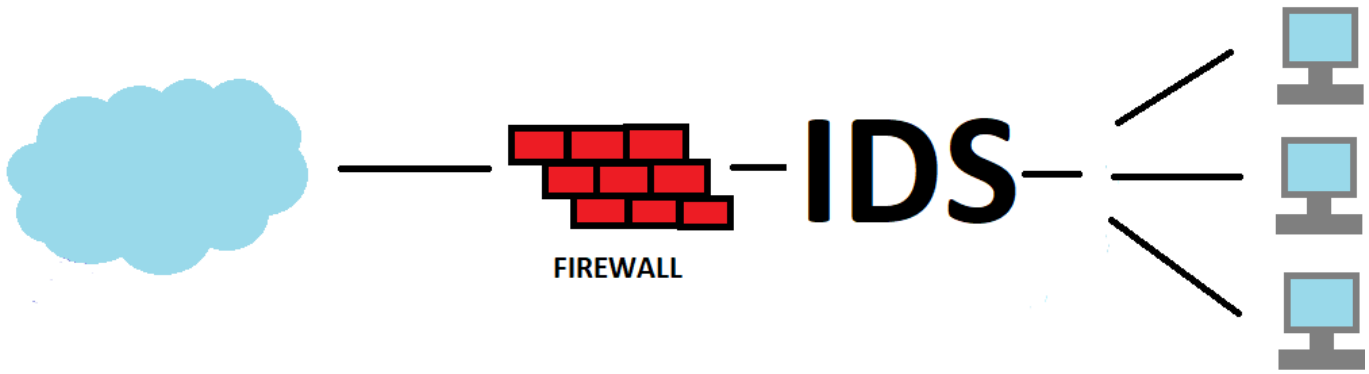

Figura 1 Posizionamento IDS

L'IDS in questo caso specifico

Alla luce dei dettagli precedentemente descritti, l'IDS che si è simulato in questo studio è un IDS applicato all'intera sottorete e caratterizzato come Anomaly-based visto l'utilizzo di algoritmi di machine learning.

Dataset UNSW-NB15

Il Dataset UNSW-NB15 è stato creato dallo strumento IXIA PerfectStorm nel laboratorio Cyber Range Lab of UNSW Canberra rilasciato nel 2015 (Moustafa & Slay, 2015) per contenere attività normale di una rete e degli attacchi sintetici a quest'ultima. I 100 GB di traffico sono stati catturati tramite il tcpdump, con un totale di 2 milioni e 540 044 record con 49

features ciascuno, generate da Argus e Bro-IDS con 12 algoritmi sviluppati appositamente, i loro significati, nello specifico, sono esplicati all'interno del file UNSW-NB15_features.csv. Questi dati sono stati salvati in 4 file csv ottenendo un Dataset ampio che si contraddistingue per la mole di dati al suo interno.

Tipologia di attacchi

All'interno del Dataset sono presenti 9 diverse tipologie di attacchi, classificati direttamente dall'autore di quest'ultimo, queste sono:

Generic

Gli attacchi classificati come Generic sono i più comuni e rappresentano gli attacchi che mirano a decriptare la chiave dei sistemi di sicurezza.

Exploits

Gli attacchi classificati come Exploits rappresentano gli attacchi che mirano all'utilizzo di vulnerabilità, errori e/o glitch dei software e dei sistemi.

Fuzzers

Gli attacchi classificati come Fuzzers rappresentano gli attacchi dove l'attaccante cerca di scoprire falle nel sistema operativo, programma o network, rendendo queste risorse inutilizzabili per un periodo o per farle crashare.

DoS

Gli attacchi classificati come Dos rappresentano gli attacchi dove l'attaccante tenta di bloccare delle risorse caricando la mole di lavoro da gestire.

Reconnaissance

Gli attacchi classificati come Reconnaissance rappresentano gli attacchi che tentano di prendere informazioni rispetto alla rete per cercare di superare la sicurezza.

Analysis

Gli attacchi classificati come Analysis rappresentano gli attacchi che entrano nelle web applications attraverso operazioni di analisi per ricercare vulnerabilità, come lo scanning, l'invio di script malevoli o la distribuzione di spam e-mail.

Backdoor

Gli attacchi classificati come Backdoor rappresentano gli attacchi che bypassano l'autenticazione ottenendo comunque accesso ad un sistema dove è necessario un'autorizzazione.

Shellcode

Gli attacchi classificati come Shellcode rappresentano gli attacchi dove l'attaccante riesce a far eseguire del codice per accedere alla shell prendendo così controllo della macchina.

Worm

Gli attacchi classificati come Worm rappresentano la minoranza in questo Dataset, questi malware si duplicano per espandersi attraverso la rete attaccando un computer all'interno di quest'ultima.

Bilanciamento del Dataset

Trattandosi di un Dataset fortemente sbilanciando contenendo 2,218,761 records di normale traffico e il resto dei 321,465 che rappresentano gli attacchi, tra cui solo 174 sono quelli classificati come Worm, era necessario un bilanciamento dei dati per poter utilizzare algoritmi di machine learning, questo perché si otterrebbe un modello che predirebbe sempre uno stesso risultato, cioè il fatto che non si tratti di un attacco visto la probabilità elevata che si tratti di traffico non rischioso.

Per sopperire al problema lo stesso autore del Dataset ha prodotto una divisione in Train e Test per l'applicazione di algoritmi di machine learning.

Train

Queste sono le occorrenze del Dataset utilizzato come Train, dove con Normal si intendono i record che rappresentano il normale traffico senza attacchi.

Normal	56000
Generic	40000

Exploits	33393
Fuzzers	18184
DoS	12264
Reconnaissance	10491
Analysis	2000
Backdoor	1746
Shellcode	1133
Worms	130

Tabella 1 Train

Test

Queste sono le occorrenze del Dataset utilizzato come Test, dove con Normal si intendono i record che rappresentano il normale traffico senza attacchi.

Normal	37000
Generic	18871
Exploits	11132
Fuzzers	6062
DoS	4089
Reconnaissance	3496
Analysis	677
Backdoor	583
Shellcode	378
Worms	44

Tabella 2 Test

Ambiente

Per realizzare la seguente analisi si è utilizzato un computer con le seguenti specifiche:
Processore Intel(R) Core(TM) i7-8750H CPU 2.20GHz 2.21 GHz,
16 GB di RAM con sistema operativo Windows 10 Home.
Per l'esecuzione del codice si è utilizzato Python come linguaggio tramite l'utilizzo di Anaconda che a sua volta utilizza Jupyter, questo si è scelto per la facilità nell'eseguire operazioni di scientific computing dei software sopra citati.

Algoritmi analizzati

Nel seguente studio si sono presi in considerazione 5 diversi algoritmi forniti dalla libreria di Python SciKit-Learn (https://scikit-learn.org/stable/), più nello

specifico si sono presi 5 dei più utilizzati in quest'ambito e dei più semplici per quanto riguarda il loro funzionamento. Ad ogni algoritmo è stato eseguito un tuning dell'iper-parametro richiesto dalla libreria tramite cross-validation nel Dataset Train, i vari valori provati sono stati selezionati considerando la potenza di calcolo a disposizione e alla loro possibile significatività nella produzione di risultati. Ogni algoritmo e stato applicato in due diversi casi: il primo per predirne la feature "attack" (la feature "label" successivamente rinominata per chiarezza) che identifica se il record si tratti di un attacco (1) o meno (0), il secondo per predirne la feature "attack-cat" che identifica la tipologia di attacco o se si tratta di traffico normale dei record. Il primo caso rappresenta un'applicazione consueta di questi algoritmi in quest'ambito, mentre il secondo caso è stato applicato come caso studio per l'analisi dell'algoritmo specifico nella classificazione di una tipologia di attacco.

K-nearest neighbors

L'algoritmo di Knn (K-nearest neighbors) si basa verificare le K istanze più vicine in base ad una distanza cambiabile, generalmente si utilizza la distanza Euclidea, per predirne il valore. Più nello specifico l'algoritmo selezionati i K elementi più vicini tra quelli che conosce restituisce il risultato maggiormente presente in quest'ultimi. L'iper-parametro più importante è K, cioè la quantità di elementi vicini presi in considerazione e solitamente più piccolo è come valore più è preciso e tendente all'overfitting.

Decision Tree

L'algoritmo di Decision Tree è sicuramente uno dei più utilizzati su una grande quantità di dati. Il suo funzionamento si basa sulla creazione di vari nodi che creano una divisione binaria di possibili strade che conducono a delle foglie. I nodi all'interno hanno una condizione basata su una feature e sul suo valore e le due divisioni si creano se quella condizione viene rispettata o meno. Le foglie rappresentano la predizione finale che non sfocia in altre divisioni e il criterio per diventare foglia o meno si basa sulla informazione, che può essere determinata in diversi criteri, che si ottiene in caso di ulteriore divisione.
Questo algoritmo come iper-parametro più importante ha il numero di foglie massime che è incisivo sul risultato in maniera significativa rappresentando la precisione con cui l'algoritmo arriva ad una predizione.

Da questo algoritmo se ne possono ottenere altri 3 che lo usano come base: Random Forest, Bagging e AdaBoost.

Bagging

L'algoritmo di Bagging si basa sull'interpellare diversi Decision Tree per poi prendere il risultato maggiormente predetto e restituirlo. Per la diversificazione dei Decision Tree si prendono vari subset del Dataframe totale e si procede alla creazione su questi, in questo modo ogni Decision Tree avrà una creazione diversa rendendo il risultato finale più significativo. L'iper-parametro più importante da tenere in considerazione è la quantità di Decision Tree creati.

Random Forest

L'algoritmo di Random Forest è una evoluzione dell'algoritmo di Bagging dove, oltre alla selezione di diversi subset del Dataframe per la creazione dei vari Decision Tree, si escludono alcune features e in questo modo la diversità nella creazione è ancora maggiore.

AdaBoost

L'algoritmo di AdaBoost si basa sulla creazione di un Decision Tree, successivamente si crea un altro Decision Tree su un subset che viene estratto cambiando i pesi dei vari record, la probabilità con cui si è selezionati, in modo che le istanze che il Decision Tree ha sbagliato precedentemente siano quelle su cui si basa la creazione dell'algoritmo successivo. Il numero di volte che questo accade è un iper-parametro importante da tenere in considerazione.

Data processing

Inizialmente si è dato uno primo sguardo ai Dataset di Train e di Test e successivamente dopo aver etichettato tutti i record, del Test di un valore e del Train di un altro, tramite una variabile flag (tr_ts), si sono uniti per poter cominciare le operazioni di pulizia dei dati, di trasformazione delle variabili categoriali per poter essere utilizzati dagli algoritmi e di analisi complessiva di tutti i dati in possesso. In più si è aggiunta la features realID per identificare i singoli record oltre al normale index fornito dalla gestione del Dataset della libreria di pandas.

Analisi iniziale del Dataset

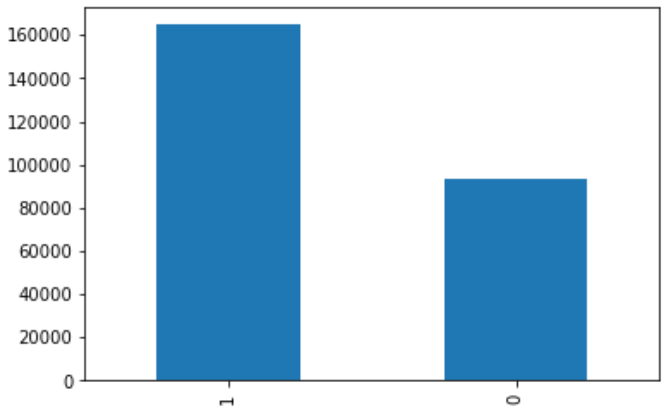

Figura 2 Record attacchi o traffico normale

Questo Bar chart rappresenta il numero di record registrati come attacchi (1) rispetto ai record di traffico normale (0) del Dataset risultante alla concatenazione del Train e del Test. Come si può evincere i due Dataset nel complesso hanno più record classificati come attacchi che normali, questo è comprensibile visto la necessità di aver un numero significativo di tutte le tipologie di attacchi.

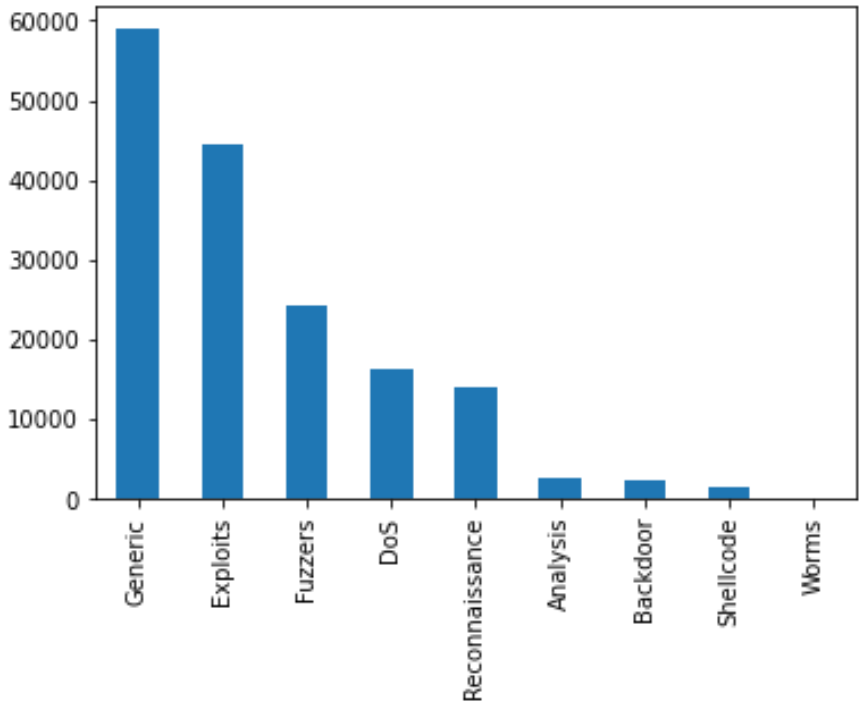

Figura 3 Occorrenze attacchi per tipo

Infatti, nel precedente bar chart si mostrano i vari tipi di attacco e le loro occorrenze nel Dataset risultante, dimostrando comunque un forte sbilanciamento tra le occorrenze dei vari tipi di attacco.

Valori assenti

All'interno dei Dataset di Train e Test non erano presenti istanze con valori a NaN su nessuna features, questo facilita il processo di preparazione dei dati

non dovendo stimare alcun valore o eliminare alcun record.

Il tipo delle features

Per poter applicare gli algoritmi in questione della libreria di SciKit-Learn è necessario che tutte le features siano numeriche.

dur	float64	tcprtt	float64
proto	object	synack	float64
service	object	ackdat	float64
state	object	smean	int64
spkts	int64	dmean	int64
dpkts	int64	trans_depth	int64
sbytes	int64	response_body_len	int64
dbytes	int64	ct_srv_src	int64
rate	float64	ct_state_ttl	int64
sttl	int64	ct_dst_ltm	int64
dttl	int64	ct_src_dport_ltm	int64
sload	float64	ct_dst_sport_ltm	int64
dload	float64	ct_dst_src_ltm	int64
sloss	int64	is_ftp_login	int64
dloss	int64	ct_ftp_cmd	int64
sinpkt	float64	ct_flw_http_mthd	int64
dinpkt	float64	ct_src_ltm	int64
sjit	float64	ct_srv_dst	int64
djit	float64	is_sm_ips_ports	int64
swin	int64	attack_cat	object
stcpb	int64	tr_ts	int64
dtcpb	int64	attack	int64
dwin	int64	realID	int64

Tabella 3 Tipi features

Le precedenti features si possono distinguere in 4 classi in base al significato, inserito all'interno del file UNSW-NB15_features.csv, e sono le seguenti:

-Categorical columns: proto, service, state, attack_cat.

-Binary columns: is_sm_ips_ports, is_ftp_login, attack.

-Numerical columns: id, dur, spkts, dpkts, sbytes, dbytes, rate, sttl, dttl, sload, dload, sloss, dloss, sinpkt, dinpkt, sjit, djit, swin, stcpb, dtcpb, dwin,

tcprtt, synack, ackdat, smean, dmean, trans_depth, response_body_len.

-Ordered quantitative columns: ct_srv_src, ct_state_ttl, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_ftp_cmd, ct_flw_http_mthd, ct_src_ltm, ct_srv_dst.

Le variabili numeriche sono variabili che assumono un qualsiasi valore numerico e solitamente sono distinti per la grande quantità di valori unici all'interno del Dataset.
Le variabili quantitative ordinate sono variabili che rappresentano una quantità e sono facilmente ordinabili perché si distingue un ordine.
Le variabili binarie si possono distinguere perché possono assumere solamente due valori, solitamente 1 e 0.
Le variabili categoriali sono variabili che possono assumere più valori non numerici solitamente rappresentati con delle stringhe come in questo caso.

Le features proto, service, state e attack_cat sono le uniche che necessitano di un cambiamento di tipo trattandosi di variabili categoriali e salvate all'interno del Dataset con il tipo object. Analizzandoli più nello specifico sono presenti rispettivamente 133, 13, 11 e 9 valori differenti all'interno dei Dataset, questo rende la Label encoding l'opzione più corretta visto la poca praticità di rappresentare un numero di features in base ai valori che può assumere una di queste colonne, inserendo 0 o 1 se rappresentato quel valore.
Per trasformale si è utilizzato il Label Encoder che offre la libreria SciKit-Learn che poi verrà riutilizzato per ritrasformare in stringhe i valori per questione di leggibilità nel caso di attack_cat.

Correlazione delle features

Università
Ca'Foscari
Venezia

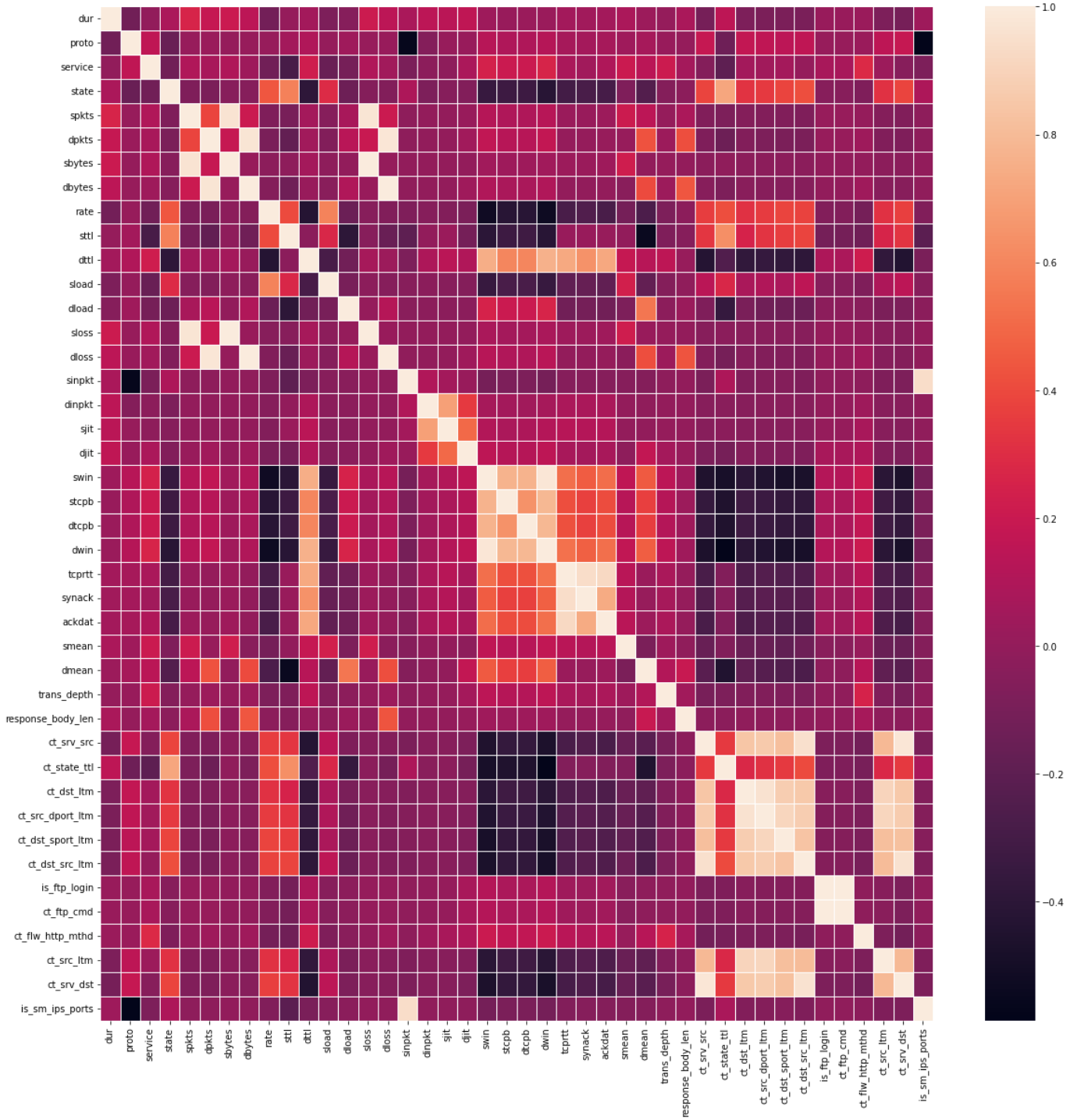

Figura 4 Heatmap correlazione tra le features

La precedente heatmap mostra le correlazioni tra le varie features, in particolare si possono notare le seguenti osservazioni:
-Molte delle variabili che rappresentano il numero di connessioni sembrano essere correlate tra loro (ct_srv_src, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_src_ltm, ct_srv_dst).
-Le variabili che danno informazioni sull'ack sono in correlazione tra loro (tcprtt, synack, ackdat).

-Alcune variabili che rappresentano informazioni sulla source (spkts, sbytes, sloss) sono correlate tra di loro come per quelle che rappresentano informazioni sul destination (dpkts, dbytes, dloss).
-Le due variabili is_ftp_login e ct_ftp_cmd sono fortemente correlate.
-Le due variabili is_sm_ips_ports e sinpkt sembrano essere correlate. -
-Alcune variabili che rappresentano informazioni sul protocollo TCP (swin, dwin, stcpb, dtcpb) sono correlate.

Per gestire queste correlazioni solitamente si procede all'eliminazione di alcune variabili, tuttavia in questo caso si è deciso di lasciare le variabile e successivamente eliminarle nel caso influiscano su alcuni algoritmi che tendono ad utilizzare tutte le features possibili.

Analisi su valori ripetuti

Per procedere ad un'analisi complessiva trattandosi di record che rappresentano il traffico di una rete si è pensato di osservare se erano presenti dei valori ripetuti tra le connessioni, come potrebbero essere uno stesso IP di destination/source, una stessa porta utilizzata o uno stesso protocollo utilizzato.

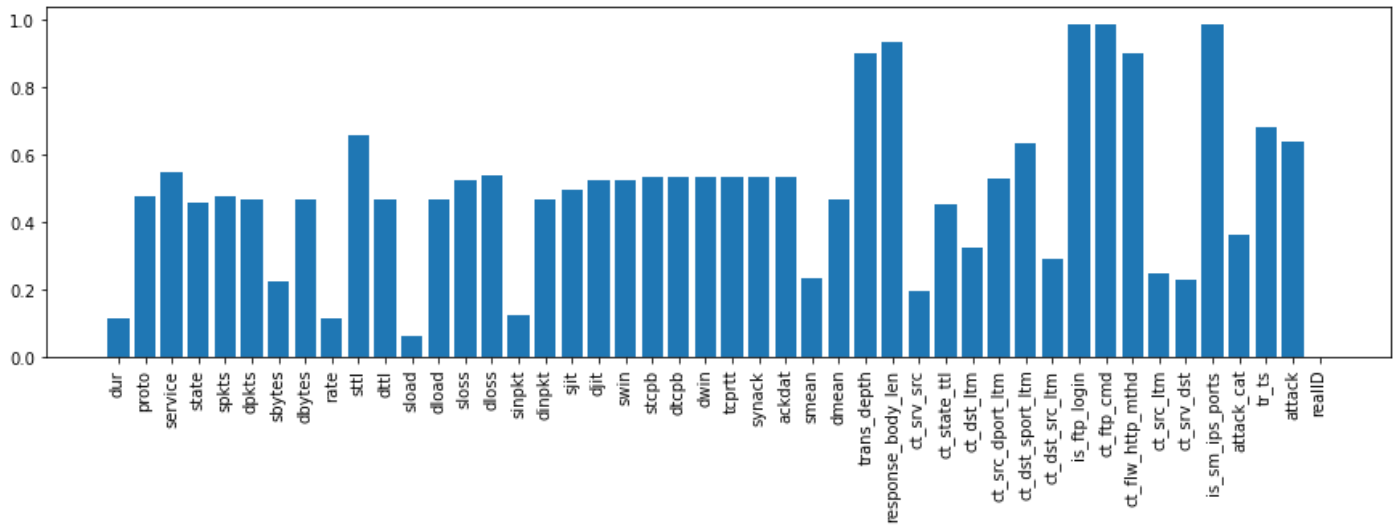

Figura 5 Percentuale presenza di valori uguali sul traffico normale

Il precedente bar chart mostra nell'asse delle x le varie features, mentre nell'asse delle y la percentuale di presenza di un determinato valore in tutti e due i Dataset delle istanze classificate come normale traffico, in questo modo si è in grado di capire se un determinato valore sia solito essere presente. I valori con un'occorrenza superiore all'80% di tutti i record sono:

column	value	percentage

trans_depth	0.0	0.900624
response_body_len	0.0	0.908796
ct_dst_sport_ltm	1.0	0.923215
is_ftp_login	0.0	0.985849
ct_ftp_cmd	0.0	0.985828
ct_flw_http_mthd	0.0	0.900624
is_sm_ips_ports	0.0	0.960452
attack_cat	-1.0	1.000000
attack	0.0	1.000000

Tabella 4 Percentuale presenza di valori uguali sul traffico normale

La precedente tabella mostra i valori interessati ma non ci sono risultati particolarmente utili visto che si mostra che alcune variabili quantitative ordinate e le variabili binarie sono quelle con la maggior presenza di un singolo valore tra tutti i record come era intuibile.
Si può comunque già notare come trans_depth e response_body_len, variabili che rappresentano la risposta del destination, abbiano sempre un valore a 0 nel 90% delle occorenze.
Questa analisi risulta più interessante se fatta sui gruppi di attacchi e poi confrontati con il normale traffico per verificare se ci sono differenze.
Successivamente si illustrano gli stessi bar chart di quello realizzato per il traffico normale solamente applicato alle istanze degli attacchi di uno specifico tipo:

Generic

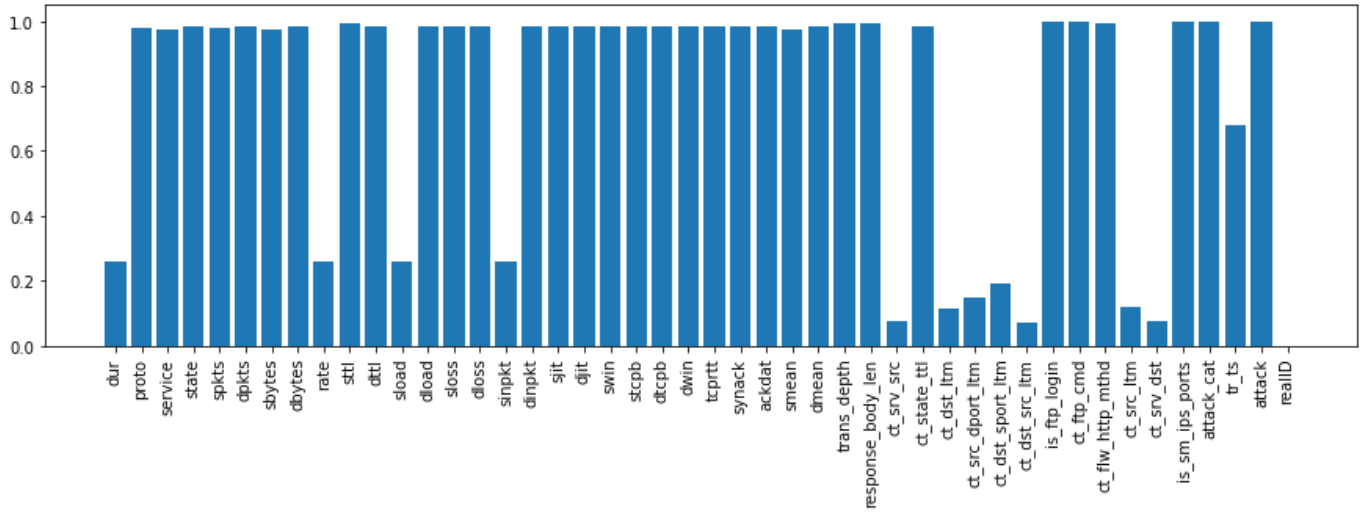

Figura 6 Percentuale presenza di valori uguali sugli attacchi Generic

column	value	percentage	generic_value	generic_percentage
proto	113.0	0.720097	119.0	0.977255
service	0.0	0.686957	2.0	0.972941
state	4.0	0.659645	5.0	0.981961
dpkts	2.0	0.148344	0.0	0.982215
sbytes	146.0	0.076516	114.0	0.972635
sttl	31.0	0.603839	254.0	0.992169
dttl	29.0	0.603473	0.0	0.982215
swin	255.0	0.719882	0.0	0.982912
dwin	255.0	0.692774	0.0	0.982912
smean	73.0	0.077204	57.0	0.972737
ct_state_ttl	0.0	0.610796	2.0	0.981961
attack_cat	-1.0	1.000000	5.0	1.000000

attack	0.0	1.000000	1.0	1.000000

Tabella 5 Percentuale presenza di valori uguali sugli attacchi Generic

Questa tabella mostra il valore maggiormente inserito in una determinata colonna (column) del traffico normale e la sua percentuale (value, percentage) e il valore maggiormente trovato negli attacchi classificati come Generic e la sua percentuale di comparsa (generic_value, generic_percentage) per ogni riga. Le colonne vengono selezionate nel caso in cui ci sia una differenza di valore maggiormente inserito tra il traffico normale e quello degli attacchi Generic e se quest'ultimo abbia un'occorrenza maggiore dell'80%.

Da questa tabella si può capire come gli attacchi di tipo Generic abbiano diversi valori per cui si distinguono dal normale traffico della rete:
- Nel 97% delle istanze si è utilizzato il protocollo udp, mentre nel traffico normale il 72% era eseguito con il protocollo tcp
- Nel 97% delle istanze si utilizza il servizio dns, mentre normalmente se ne utilizzano altri non descritti nel 69% dei casi.
- Nel 98% delle istanze si ha uno stato del protocollo INT, mentre normalmente nel 66% dei casi si ha uno stato FIN.
- Nel 98% delle istanze si ha 0 come numero di pacchetti da destination a source, mentre normalmente può variare.
- Nel 97% delle istanze si ha 114 bytes trasferiti da source a destination, mentre normalmente può variare.
- Nel 99% delle istanze si ha 254 time to live value da source a destination, mentre normalmente si ha come valore 31 nel 60% dei casi.
- Nel 98% delle istanze si ha 0 come time to live value da destination a source, mentre normalmente si ha come valore 29 nel 60% dei casi.
- Nel 98% delle istanze si ha 0 come source e destination TCP window advertisement, mentre normalmente si ha come valori 255 rispettivamente nel 72% e nel 69% dei casi.
- Nel 97% delle istanze si ha 57 come media della dimensione del pacchetto spedito dalla source, mentre normalmente varia.
- Nel 98% delle istanze si ha 2 come numero per ogni stato secondo uno specifico range di valori per source e destination time to live, mentre normalmente è 0 nel 61% dei casi.

(attack_cat e attack non sono prese in considerazione visto che rappresentano le due variabili di interesse).
Analizzando in questo modo gli attacchi di tipo Generic si possono già intuire alcune interessanti peculiarità. Questi tipi di attacchi sono i più

presenti in entrambi i Dataset e sembrano anche distinguersi per numerose features rispetto al normale traffico della rete, in più per aver percentuali così alte di occorrenze di un singolo valore dovrebbero essere facilmente riconoscibili dagli algoritmi presi in considerazione.

Exploits

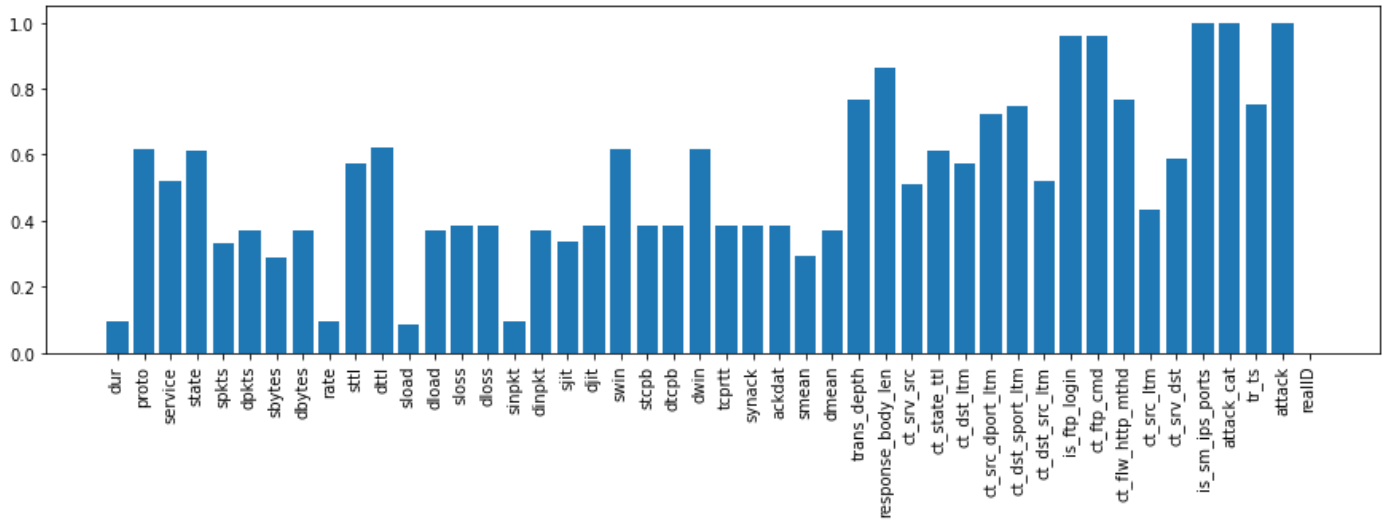

Figura 7 Percentuale presenza di valori uguali sugli attacchi Exploits

Le istanze degli attacchi classificati Exploits, selezionando le colonne come in precedenza, non sembrano distinguersi per nessuna occorrenza sopra all'80% delle istanze di un valore differente rispetto al normale traffico.

Fuzzers

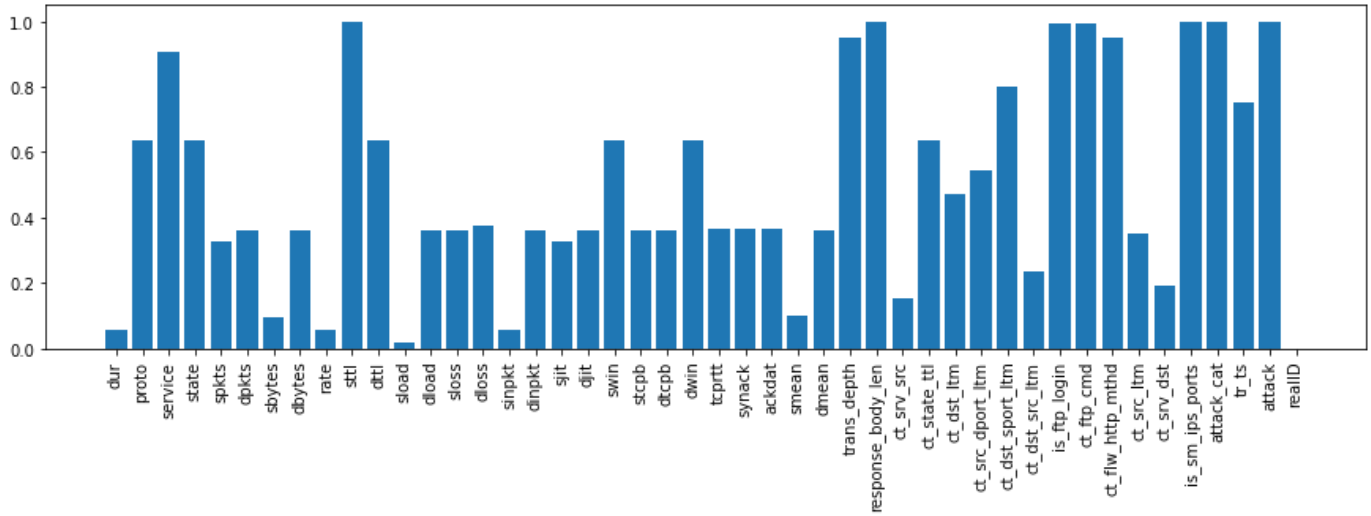

Figura 8 Percentuale presenza di valori uguali sugli attacchi Fuzzers

column	value	percentage	fuzzers_value	fuzzers_percentage

sttl	31.0	0.603839	254.0	0.996494
attack_cat	-1.0	1.000000	4.0	1.000000
attack	0.0	1.000000	1.0	1.000000

Tabella 6 Percentuale presenza di valori uguali sugli attacchi Fuzzers

Selezionando le colonne allo stesso modo, gli attacchi di tipo Fuzzers sembrano distinguersi dal traffico normale solamente per un time to live value da source a destination di 254 nel 99% delle istanze.

Reconnaissance

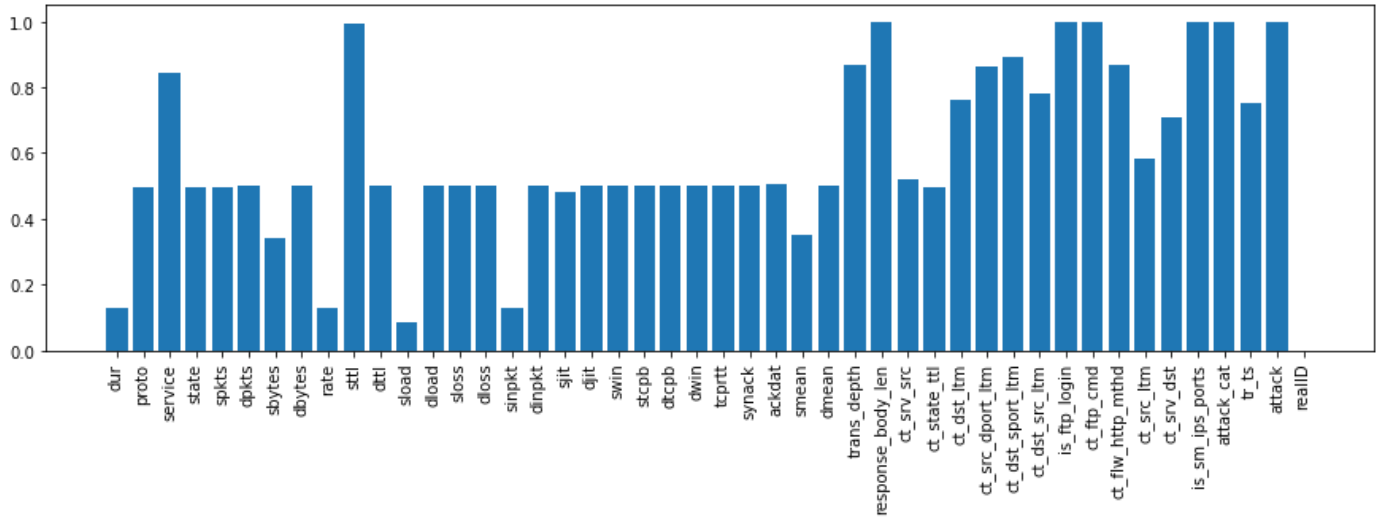

Figura 9 Percentuale presenza di valori uguali sugli attacchi Reconnaissance

column	value	percentage	reconnaissance_value	reconnaissance_percentage
sttl	31.0	0.603839	254.0	0.995353
attack_cat	-1.0	1.000000	6.0	1.000000
attack	0.0	1.000000	1.0	1.000000

Tabella 7 Percentuale presenza di valori uguali sugli attacchi Reconnaissance

Selezionando le colonne allo stesso modo, gli attacchi di tipo Reconnaissance sembrano distinguersi dal traffico normale solamente per un time to live value da source a destination di 254 nel 99% delle istanze.

DoS

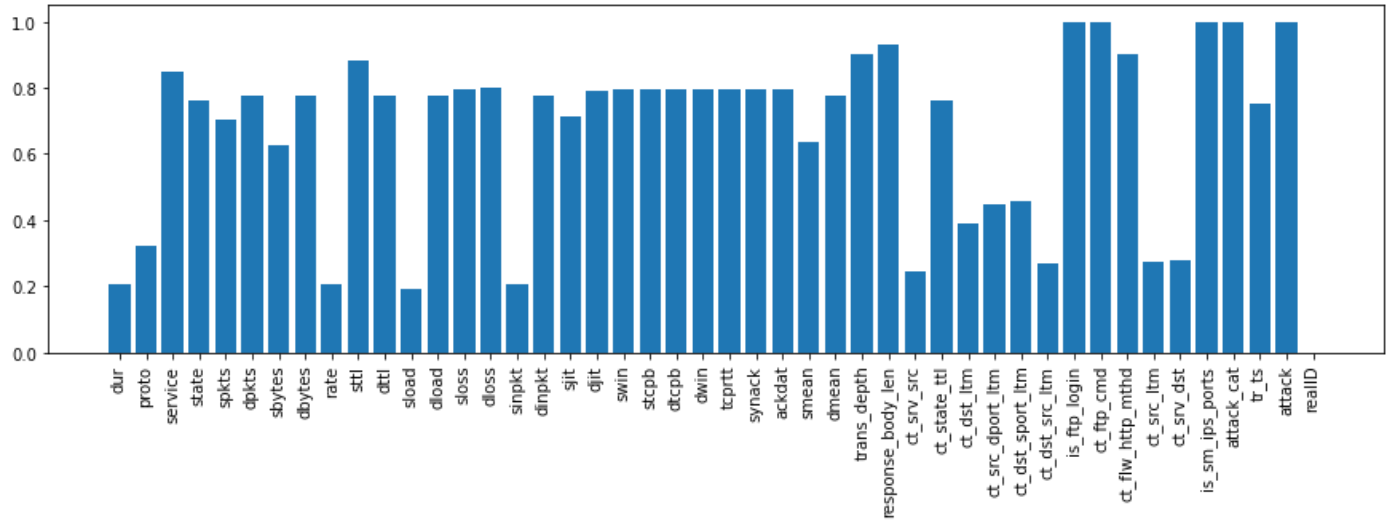

Figura 10 Percentuale presenza di valori uguali sugli attacchi DoS

column	value	percentage	dos_value	dos_percentage
sttl	31.0	0.603839	254.0	0.880634
attack_cat	-1.0	1.000000	2.0	1.000000
attack	0.0	1.000000	1.0	1.000000

Tabella 8 Percentuale presenza di valori uguali sugli attacchi DoS

Selezionando le colonne allo stesso modo, gli attacchi di tipo DoS sembrano distinguersi dal traffico normale solamente per un time to live value da source a destination di 254 nel 88% delle istanze.

Backdoors

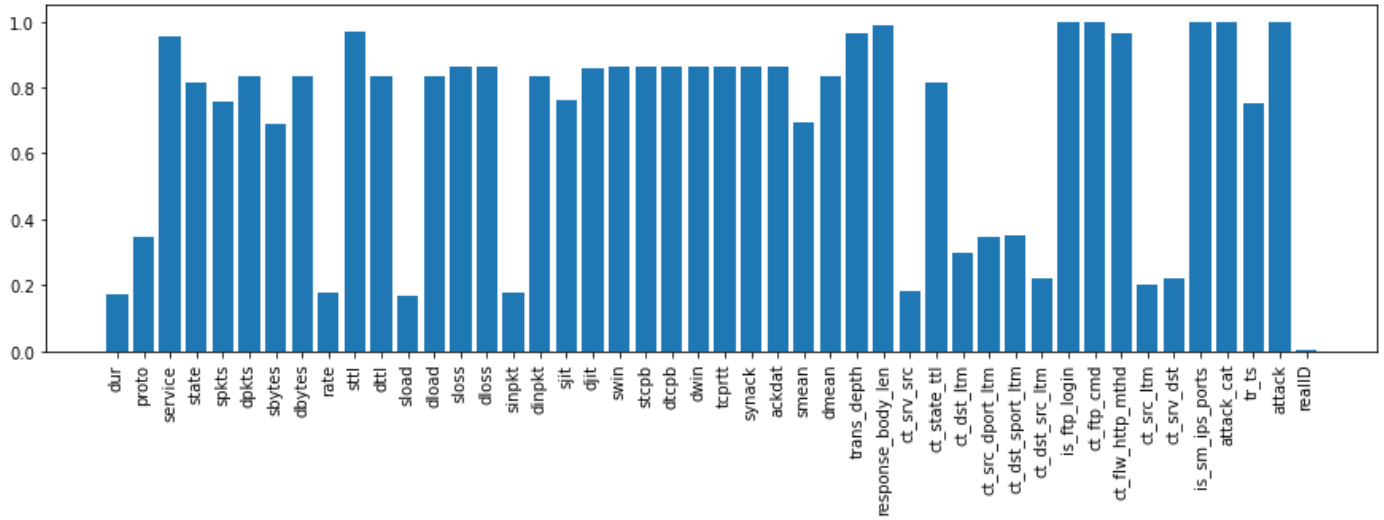

Figura 11 Percentuale presenza di valori uguali sugli attacchi Backdoors

column	value	percentage	backdoors_value	backdoors_percentage
sttl	31.0	0.603839	254.0	0.970803
attack_cat	-1.0	1.000000	1.0	1.000000
attack	0.0	1.000000	1.0	1.000000

Tabella 9 Percentuale presenza di valori uguali sugli attacchi Backdoors

Selezionando le colonne allo stesso modo, gli attacchi di tipo Backdoors sembrano distinguersi dal traffico normale solamente per un time to live value da source a destination di 254 nel 88% delle istanze.

Analysis

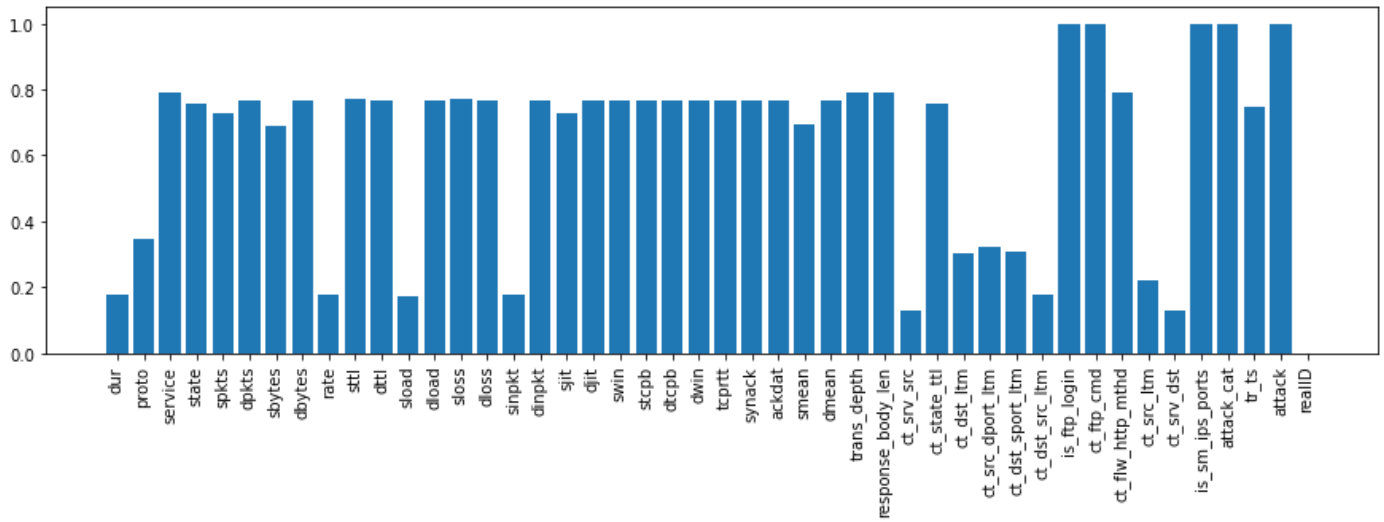

Figura 12 Percentuale presenza di valori uguali sugli attacchi Analysis

Le istanze degli attacchi classificati Analysis, selezionando le colonne come in precedenza, non sembrano distinguersi per nessuna occorrenza sopra all'80% delle istanze di un valore differente rispetto al normale traffico.

Shellcode

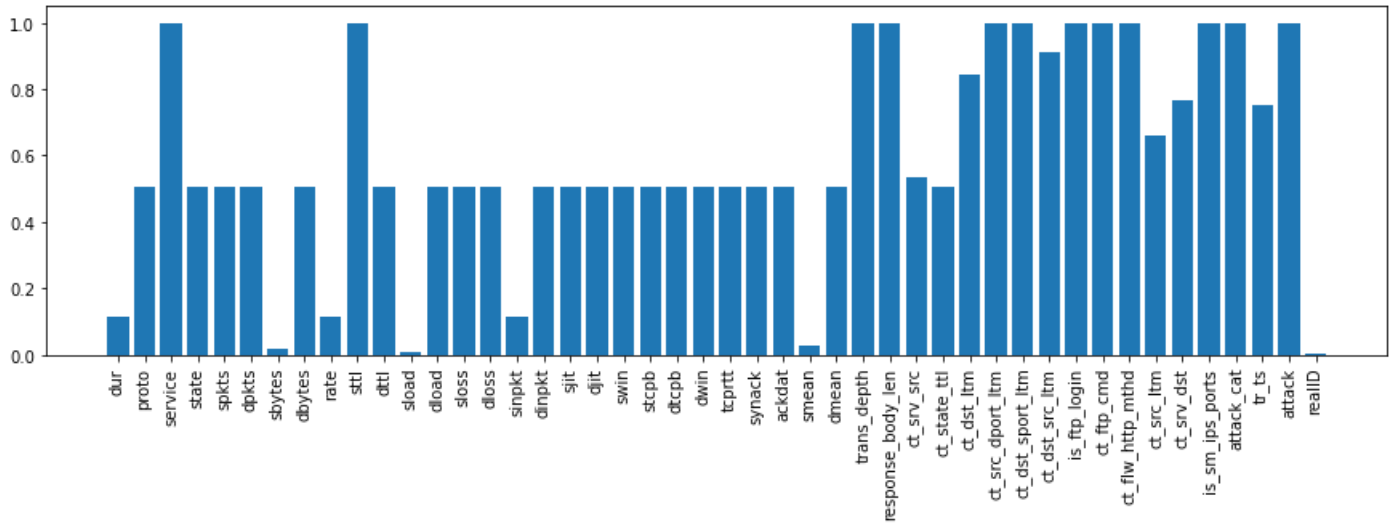

Figura 13 Percentuale presenza di valori uguali sugli attacchi Shellcode

column	value	percentage	shellcode_value	shellcode_percentage
sttl	31.0	0.603839	254.0	1.0
attack_cat	-1.0	1.000000	7.0	1.0
attack	0.0	1.000000	1.0	1.0

Tabella 10 Percentuale presenza di valori uguali sugli attacchi Shellcode

Selezionando le colonne allo stesso modo, gli attacchi di tipo Shellcode sembrano distinguersi dal traffico normale solamente per un time to live value da source a destination di 254 nel 100% delle istanze.

Worms

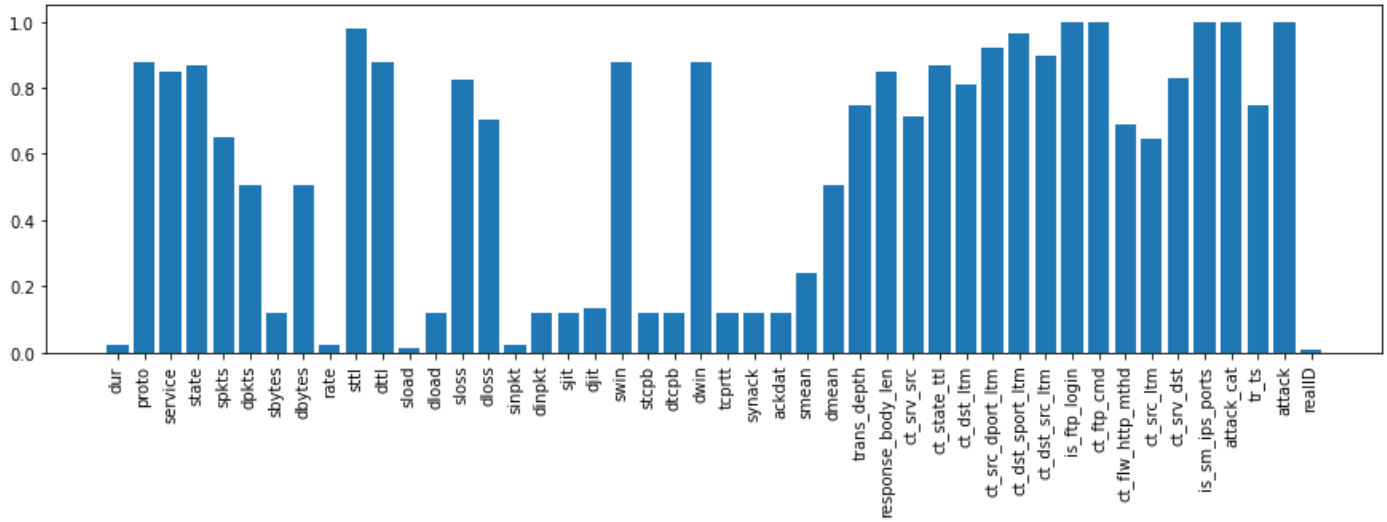

Figura 14 Percentuale presenza di valori uguali sugli attacchi Worms

column	value	percentage	worms_value	worms_percentage
service	0.0	0.686957	5.0	0.850575
sttl	31.0	0.603839	254.0	0.977011
dttl	29.0	0.603473	252.0	0.879310
sloss	0.0	0.280591	2.0	0.821839
ct_state_ttl	0.0	0.610796	1.0	0.867816
ct_srv_dst	2.0	0.173785	1.0	0.827586
attack_cat	-1.0	1.000000	8.0	1.000000
attack	0.0	1.000000	1.0	1.000000

Tabella 11 Percentuale presenza di valori uguali sugli attacchi Worms

Selezionando le colonne allo stesso modo, gli attacchi di tipo Worms sembrano distinguersi, almeno in questi pochi record classificati in questo modo presenti in questo Dataset, per diversi aspetti:
- Nel 85% delle istanze si ha http come servizio utilizzato, mentre normalmente se ne utilizzano altri non descritti nel 69% dei casi.
- Nel 98% delle istanze si ha 254 time to live value da source a destination, mentre normalmente si ha come valore 31 nel 60% dei casi.

- Nel 98% delle istanze si ha 252 time to live value da destination a source, mentre normalmente si ha come valore 29 nel 60% dei casi.
- Nel 82% delle istanze si ha 2 come pacchetti ritrasmessi o cancellati da source, mentre normalmente può variare.
- Nel 87% delle istanze si ha 2 come numero per ogni stato secondo uno specifico range di valori per source e destination time to live, mentre normalmente è 0 nel 61% dei casi.
- Nel 83% delle istanze si ha 1 come numero di connessioni con lo stesso destination e servizio in 100 connessioni secondo l'ultima fatta, mentre normalmente può variare.

Analisi complessiva dei precedenti grafici

Guardando nel complesso i grafici precedenti si può notare che escludendo Generic e Worms gli altri tipi di attacchi non sembrano distinguersi dal traffico normale, seguendo questa prima analisi sulla ripetizione di un determinato valore, se non per la colonna sttl, escludendo gli attacchi di tipo Analysis ed Exploits che non hanno valori ripetuti estranei.
Anche se è un'analisi che non da dei risultati concreti sull'andamento di un possibile algoritmo, già da qui si può intuire che sttl sarà una delle variabili di maggiore importanza nella creazione di questi modelli di predizione.

Analisi sulla media di variabili numeriche

Successivamente all'analisi precedente si è voluto inserire un'analisi più approfondita sulle variabili numeriche valutandone la media. All'interno di questa analisi si sono anche inserite le variabili quantitative ordinali che comunque hanno una media che può indicare qualche informazione.
Per la realizzazione di questa analisi si è assunto l'andamento normale dei dati e l'indipendenza tra il traffico normale dei dati e i vari attacchi, questo per utilizzare il test t di Student, della libreria di Python scipy (https://scipy.org/), che permette la verifica di differenze statisticamente significative tra due medie di campioni indipendenti data la normalità dei dati.
Realizzato quindi un Dataframe avente come righe le varie features e come colonne le varie categorie di traffico prese in considerazione e riempito con le medie di ogni feature si è realizzato successivamente un altro Dataframe simile con però esclusa la colonna rappresentativa del traffico normale, perché all'interno era presente il p-value del t test risultante dal confronto tra le varie categorie di attacchi e il traffico normale. Per considerare una media significativamente differente si sono tenuti i p-value minori del 0.05, risultando ai risultati seguenti analizzati per gli attacchi di ogni tipo:

Generic

columns	normal	generic
dur	1.015407e+00	6.115685e-02
spkts	2.756317e+01	2.576770e+00
dpkts	3.302794e+01	1.159926e+00
sbytes	4.092444e+03	4.462890e+02
dbytes	2.613820e+04	1.319714e+03
rate	1.958829e+04	2.098332e+05
sttl	9.489008e+01	2.524860e+02
dttl	8.184868e+01	4.348015e+00
sload	2.976812e+07	9.848884e+07
dload	1.788697e+06	3.441147e+03
sloss	4.729785e+00	1.553057e-01
dloss	1.217184e+01	5.230589e-01
sinpkt	2.344196e+03	2.631516e+00
dinpkt	1.427879e+02	1.705135e+00
sjit	6.491785e+03	1.195731e+02
djit	8.375593e+02	3.241149e+01
swin	1.835943e+02	4.357494e+00
stcpb	1.491325e+09	3.750823e+07

dtcpb	1.480589e+09	3.634370e+07
dwin	1.766758e+02	4.357494e+00
tcprtt	4.908785e-02	2.061324e-03
synack	2.675136e-02	9.859433e-04
ackdat	2.233650e-02	1.075381e-03
smean	1.454127e+02	6.195687e+01
dmean	2.214310e+02	7.626658e+00
trans_depth	1.001183e-01	7.626845e-03
response_body_len	3.218943e+03	4.616946e+02
ct_srv_src	5.829548e+00	2.466320e+01
ct_state_ttl	6.957204e-01	1.983693e+00
ct_dst_ltm	3.431656e+00	1.713733e+01
ct_src_dport_ltm	1.779376e+00	1.692815e+01
ct_dst_sport_ltm	1.148581e+00	1.299193e+01
ct_dst_src_ltm	3.889355e+00	2.400489e+01
ct_ftp_cmd	1.425806e-02	0.000000e+00
ct_flw_http_mthd	1.431075e-01	8.051502e-03
ct_src_ltm	3.978978e+00	1.728724e+01
ct_srv_dst	5.535043e+00	2.465139e+01

Tabella 12 Media features dopo t test per attacchi Generic

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo Generic sembrano differire in media per numerosi aspetti dal traffico normale, più nello specifico nell'ordine rappresentato nella tabella:

- Una durata minore (dur).
- Un numero minore di pacchetti da source a destination (spkts).
- Un numero minore di pacchetti da destination a source (dpkts).
- Un numero minore di bytes da source a destination (sbytes).
- Un numero minore di bytes da destination a source (dbytes).
- Un rate maggiore (rate).
- Un valore di time to live da source a destination maggiore (sttl).
- Un valore di time to live da destination a source minore (dttl).
- Un numero maggiore di bits al secondo di source (sload).
- Un numero minore di bits al secondo di destination (dload).
- Un numero minore di pacchetti ritrasmessi o cancellati di source (sloss).
- Un numero minore di pacchetti ritrasmessi o cancellati di destination (dloss).
- Un tempo minore di arrivo di pacchetti intermedi di source (sinpkt).
- Un tempo minore di arrivo di pacchetti intermedi di destination (dinpkt).
- Un tempo minore di jitter di source (sjit).
- Un tempo minore di jitter di destination (djit).
- Un minor valore di TCP window advertisement di source (swin).
- Un numero minore di TCP base sequence di source (stcpb).
- Un numero minore di TCP base sequence di destination (dtcpb).
- Un minor valore di TCP window advertisement di destination (dwin).
- Un minor valore della somma di synack e ackdat (tcprtt).
- Una minor media della dimensione del pacchetto ritrasmesso da source (smean).
- Una minor media della dimensione del pacchetto ritrasmesso da destination (dmean).
- Una minor profondità della pipeline della connessione della transizione domanda/risposta http (trans_depth).
- Un minor dimensione degli effettivi dati trasmessi dal server del servizio http (response_body_len).
- Un numero maggiore di connessione con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_src).
- Un numero maggiore di per ogni stato secondo uno specifico range di valori per il time to live da source a destination (ct_state_ttl).
- Un numero maggiore di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).
- Un numero maggiore di connessioni con lo stesso indirizzo di source e la stessa porta di destination in 100 connessioni secondo l'ultima fatta

(ct_dst_dport_ltm).
- Un numero maggiore di connessioni con lo stesso indirizzo di destination e la stessa porta di source in 100 connessioni secondo l'ultima fatta (ct_dst_sport_ltm).
- Un numero maggiore di connessioni con gli stessi indirizzi di source e destination in 100 connessioni secondo l'ultima fatta (ct_dst_src_ltm).
- Uno 0 come numero di flussi che hanno un comando in una sessione ftp rispetto al traffico normale (ct_ftp_cmd).
- Un minor numero di flussi con metodi come Get o Post in servizi http (ct_flw_http_mthd).
- Un maggior numero di connessioni con lo stesso indirizzo di source in 100 connessioni secondo l'ultima fatta (ct_src_ltm).
- Un numero maggiore di connessioni con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_dst).

Questa numerosa lista di differenze in media rispetto il traffico normale rispecchia l'analisi precedente, aggiungendo anche numerose features interessanti che gli algoritmi utilizzati potrebbero utilizzare per identificare i vari attacchi se sono simili anche in altre tipologie di quest'ultimi.

Exploits

columns	normal	exploits
dur	1.015407e+00	2.122902e+00
spkts	2.756317e+01	3.396202e+01
dpkts	3.302794e+01	2.337586e+01
sbytes	4.092444e+03	2.965500e+04
dbytes	2.613820e+04	1.774804e+04
rate	1.958829e+04	6.796271e+04
sttl	9.489008e+01	1.722062e+02
dttl	8.184868e+01	1.567561e+02

sload	2.976812e+07	6.116571e+07
dload	1.788697e+06	5.891044e+04
sloss	4.729785e+00	1.283924e+01
dloss	1.217184e+01	8.644020e+00
sinpkt	2.344196e+03	8.951289e+01
dinpkt	1.427879e+02	5.865072e+01
sjit	6.491785e+03	3.606613e+03
djit	8.375593e+02	1.009459e+03
swin	1.835943e+02	1.571693e+02
stcpb	1.491325e+09	1.326784e+09
dtcpb	1.480589e+09	1.329089e+09
dwin	1.766758e+02	1.571579e+02
tcprtt	4.908785e-02	7.419381e-02
synack	2.675136e-02	3.540950e-02
ackdat	2.233650e-02	3.878431e-02
smean	1.454127e+02	1.887130e+02
dmean	2.214310e+02	1.802631e+02
trans_depth	1.001183e-01	2.511623e-01
response_body_len	3.218943e+03	2.566384e+03

ct_srv_src	5.829548e+00	2.927075e+00
ct_state_ttl	6.957204e-01	1.406917e+00
ct_dst_ltm	3.431656e+00	1.997372e+00
ct_src_dport_ltm	1.779376e+00	1.551106e+00
ct_dst_sport_ltm	1.148581e+00	1.509916e+00
ct_dst_src_ltm	3.889355e+00	2.914565e+00
ct_ftp_cmd	1.425806e-02	4.087591e-02
ct_flw_http_mthd	1.431075e-01	2.549354e-01
ct_src_ltm	3.978978e+00	3.280337e+00
ct_srv_dst	5.535043e+00	2.697586e+00

Tabella 13 Media features dopo t test per attacchi Exploits

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo Exploits sembrano differire in media dal traffico normale per le stesse features di Generic, tuttavia sono presenti delle differenze:
- Una durata maggiore (dur).
- Un numero maggiore di pacchetti da source a destination (spkts).
- Un numero maggiore di bytes da source a destination (sbytes).
- Un valore di time to live da destination a source maggiore (dttl).
- Un numero maggiore di pacchetti ritrasmessi o cancellati di source (sloss).
- Un tempo maggiore di jitter di destination (djit).
- Un maggior valore della somma di synack e ackdat (tcprtt).
- Una maggior media della dimensione del pacchetto ritrasmesso da source (smean).
- Una maggior profondità della pipeline della connessione della transizione domanda/risposta http (trans_depth).
- Un numero minore di connessione con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_src).
- Un numero minore di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).

- Un numero minore di connessioni con lo stesso indirizzo di source e la stessa porta di destination in 100 connessioni secondo l'ultima fatta (ct_dst_dport_ltm).
- Un numero minore di connessioni con gli stessi indirizzi di source e destination in 100 connessioni secondo l'ultima fatta (ct_dst_src_ltm).
- Un numero maggiore di flussi che hanno un comando in una sessione ftp rispetto al traffico normale (ct_ftp_cmd).
- Un maggior numero di flussi con metodi come Get o Post in servizi http (ct_flw_http_mthd).
- Un minor numero di connessioni con lo stesso indirizzo di source in 100 connessioni secondo l'ultima fatta (ct_src_ltm).
- Un numero minore di connessioni con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_dst).

Fuzzers

columns	normal	fuzzers
dur	1.015407e+00	2.673938e+00
spkts	2.756317e+01	1.376483e+01
dpkts	3.302794e+01	6.095686e+00
sbytes	4.092444e+03	6.764303e+03
dbytes	2.613820e+04	5.187721e+02
rate	1.958829e+04	6.727615e+04
sttl	9.489008e+01	2.534079e+02
dttl	8.184868e+01	1.609003e+02
sload	2.976812e+07	1.276930e+08
dload	1.788697e+06	3.221345e+03
sloss	4.729785e+00	3.863648e+00

dloss	1.217184e+01	1.152644e+00
sinpkt	2.344196e+03	4.109509e+02
dinpkt	1.427879e+02	3.184076e+02
sjit	6.491785e+03	2.159782e+04
swin	1.835943e+02	1.627431e+02
stcpb	1.491325e+09	1.368714e+09
dtcpb	1.480589e+09	1.365963e+09
dwin	1.766758e+02	1.627431e+02
tcprtt	4.908785e-02	9.878788e-02
synack	2.675136e-02	5.217042e-02
ackdat	2.233650e-02	4.661746e-02
smean	1.454127e+02	2.388262e+02
dmean	2.214310e+02	3.883160e+01
trans_depth	1.001183e-01	4.846160e-02
response_body_len	3.218943e+03	2.734224e+00
ct_srv_src	5.829548e+00	6.124557e+00
ct_state_ttl	6.957204e-01	1.421224e+00
ct_dst_ltm	3.431656e+00	2.527345e+00
ct_src_dport_ltm	1.779376e+00	2.133548e+00

ct_dst_sport_ltm	1.148581e+00	1.416399e+00
ct_dst_src_ltm	3.889355e+00	4.412975e+00
ct_ftp_cmd	1.425806e-02	5.856636e-03
ct_flw_http_mthd	1.431075e-01	6.330941e-02
ct_src_ltm	3.978978e+00	3.376557e+00
ct_srv_dst	5.535043e+00	5.436691e+00

Tabella 14 Media features dopo t test per attacchi Fuzzers

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo Fuzzers sembrano differire in media dal traffico normale per le stesse features di Generic tranne per djit, tuttavia sono presenti delle differenze:
- Una durata maggiore (dur).
- Un numero maggiore di bytes da source a destination (sbytes).
- Un valore di time to live da destination a source maggiore (dttl).
- Un tempo maggiore di arrivo di pacchetti intermedi di destination (dinpkt).
- Un tempo maggiore di jitter di source (sjit).
- Un maggior valore della somma di synack e ackdat (tcprtt).
- Una maggior media della dimensione del pacchetto ritrasmesso da source (smean).
- Un numero minore di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).
- Un numero minore di flussi che hanno un comando in una sessione ftp rispetto al traffico normale (ct_ftp_cmd).
- Un minor numero di connessioni con lo stesso indirizzo di source in 100 connessioni secondo l'ultima fatta (ct_src_ltm).
- Un numero minore di connessioni con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_dst).

Reconnaissance

columns	normal	reconnaissance
spkts	2.756317e+01	7.034604e+00

dpkts	3.302794e+01	4.983771e+00
sbytes	4.092444e+03	7.513813e+02
dbytes	2.613820e+04	1.797089e+03
rate	1.958829e+04	9.971873e+04
sttl	9.489008e+01	2.530152e+02
dttl	8.184868e+01	1.256823e+02
sload	2.976812e+07	7.049143e+07
dload	1.788697e+06	2.425574e+03
sloss	4.729785e+00	9.982126e-01
dloss	1.217184e+01	1.101237e+00
sinpkt	2.344196e+03	6.995734e+01
dinpkt	1.427879e+02	6.736751e+01
sjit	6.491785e+03	3.509477e+03
djit	8.375593e+02	1.073624e+02
swin	1.835943e+02	1.269804e+02
stcpb	1.491325e+09	1.066815e+09
dtcpb	1.480589e+09	1.079445e+09
dwin	1.766758e+02	1.269804e+02
tcprtt	4.908785e-02	6.002207e-02

synack	2.675136e-02	2.855424e-02
ackdat	2.233650e-02	3.146783e-02
smean	1.454127e+02	7.910117e+01
dmean	2.214310e+02	2.409023e+01
trans_depth	1.001183e-01	1.323372e-01
response_body_len	3.218943e+03	3.086273e+01
ct_srv_src	5.829548e+00	2.555516e+00
ct_state_ttl	6.957204e-01	1.515622e+00
ct_dst_ltm	3.431656e+00	1.667548e+00
ct_src_dport_ltm	1.779376e+00	1.241367e+00
ct_dst_sport_ltm	1.148581e+00	1.211625e+00
ct_dst_src_ltm	3.889355e+00	2.100307e+00
ct_ftp_cmd	1.425806e-02	0.000000e+00
ct_src_ltm	3.978978e+00	2.705012e+00
ct_srv_dst	5.535043e+00	1.985487e+00

Tabella 15 Media features dopo t test per attacchi Reconnaissance

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo Reconnaissance sembrano differire in media dal traffico normale per le stesse features di Generic tranne per dur e ct_flw_http_mthd, tuttavia sono presenti delle differenze:
- Un valore di time to live da destination a source maggiore (dttl).
- Un maggior valore della somma di synack e ackdat (tcprtt).
- Una maggior profondità della pipeline della connessione della transizione

domanda/risposta http (trans_depth).
- Un numero minore di connessione con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_src).
- Un numero minore di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).
- Un numero minore di connessioni con lo stesso indirizzo di destination e la stessa porta di source in 100 connessioni secondo l'ultima fatta (ct_dst_sport_ltm).
- Un numero minore di connessioni con gli stessi indirizzi di source e destination in 100 connessioni secondo l'ultima fatta (ct_dst_src_ltm).
- Un minor numero di connessioni con lo stesso indirizzo di source in 100 connessioni secondo l'ultima fatta (ct_src_ltm).
- Un numero minore di connessioni con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_dst).

DoS

columns	normal	dos
dur	1.015407e+00	2.448795e+00
spkts	2.756317e+01	2.411735e+01
dpkts	3.302794e+01	2.125231e+01
sbytes	4.092444e+03	1.836079e+04
dbytes	2.613820e+04	2.174713e+04
rate	1.958829e+04	1.461659e+05
sttl	9.489008e+01	2.308610e+02
dttl	8.184868e+01	5.274494e+01
sload	2.976812e+07	1.233523e+08
dload	1.788697e+06	1.700825e+04
sloss	4.729785e+00	7.803278e+00

dloss	1.217184e+01	8.477588e+00
sinpkt	2.344196e+03	7.933471e+01
dinpkt	1.427879e+02	1.903270e+01
sjit	6.491785e+03	9.429872e+02
djit	8.375593e+02	1.186085e+02
swin	1.835943e+02	5.201981e+01
stcpb	1.491325e+09	4.356244e+08
dtcpb	1.480589e+09	4.325281e+08
dwin	1.766758e+02	5.201981e+01
tcprtt	4.908785e-02	2.338973e-02
synack	2.675136e-02	1.104527e-02
ackdat	2.233650e-02	1.234446e-02
smean	1.454127e+02	1.391059e+02
dmean	2.214310e+02	4.846848e+01
ct_srv_src	5.829548e+00	4.543998e+00
ct_state_ttl	6.957204e-01	1.831835e+00
ct_dst_ltm	3.431656e+00	2.443711e+00
ct_src_dport_ltm	1.779376e+00	2.123769e+00
ct_dst_sport_ltm	1.148581e+00	2.098025e+00

ct_dst_src_ltm	3.889355e+00	4.686724e+00
ct_ftp_cmd	1.425806e-02	1.406470e-03
ct_flw_http_mthd	1.431075e-01	1.065248e-01
ct_src_ltm	3.978978e+00	4.299945e+00
ct_srv_dst	5.535043e+00	4.404696e+00

Tabella 16 Media features dopo t test per attacchi DoS

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo DoS sembrano differire in media dal traffico normale per le stesse features di Generic tranne per trans_depth e response_body_len, tuttavia sono presenti delle differenze:
- Una durata maggiore (dur).
- Un numero maggiore di bytes da source a destination (sbytes).
- Un numero maggiore di pacchetti ritrasmessi o cancellati di source (sloss).
- Un numero minore di connessione con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_src).
- Un numero minore di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).
- Un numero minore di flussi che hanno un comando in una sessione ftp rispetto al traffico normale (ct_ftp_cmd).
- Un numero minore di connessioni con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_dst).

Backdoors

columns	normal	backdoors
dur	1.015407e+00	2.224558e+00
spkts	2.756317e+01	6.767282e+00
dpkts	3.302794e+01	1.458995e+00
sbytes	4.092444e+03	1.520862e+03

dbytes	2.613820e+04	2.952941e+02
rate	1.958829e+04	1.492100e+05
sttl	9.489008e+01	2.478351e+02
dttl	8.184868e+01	3.657192e+01
sload	2.976812e+07	1.203970e+08
dload	1.788697e+06	2.573607e+03
sloss	4.729785e+00	3.271790e-01
dloss	1.217184e+01	3.598111e-01
sinpkt	2.344196e+03	8.664514e+01
dinpkt	1.427879e+02	1.581287e+01
sjit	6.491785e+03	9.046962e+02
djit	8.375593e+02	4.472223e+01
swin	1.835943e+02	3.536496e+01
stcpb	1.491325e+09	3.078061e+08
dtcpb	1.480589e+09	2.952011e+08
dwin	1.766758e+02	3.536496e+01
tcprtt	4.908785e-02	1.774124e-02
synack	2.675136e-02	8.403958e-03
ackdat	2.233650e-02	9.337280e-03

smean	1.454127e+02	1.032520e+02
dmean	2.214310e+02	1.783856e+01
trans_depth	1.001183e-01	3.649635e-02
response_body_len	3.218943e+03	1.168742e+01
ct_srv_src	5.829548e+00	6.127523e+00
ct_state_ttl	6.957204e-01	1.890940e+00
ct_dst_ltm	3.431656e+00	2.896522e+00
ct_src_dport_ltm	1.779376e+00	2.557321e+00
ct_dst_sport_ltm	1.148581e+00	2.546587e+00
ct_dst_src_ltm	3.889355e+00	4.774581e+00
ct_ftp_cmd	1.425806e-02	0.000000e+00
ct_flw_http_mthd	1.431075e-01	4.594246e-02
ct_src_ltm	3.978978e+00	4.760842e+00
ct_srv_dst	5.535043e+00	5.842851e+00

Tabella 17 Media features dopo t test per attacchi Backdoors

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo Backdoors sembrano differire in media dal traffico normale per le stesse features di Generic, tuttavia sono presenti delle differenze:
- Una durata maggiore (dur).
- Un numero minore di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).

Analysis

columns	normal	analysis
dur	1.015407e+00	1.416942e+00
spkts	2.756317e+01	5.354875e+00
dpkts	3.302794e+01	2.123272e+00
sbytes	4.092444e+03	8.396451e+02
dbytes	2.613820e+04	2.961845e+02
rate	1.958829e+04	1.478083e+05
sttl	9.489008e+01	2.100366e+02
dttl	8.184868e+01	5.873142e+01
sload	2.976812e+07	1.175204e+08
dload	1.788697e+06	2.530481e+03
sloss	4.729785e+00	4.882331e-01
dloss	1.217184e+01	4.919686e-01
sinpkt	2.344196e+03	1.536932e+02
sjit	6.491785e+03	1.139799e+04
djit	8.375593e+02	3.814796e+02
swin	1.835943e+02	5.924916e+01
stcpb	1.491325e+09	5.035461e+08
dtcpb	1.480589e+09	4.955274e+08

dwin	1.766758e+02	5.924916e+01
tcprtt	4.908785e-02	3.726418e-02
synack	2.675136e-02	1.964133e-02
ackdat	2.233650e-02	1.762286e-02
smean	1.454127e+02	9.823907e+01
dmean	2.214310e+02	3.312663e+01
trans_depth	1.001183e-01	2.080687e-01
response_body_len	3.218943e+03	3.902353e+01
ct_srv_src	5.829548e+00	6.272693e+00
ct_state_ttl	6.957204e-01	1.792305e+00
ct_dst_ltm	3.431656e+00	3.032873e+00
ct_src_dport_ltm	1.779376e+00	2.673889e+00
ct_dst_sport_ltm	1.148581e+00	2.613747e+00
ct_dst_src_ltm	3.889355e+00	5.056406e+00
ct_ftp_cmd	1.425806e-02	0.000000e+00
ct_flw_http_mthd	1.431075e-01	1.242062e+00
ct_src_ltm	3.978978e+00	4.955547e+00
ct_srv_dst	5.535043e+00	6.087785e+00

Tabella 18 Media features dopo t test per attacchi Analysis

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo

Analysis sembrano differire in media dal traffico normale per le stesse features di Generic tranne per dinpkt, tuttavia sono presenti delle differenze:
- Una durata maggiore (dur).
- Un tempo maggiore di jitter di source (sjit).
- Una maggior profondità della pipeline della connessione della transizione domanda/risposta http (trans_depth).
- Un numero minore di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).
- Un maggior numero di flussi con metodi come Get o Post in servizi http (ct_flw_http_mthd).

Shellcode

columns	normal	shellcode
dur	1.015407e+00	3.628556e-01
spkts	2.756317e+01	5.964924e+00
dpkts	3.302794e+01	3.289212e+00
sbytes	4.092444e+03	5.328352e+02
dbytes	2.613820e+04	1.463997e+02
rate	1.958829e+04	9.914558e+04
sttl	9.489008e+01	2.540000e+02
dttl	8.184868e+01	1.250827e+02
sload	2.976812e+07	1.304651e+08
dload	1.788697e+06	2.374147e+03
sloss	4.729785e+00	9.920582e-01
dloss	1.217184e+01	4.963600e-01
sinpkt	2.344196e+03	3.911035e+01

dinpkt	1.427879e+02	5.455379e+01
sjit	6.491785e+03	2.348062e+03
djit	8.375593e+02	8.842109e+01
swin	1.835943e+02	1.265718e+02
stcpb	1.491325e+09	1.077551e+09
dtcpb	1.480589e+09	1.080764e+09
dwin	1.766758e+02	1.265718e+02
tcprtt	4.908785e-02	6.177922e-02
ackdat	2.233650e-02	3.232990e-02
smean	1.454127e+02	1.227399e+02
dmean	2.214310e+02	2.218134e+01
trans_depth	1.001183e-01	0.000000e+00
response_body_len	3.218943e+03	0.000000e+00
ct_srv_src	5.829548e+00	2.310390e+00
ct_state_ttl	6.957204e-01	1.505625e+00
ct_dst_ltm	3.431656e+00	1.585705e+00
ct_src_dport_ltm	1.779376e+00	1.000000e+00
ct_dst_sport_ltm	1.148581e+00	1.000000e+00
ct_dst_src_ltm	3.889355e+00	1.353408e+00

ct_ftp_cmd	1.425806e-02	0.000000e+00
ct_flw_http_mthd	1.431075e-01	0.000000e+00
ct_src_ltm	3.978978e+00	2.342819e+00
ct_srv_dst	5.535043e+00	1.568498e+00

Tabella 19 Media features dopo t test per attacchi Shellcode

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo Shellcode sembrano differire in media dal traffico normale per le stesse features di Generic tranne per synack, tuttavia sono presenti delle differenze:
- Un valore di time to live da destination a source maggiore (dttl).
- Un maggior valore della somma di synack e ackdat (tcprtt), con però un valore in media uguale del synack nel normale traffico.
- 0 come media del della profondità della pipeline della connessione della transizione domanda/risposta http (trans_depth).
- 0 come media della dimensione degli effettivi dati trasmessi dal server del servizio http (response_body_len).
- Un numero minore di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).
- 1 di media del numero maggiore di connessioni con lo stesso indirizzo di source e la stessa porta di destination in 100 connessioni secondo l'ultima fatta (ct_dst_dport_ltm).
- 1 di media del numero maggiore di connessioni con lo stesso indirizzo di destination e la stessa porta di source in 100 connessioni secondo l'ultima fatta (ct_dst_sport_ltm).
- Un numero minore di connessioni con gli stessi indirizzi di source e destination in 100 connessioni secondo l'ultima fatta (ct_dst_src_ltm).
- 0 di media del numero di flussi con metodi come Get o Post in servizi http (ct_flw_http_mthd).
- Un minor numero di connessioni con lo stesso indirizzo di source in 100 connessioni secondo l'ultima fatta (ct_src_ltm).
- Un numero minore di connessioni con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_dst).

Worms

columns	normal	worms
spkts	2.756317e+01	1.821839e+01
dpkts	3.302794e+01	6.293103e+01
dbytes	2.613820e+04	7.696547e+04
sttl	9.489008e+01	2.495862e+02
dttl	8.184868e+01	2.215862e+02
sload	2.976812e+07	7.269864e+07
dload	1.788697e+06	1.327594e+05
sloss	4.729785e+00	2.011494e+00
dloss	1.217184e+01	2.947701e+01
sinpkt	2.344196e+03	7.188448e+01
swin	1.835943e+02	2.242241e+02
stcpb	1.491325e+09	2.007980e+09
dtcpb	1.480589e+09	1.808773e+09
dwin	1.766758e+02	2.242241e+02
tcprtt	4.908785e-02	1.213610e-01
synack	2.675136e-02	5.938307e-02
ackdat	2.233650e-02	6.197793e-02
smean	1.454127e+02	1.832644e+02

trans_depth	1.001183e-01	7.471264e-01
response_body_len	3.218943e+03	3.337710e+04
ct_srv_src	5.829548e+00	1.528736e+00
ct_state_ttl	6.957204e-01	1.109195e+00
ct_dst_ltm	3.431656e+00	1.425287e+00
ct_src_dport_ltm	1.779376e+00	1.097701e+00
ct_dst_src_ltm	3.889355e+00	1.224138e+00
ct_flw_http_mthd	1.431075e-01	8.965517e-01
ct_src_ltm	3.978978e+00	2.614943e+00
ct_srv_dst	5.535043e+00	1.339080e+00

Tabella 20 Media features dopo t test per attacchi Worms

Selezionando le colonne del Dataframe originario, secondo il criterio del p-value minore del 0.05, come per l'analisi precedente gli attacchi di tipo Worms sembrano differire in media per numerosi aspetti dal traffico normale, più nello specifico nell'ordine rappresentato nella tabella:
- Un numero minore di pacchetti da source a destination (spkts).
- Un numero maggiore di pacchetti da destination a source (dpkts).
- Un numero maggiore di bytes da destination a source (dbytes).
- Un valore di time to live da source a destination maggiore (sttl).
- Un valore di time to live da destination a source maggiore (dttl).
- Un numero maggiore di bits al secondo di source (sload).
- Un numero minore di bits al secondo di destination (dload).
- Un numero minore di pacchetti ritrasmessi o cancellati di source (sloss).
- Un numero maggiore di pacchetti ritrasmessi o cancellati di destination (dloss).
- Un tempo minore di arrivo di pacchetti intermedi di source (sinpkt).
- Un maggior valore di TCP window advertisement di source (swin).
- Un numero maggiore di TCP base sequence di source (stcpb).
- Un numero maggiore di TCP base sequence di destination (dtcpb).
- Un maggior valore di TCP window advertisement di destination (dwin).

- Un maggior valore della somma di synack e ackdat (tcprtt).
- Una maggior media della dimensione del pacchetto ritrasmesso da source (smean).
- Una maggior profondità della pipeline della connessione della transizione domanda/risposta http (trans_depth).
- Un maggior dimensione degli effettivi dati trasmessi dal server del servizio http (response_body_len).
- Un numero minore di connessione con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_src).
- Un numero maggiore di per ogni stato secondo uno specifico range di valori per il time to live da source a destination (ct_state_ttl).
- Un numero minor di connessioni verso lo stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_dst_ltm).
- Un numero minore di connessioni con lo stesso indirizzo di source e la stessa porta di destination in 100 connessioni secondo l'ultima fatta (ct_dst_dport_ltm).
- Un numero minore di connessioni con gli stessi indirizzi di source e destination in 100 connessioni secondo l'ultima fatta (ct_dst_src_ltm).
- Un maggior numero di flussi con metodi come Get o Post in servizi http (ct_flw_http_mthd).
- Un minor numero di connessioni con lo stesso indirizzo di source in 100 connessioni secondo l'ultima fatta (ct_src_ltm).
- Un numero minore di connessioni con lo stesso servizio e stesso indirizzo di destination in 100 connessioni secondo l'ultima fatta (ct_srv_dst).

Conclusioni Analisi iniziale del Dataset

Analizzando complessivamente il Dataset si è potuto notare come effettivamente sono presenti differenze tra i record classificati come traffico normale e quelli classificati come attacchi; tuttavia, queste differenze sono presenti in varie features, che sono anche correlate tra loro, e in diversi modi:
- Riportando valori in media maggiori o minori rispetto al normale traffico.
- Riportando uno stesso valore i numerosi record quando normalmente non è così.
- Non riportando uno stesso valore che, invece, è presente in numerosi record classificati come traffico normale.

Si è potuto notare come le features spesso differenti sono praticamente tutte dipendentemente dal tipo di attacco, e le differenze possono essere evidenti come in molte features nel caso degli attacchi di tipo Generic o molto più sottili come nel caso degli attacchi di tipo Exploits dove mediamente delle features differivano in maniera significativa ma comunque meno rispetto ad altri attacchi. Queste differenze così diverse tra loro seppur nelle stesse features in base al tipo di

attacco potrebbe rendere il modello molto più inefficace di quel che si possa pensare vedendo i risultati da questa prima analisi, soprattutto perché gli algoritmi basati sul Decision Tree cercano la previsione attraverso numerose divisioni fatte da più condizioni, rendendo difficile cercare la predizione corretta se i tipi degli attacchi sono così differenti tra loro e per lo stesso motivo l'algoritmo Knn (K-nearest neighbors) potrebbe trovare difficoltà nella ricerca di record simili a quello che deve predire.
La trasformazione delle features categoriali in numeriche potrebbe portare a delle altre imprecisioni nella predizione di un modello visto che l'encoding in valori numerici potrebbe creare degli ordinamenti all'interno della feature che non dovrebbero esistere; tuttavia, l'opzione di realizzare un One Hot Encoding, dove si creano colonne per rappresentare ogni possibile valore assunto da una features, era reso impossibile dalla quantità di valori diversi assunti dalla variabile.
In più anche il fatto delle features molto correlate tra loro potrebbe essere svantaggioso anche togliendo le features meno importanti per gli algoritmi analizzando i primi risultati, questo perché la creazione dell'intero algoritmo potrebbe essere traviata da queste features così legate tra loro, tuttavia, si è preso la decisione di provare almeno inizialmente ad utilizzare quante più features possibili per studiarne anche la selezione successiva dell'algoritmo stesso.

Training degli algoritmi

Successivamente all'operazione di preparazione dei dati e di un'analisi complessiva si sono divisi nuovamente i due Dataset Train e Test, utilizzando la variabile tr_ts precedentemente creata, per poter allenare così il modello sul Dataset Train su una parte delle features non target, tutte tranne le due possibili variabili target e escludendo realID che identifica singolarmente il record, cercando di predire quelle target, cioè attack o attack_cat.
Come accennato precedentemente ogni algoritmo verrà applicato in due modi: uno per predire la variabile attack e l'altro per predire la variabile attack_cat.
Ogni allenamento dei diversi modelli ha bisogno di un tuning di una variabile accennata nella descrizione dell'algoritmo, per farlo si è utilizzato il metodo della cross validation con una divisione in 5, che consiste in un modo utile a non creare modelli soggetti troppo all'overfitting, il modello predice bene sui record dove si allena ma male in altri casi, e per cercare il valore dell'iper-parametro complessivamente più corretto. Più precisamente la cross validation non fa altro che dividere in un numero di divisioni prestabilito un Dataset e fa creare e allenare il modello su queste divisioni ogni volta

differenti, successivamente guardando la media dei risultati si prendono i valori degli iper-parametri dei modelli migliori. In questo caso si è utilizzato lo scoring accuracy per decretare quale sia il migliore tra i modelli con la cross validation, questo in caso di predizione della variabile attack diventa jaccard score essendo una variabile binaria. Gli score sono realizzati secondo le seguenti formule:

$$Accuracy = \frac{Numero\ corretto\ di\ predizioni}{Numero\ totale\ di\ predizioni}$$

$$Jaccard = \frac{TP + TN}{TP + TN + FP + FN}$$

(TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative)

Successivamente sono presentati i vari algoritmi e il loro score per ogni possibile iper-parametro provato in tutti e due i casi, predizione di attack e predizione di attack_cat.

Decision Tree

Per l'algoritmo Decision Tree l'iper-parametro che indica il numero massimo di foglie ha potuto assumere i seguenti possibili valori 5, 10, 50, 100, 500, 1000 per rientrare in tempi accettabili vista la strumentazione. I vari cross validation sui possibili valori di questo parametro con le due applicazioni possibili hanno presentato le seguenti medie di risultati:

Decision Tree predizione su attack

valore iper-parametro	accuracy
5	0.872979017138996
10	0.8832901392530191
50	0.9103802164556078
100	0.9132545882927412
500	0.9132831483909658
1000	0.912621601307085

Tabella 21 Iper-parametro per Decision Tree su attack

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 500.

Decision Tree predizione su attack_cat

valore iper-parametro	accuracy
5	0.6497278501513888
10	0.7021173506685185
50	0.7584304008753971
100	0.7738632240607195
500	0.7837582468674095
1000	0.7819959494671919

Tabella 22 Iper-parametro per Decision Tree su attack_cat

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 500.

K-nearest neighbors

Per l'algoritmo K-nearest neighbors l'iper-parametro che indica la quantità di elementi vicini presi in considerazione ha potuto assumere i seguenti possibili valori 2000, 50, 10 per rientrare in tempi accettabili vista la strumentazione. I vari cross validation sui possibili valori di questo parametro con le due applicazioni possibili hanno presentato le seguenti medie di risultati:

K-nearest neighbors predizione su attack

valore iper-parametro	accuracy
2000	0.8034629444353447
50	0.8835809576703
10	0.8681026095101672

Tabella 23 Iper-parametro per K-nearest neighbors su attack

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 50.

K-nearest neighbors predizione su attack_cat

valore iper-parametro	accuracy
2000	0.5908598045087168
50	0.6569710210986133
10	0.6415382566220387

Tabella 24 Iper-parametro per K-nearest neighbors su attack_cat

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 50.

Random Forest

Per l'algoritmo Random Forest l'iper-parametro che indica il numero di Decision Tree utilizzati ha potuto assumere i seguenti possibili valori 10, 50, 100 per rientrare in tempi accettabili vista la strumentazione. I vari cross validation sui possibili valori di questo parametro con le due applicazioni possibili hanno presentato le seguenti medie di risultati:

Random Forest predizione su attack

valore iper-parametro	accuracy
10	0.9094107301175027
50	0.909981006402789
100	0.9104885882505165

Tabella 25 Iper-parametro per Random Forest su attack

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 100.

Random Forest predizione su attack_cat

valore iper-parametro	accuracy
10	0.7620634040620657
50	0.7669852109541755
100	0.7674756841620677

Tabella 26 Iper-parametro per Random Forest su attack_cat

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 100.

AdaBoost

Per l'algoritmo AdaBoost si è utilizzato come Decision Tree di base il migliore per ogni tipo di predizione (su attack e attack_cat) ottenuto dall'applicazione precedente dell'algoritmo e l'iper-parametro, che indica il numero di Decision Tree creati da quello di base, ha potuto assumere i seguenti possibili valori 5, 10, 50 per rientrare in tempi accettabili vista la strumentazione. I vari cross validation sui possibili valori di questo parametro con le due applicazioni possibili hanno presentato le seguenti medie di risultati:

AdaBoost predizione su attack

valore iper-parametro	accuracy

5	0.9119714697407811
10	0.908794787208322
50	0.9082472569036877

Tabella 27 Iper-parametro per AdaBoost su attack

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 5.

AdaBoost predizione su attack_cat

valore iper-parametro	accuracy
5	0.6874257385974366
10	0.7316541017640782
50	0.7279698193321714

Tabella 28 Iper-parametro per AdaBoost su attack_cat

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 10.

Bagging

Per l'algoritmo Bagging si è utilizzato come Decision Tree di base il migliore per ogni tipo di predizione (su attack e attack_cat) ottenuto dall'applicazione precedente dell'algoritmo e l'iper-parametro, che indica il numero di Decision Tree creati da quello di base, ha potuto assumere i seguenti possibili valori 10, 100, 150 per rientrare in tempi accettabili vista la strumentazione. I vari cross validation sui possibili valori di questo parametro con le due applicazioni possibili hanno presentato le seguenti medie di risultati:

Bagging predizione su attack

valore iper-parametro	accuracy
10	0.9130607209486925
100	0.9118345271141928
150	0.9111786606302061

Tabella 29 Iper-parametro per Bagging su attack

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 10.

Bagging predizione su attack_cat

valore iper-parametro	accuracy

10	0.7859368398670895
100	0.7850186178124364
150	0.7843684584367255

Tabella 30 Iper-parametro per Bagging su attack_cat

Il risultato migliore tramite questo metodo di scoring e ottenuto con l'iper-parametro con il valore di 10.

Analisi sulla ricerca dell'iper-parametro

Solitamente per la predizione di attack_cat sono necessari valori più alti dell'iper-parametro essendo una predizione più complicata da fare, mentre per la predizione di attack è il contrario, questo perché, grazie alla cross validation, i modelli che soffrono di overfitting sono penalizzati. Tuttavia, analizzando questi risultati questo comportamento è dato solamente da AdaBoost, mentre per il resto degli algoritmi utilizzano lo stesso valore dell'iper-parametro e solamente Random Forest utilizza quello più elevato di quelli disponibili, Knn (K-nearest neighbors) è il contrario essendo che il valore più basso indica più precisione, mostrando che negli altri algoritmi si è soggetti a overfitting.

Features importance

Per ogni algoritmo, escludendo Knn (K-nearest neighbors) e quello di Bagging, si sono creati grafici o liste per visualizzare l'importanza delle features secondo l'algoritmo specifico. Per l'algoritmo Random Forest, precedentemente alla produzione di tale lista o grafico, si è provveduto ad eliminare, facendo un ulteriore test tramite cross validation, le features che portavano ad un errore maggiore. Per l'algoritmo Decision Tree si è anche presentato visibilmente la struttura dell'albero creato.

Decision Tree

Decision Tree predizione su attack

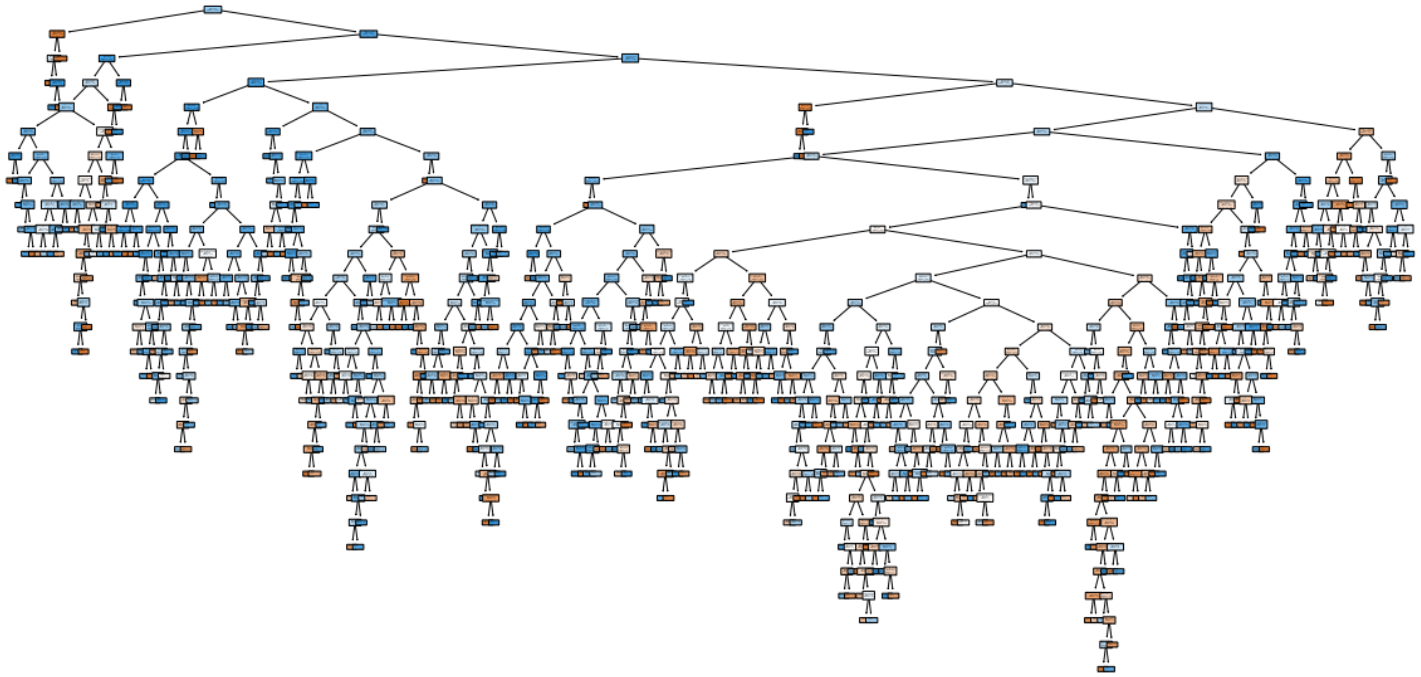

Figura 15 Albero per predizione su attack

La precedente figura rappresenta il Decision Tree creato con l'iper-parametro migliore secondo l'analisi precedente. Quest'albero ha come features più importanti quelle rappresentate dal seguente bar chart.

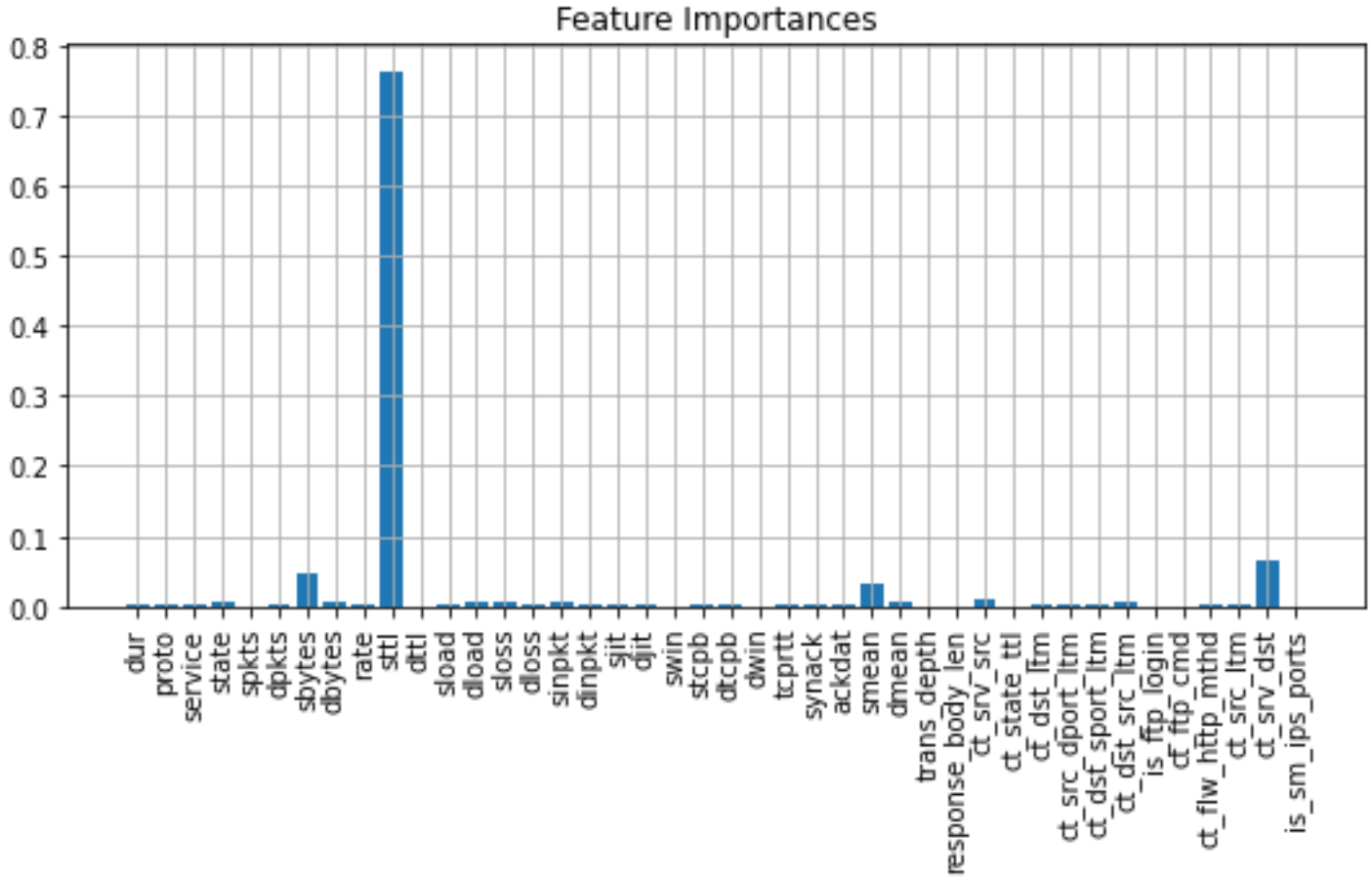

Figura 16 Importanza delle features per Decision Tree su attack

Questo bar chart ha sull'asse delle x tutte le features utilizzabili per predire e l'altezza di ogni barra indica l'importanza di quest'ultima in una scala che va da 0 a 1. La lista delle features in ordine dalla più importante a quella meno importante è la seguente:
'sttl', 'ct_src_dport_ltm', 'sbytes', 'tcprtt', 'smean', 'state', 'dload', 'ct_srv_src', 'synack', 'dloss', 'sload', 'dbytes', 'stcpb', 'dtcpb', 'dttl', 'sjit', 'sinpkt', 'service', 'dwin', 'ct_dst_ltm', 'trans_depth', 'dur', 'dmean', 'dinpkt', 'djit', 'swin', 'response_body_len', 'dpkts', 'sloss', 'rate', 'ct_state_ttl', 'proto', 'ackdat', 'spkts'.
Se una features non è presente dalla lista allora non è utilizzata dall'algoritmo.

Decision Tree predizione su attack_cat

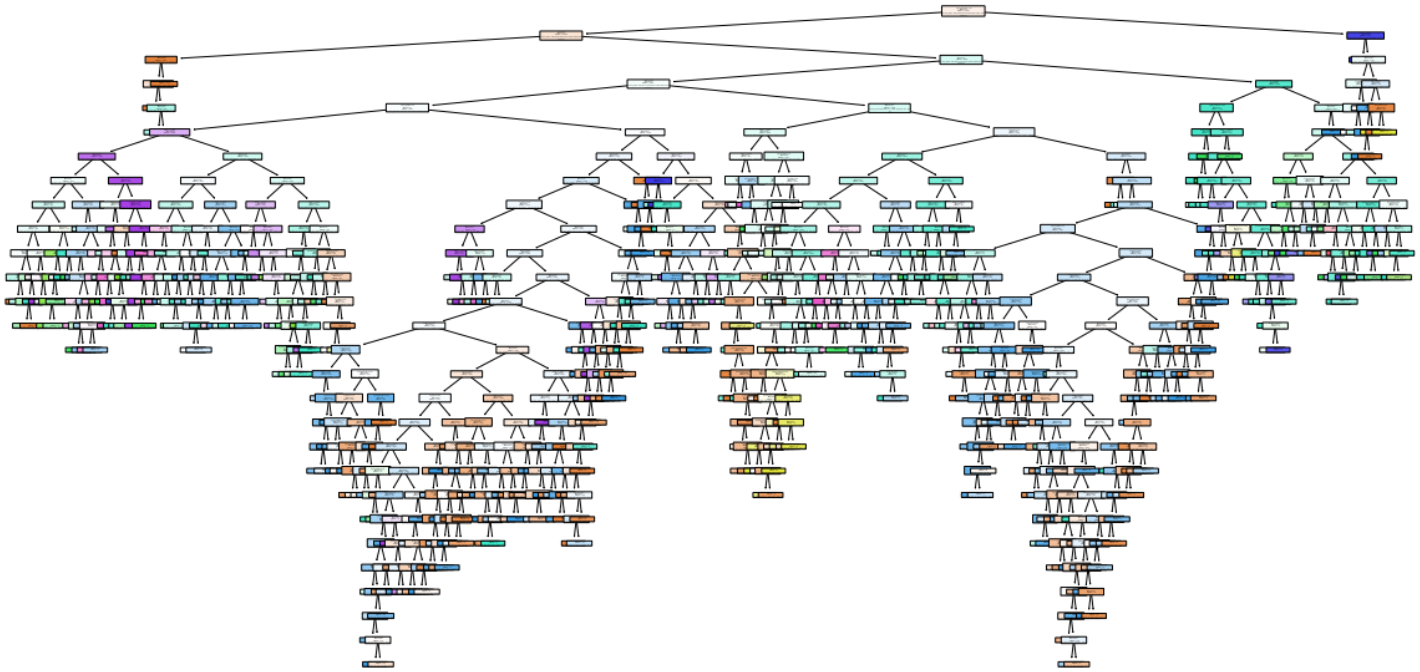

Figura 17 Albero per predizione su attack_cat

La precedente figura rappresenta il Decision Tree creato con l'iper-parametro migliore secondo l'analisi precedente. Quest'albero ha come features più importanti quelle rappresentate dal seguente bar chart.

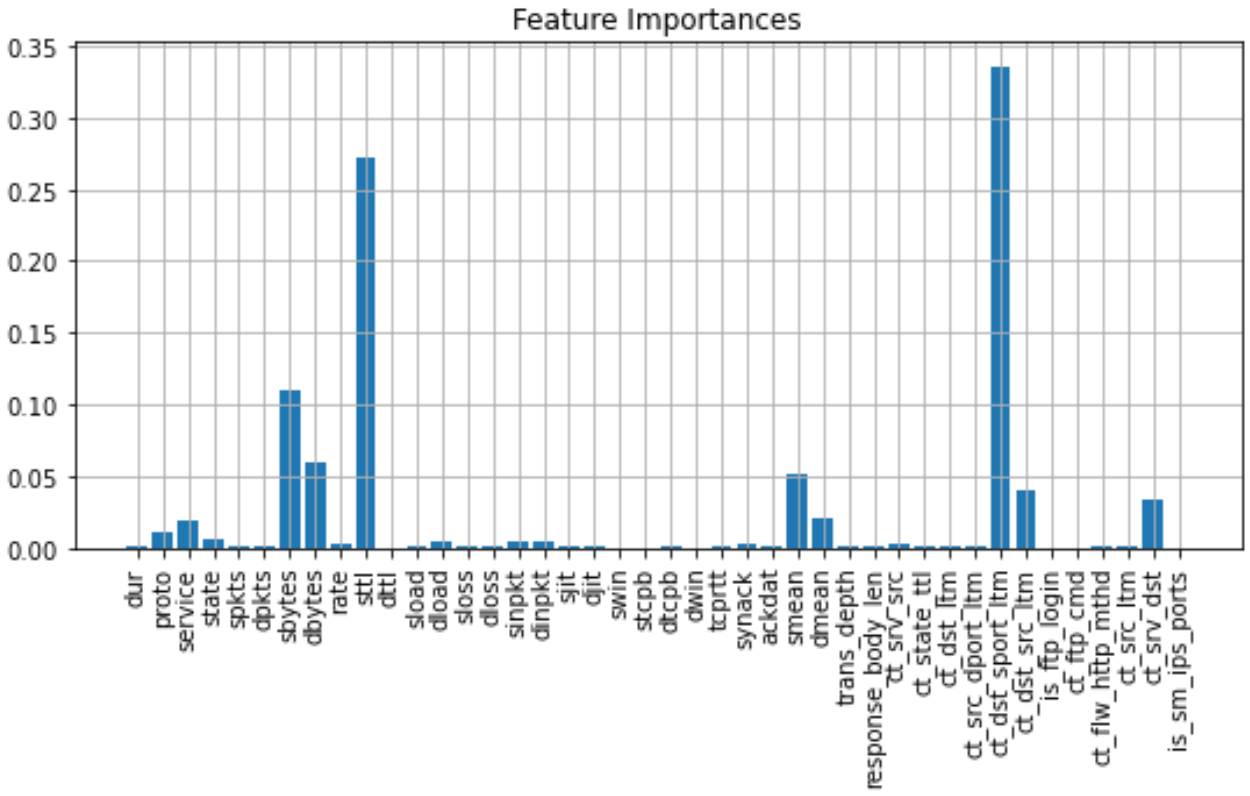

Figura 18 Importanza delle features per Decision Tree su attack_cat

Università
Ca'Foscari
Venezia

Questo bar chart ha sull'asse delle x tutte le features utilizzabili per predire e l'altezza di ogni barra indica l'importanza di quest'ultima in una scala che va da 0 a 1. La lista delle features in ordine dalla più importante a quella meno importante è la seguente:
'ct_dst_sport_ltm', 'sttl', 'sbytes', 'dbytes', 'smean', 'ct_dst_src_ltm', 'ct_srv_dst', 'dmean', 'service', 'proto', 'state', 'dload', 'dinpkt', 'sinpkt', 'ct_srv_src', 'rate', 'synack', 'tcprtt', 'ct_src_ltm', 'spkts', 'sjit', 'sloss', 'dpkts', 'trans_depth', 'ct_src_dport_ltm', 'ct_dst_ltm', 'ct_flw_http_mthd', 'sload', 'dloss', 'response_body_len', 'dur', 'djit', 'ct_state_ttl', 'ackdat', 'dtcpb', 'stcpb'.
Se una features non è presente dalla lista allora non è utilizzata dall'algoritmo.

Random Forest

Random Forest predizione su attack

La lista delle features in ordine dalla più importante a quella meno importante prima dell'eliminazione di alcune features è la seguente:
'sttl', 'ct_state_ttl', 'dload', 'rate', 'sload', 'dttl', 'dur', 'dinpkt', 'ct_srv_dst', 'sbytes', 'dbytes', 'smean', 'dmean', 'ackdat', 'ct_dst_src_ltm', 'synack', 'tcprtt', 'sinpkt', 'ct_srv_src', 'dpkts', 'ct_dst_ltm', 'djit', 'sjit', 'stcpb', 'ct_dst_sport_ltm', 'ct_src_ltm', 'spkts', 'dtcpb', 'sloss', 'ct_src_dport_ltm', 'proto', 'swin', 'state', 'service', 'is_sm_ips_ports', 'dloss', 'response_body_len', 'ct_flw_http_mthd', 'trans_depth', 'dwin', 'ct_ftp_cmd', 'is_ftp_login'.

Per eliminare le features si è utilizzato la cross validation con lo stesso scoring (accuracy) partendo con un modello che utilizza una sola features, la più importante della lista precedente, mostrando poi il risultato ottenuto per poi passare ad utilizzarne due selezionandole in ordine di importanza secondo la lista precedente. Il seguente grafico mostra i vari risultati partendo dal primo che utilizza una sola features fino all'ultimo che le utilizza tutte.

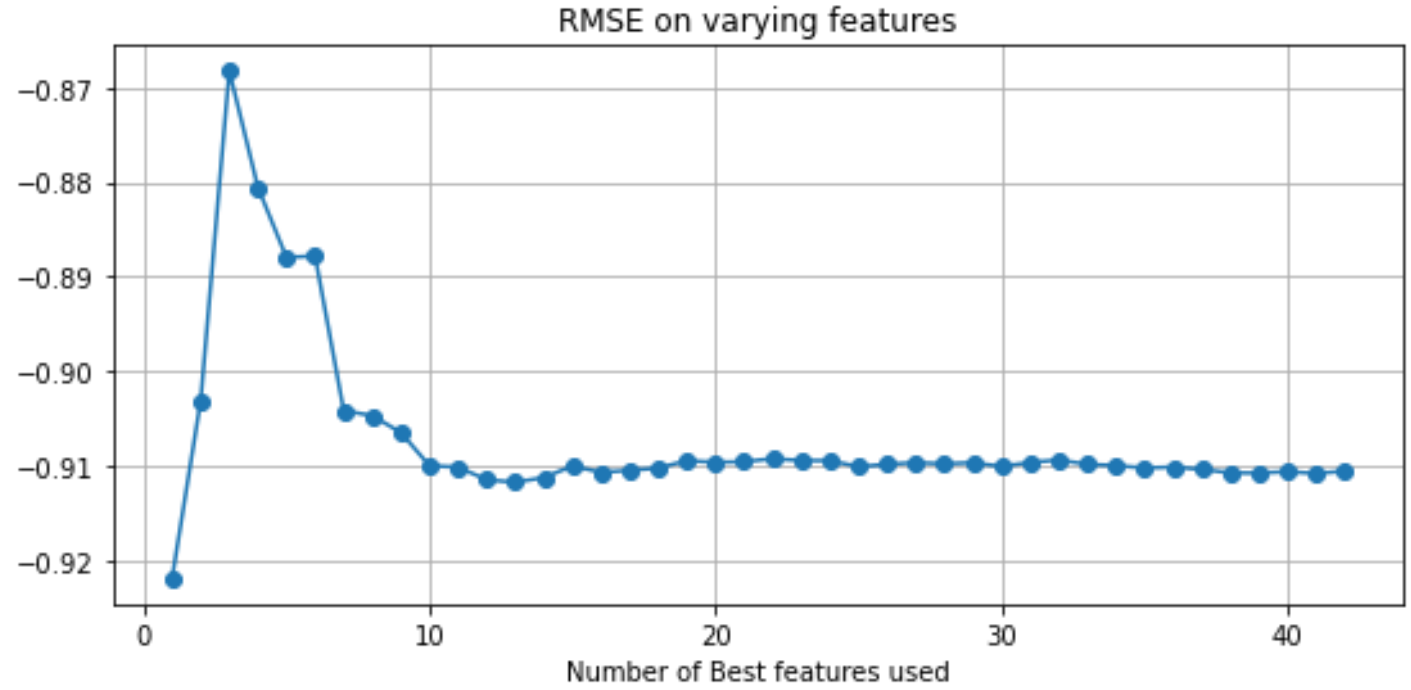

Figura 19 Selezione features per Random Forest su attack

Il risultato migliore è stato ottenuto dal modello creato con 1 features, lo score è visibile a destra con segno negativo, in più si può notare come aggiungendo la seconda e la terza features si ha un peggioramento significativo sullo scoring per poi stabilizzarsi su una media di 0.91 verso l'utilizzo di 10 features in poi, questo fenomeno di peggioramento significativo all'aggiunta di features non è solito in altri casi, tuttavia sembra che l'utilizzo di tutte le condizioni su un'unica variabile produca risultati migliori rispetto ad averne di più, forse per la grande importanza che ha quest'ultima sulla predizione. La media degli score è 0.9105056998176579 mentre lo score migliore è 0.9218547589293102.

L'unica features utilizzata in conclusione è sttl.

L'eliminazione delle features ha portato un miglioramento significativo.

Random Forest predizione su attack_cat

La lista delle features in ordine dalla più importante a quella meno importante prima dell'eliminazione di alcune features è la seguente: 'sbytes', 'sttl', 'ct_dst_sport_ltm', 'smean', 'ct_srv_dst', 'ct_state_ttl', 'ct_src_dport_ltm', 'ct_dst_src_ltm', 'ct_srv_src', 'sload', 'rate', 'service', 'dload', 'dbytes', 'dmean', 'ct_dst_ltm', 'dur', 'ackdat', 'ct_src_ltm', 'synack', 'tcprtt', 'dinpkt', 'sinpkt', 'proto', 'dttl', 'dpkts', 'sjit', 'djit', 'stcpb', 'dloss', 'sloss', 'dtcpb', 'spkts', 'state', 'response_body_len', 'ct_flw_http_mthd',

'is_sm_ips_ports', 'trans_depth', 'swin', 'dwin', 'ct_ftp_cmd', 'is_ftp_login'.

Per eliminare le features si è utilizzato la cross validation con lo stesso scoring (accuracy) partendo con un modello che utilizza una sola features, la più importante della lista precedente, mostrando poi il risultato ottenuto per poi passare ad utilizzarne due selezionandole in ordine di importanza secondo la lista precedente. Il seguente grafico mostra i vari risultati partendo dal primo che utilizza una sola features fino all'ultimo che le utilizza tutte.

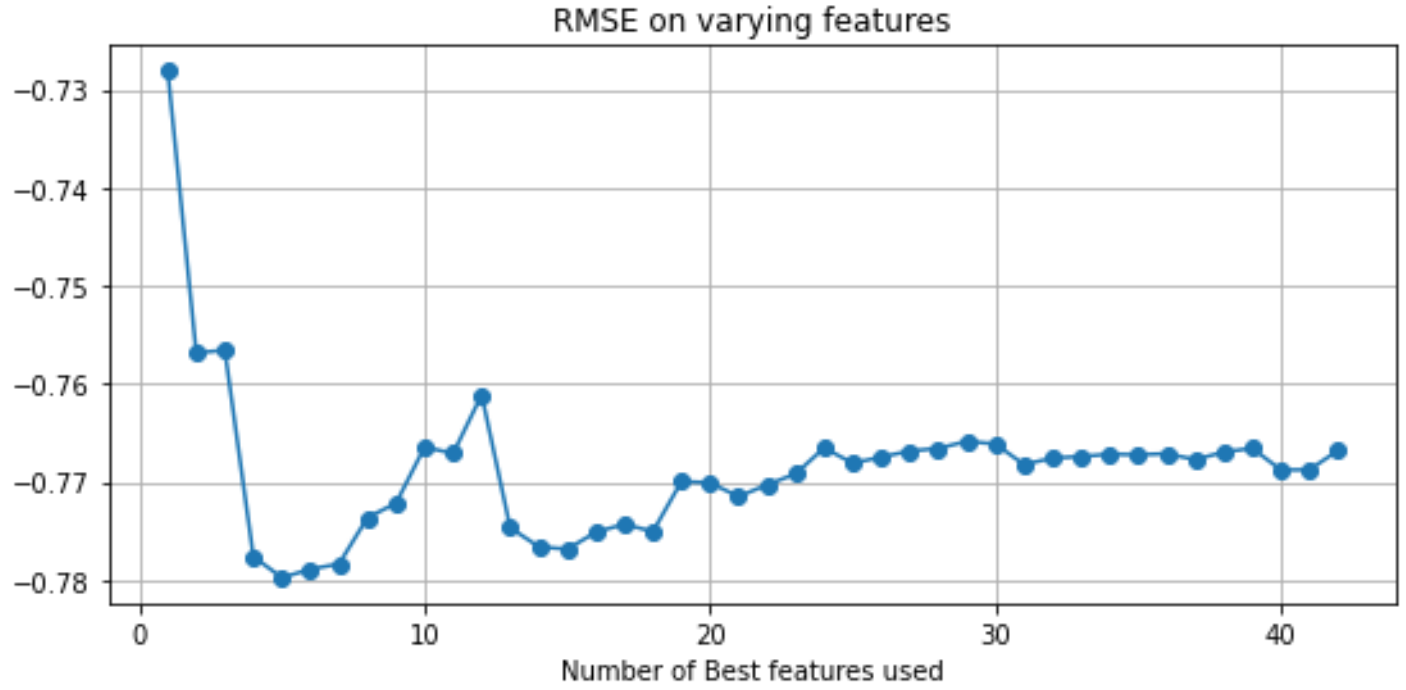

Figura 20 Selezione features per Random Forest su attack_cat

Il risultato migliore è stato ottenuto dal modello creato con 5 features, lo scor e è visibile a destra con segno negativo, in più si può notare come aggiunge ndo alcune features generalmente si ha un peggioramento sullo scoring per poi stabilizzarsi su una media di 0.766 verso l'utilizzo di 20 features in poi. Q uesto comportamento è molto più comune rispetto al precedente anche se c omunque con poche features sembra avere uno scoring più alto. La media d egli score è 0.7668198139376343 mentre lo score migliore è 0.7797831185 34071.

Le features utilizzate in conclusione in ordine dalla più importante a quella meno importante è la seguente e con il seguente bar chart a descriverle maggiormente:
'sbytes', 'sttl', 'ct_dst_sport_ltm', 'smean', 'ct_srv_dst'.

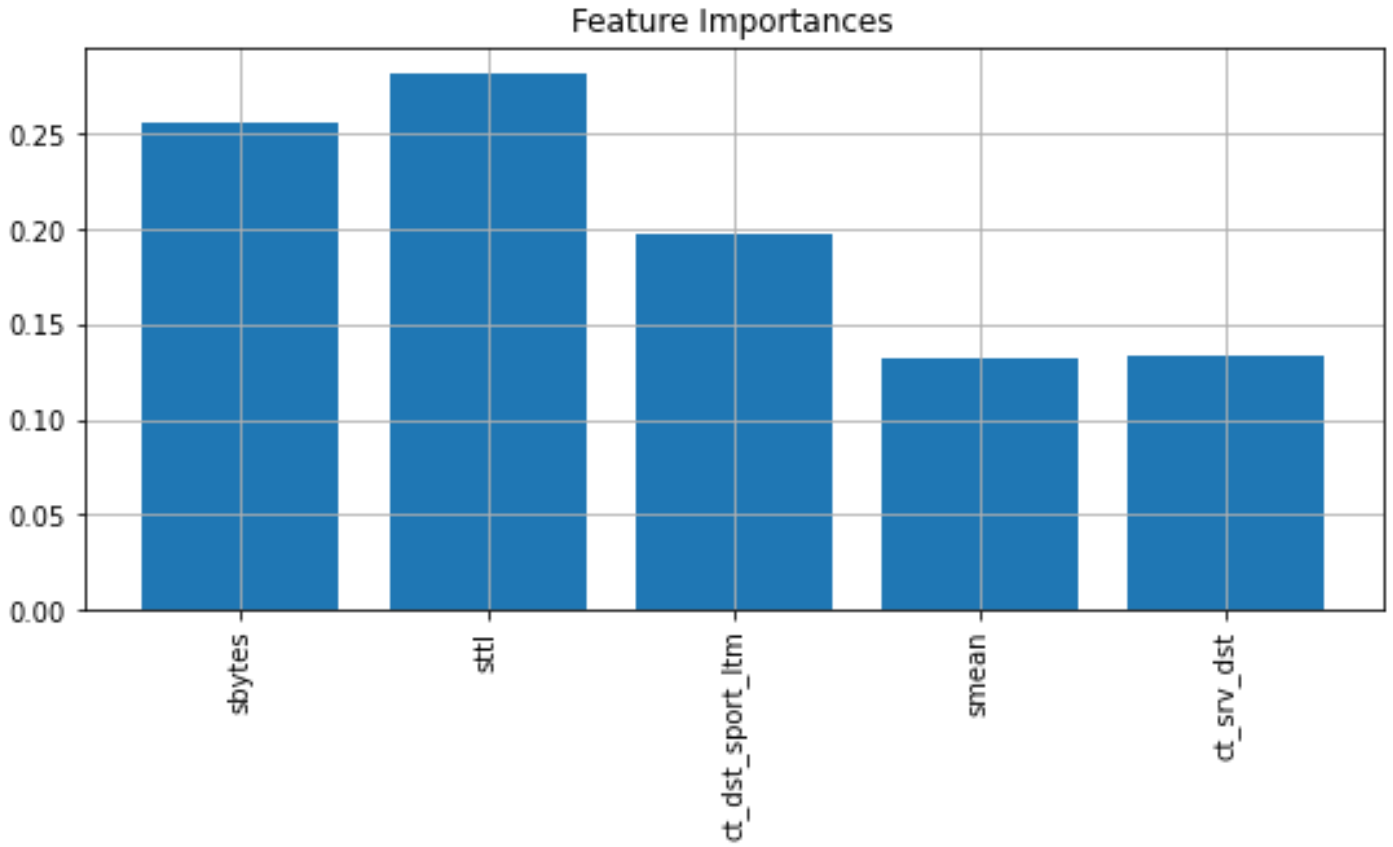

Figura 21 Importanza delle features per Random Forest su attack_cat

L'eliminazione delle features ha portato un miglioramento significativo.

AdaBoost

AdaBoost predizione su attack

Le features utilizzate in conclusione in ordine dalla più importante a quella meno importante è la seguente e con il seguente bar chart a descriverle maggiormente:
'sttl', 'sbytes', 'tcprtt', 'djit', 'swin', 'dtcpb', 'dttl', 'stcpb', 'dmean', 'dwin', 'sjit', 'is_ftp_login', 'ct_dst_ltm', 'sload', 'dinpkt', 'ct_state_ttl', 'dloss', 'dur', 'ct_dst_src_ltm', 'sinpkt', 'dbytes', 'rate', 'response_body_len', 'proto', 'synack', 'ct_srv_src', 'service', 'smean', 'trans_depth', 'dload', 'spkts', 'state', 'sloss', 'ct_dst_sport_ltm', 'dpkts', 'ct_src_dport_ltm', 'ackdat'.

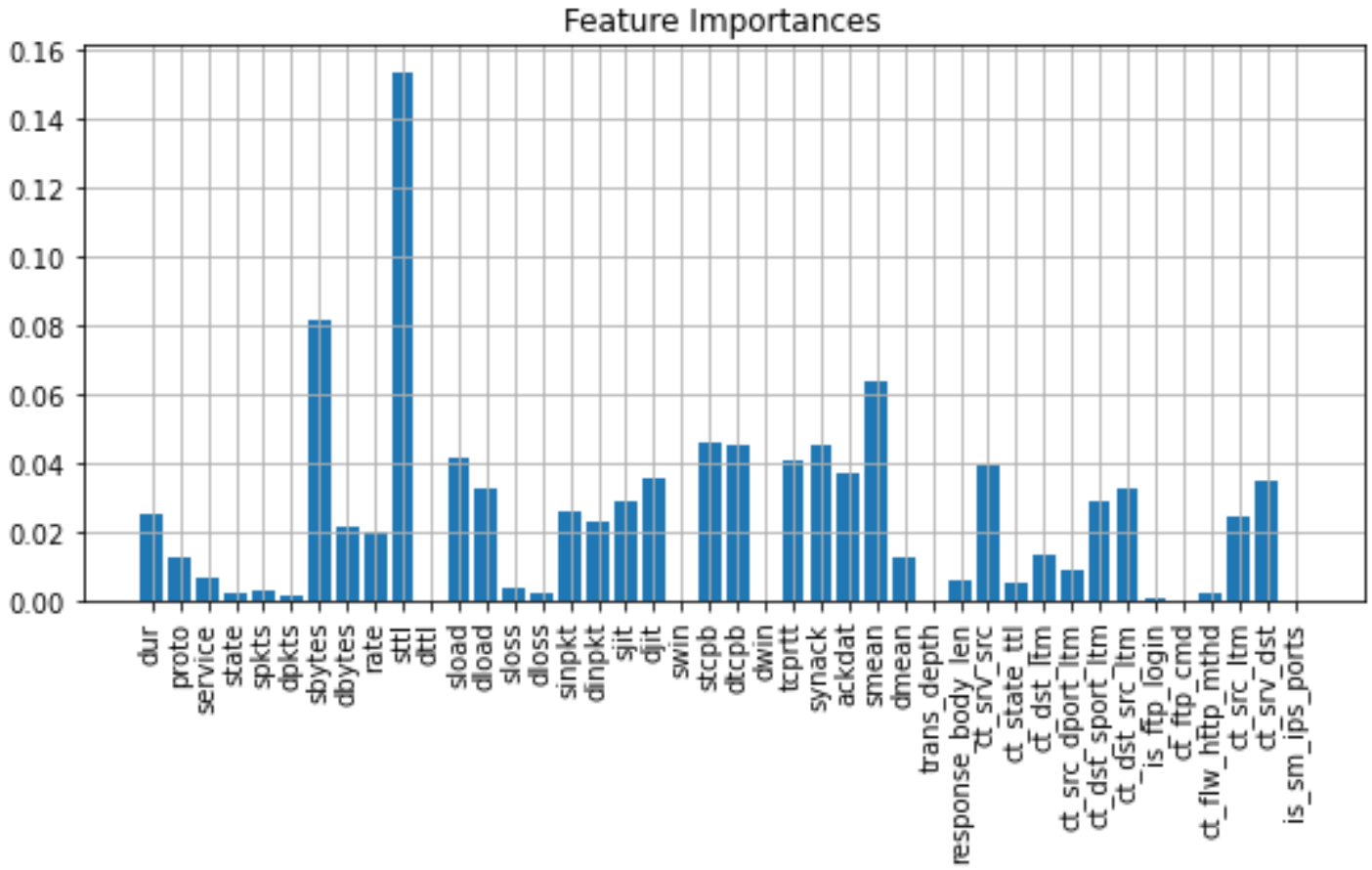

Figura 22 Importanza delle features per AdaBoost su attack

AdaBoost predizione su attack_cat

Le features utilizzate in conclusione in ordine dalla più importante a quella meno importante è la seguente e con il seguente bar chart a descriverle maggiormente:
'tcprtt', 'sbytes', 'ct_ftp_cmd', 'ct_dst_ltm', 'swin', 'service', 'ct_state_ttl', 'sttl', 'sload', 'proto', 'dloss', 'dmean', 'dttl', 'response_body_len', 'dtcpb', 'rate', 'dbytes', 'stcpb', 'synack', 'is_ftp_login', 'trans_depth', 'djit', 'dinpkt', 'smean', 'dur', 'dwin', 'ct_srv_src', 'sinpkt', 'ct_dst_src_ltm', 'sjit', 'spkts', 'state', 'sloss', 'ackdat', 'dpkts', 'dload', 'ct_dst_sport_ltm', 'ct_flw_http_mthd', 'ct_src_dport_ltm'.

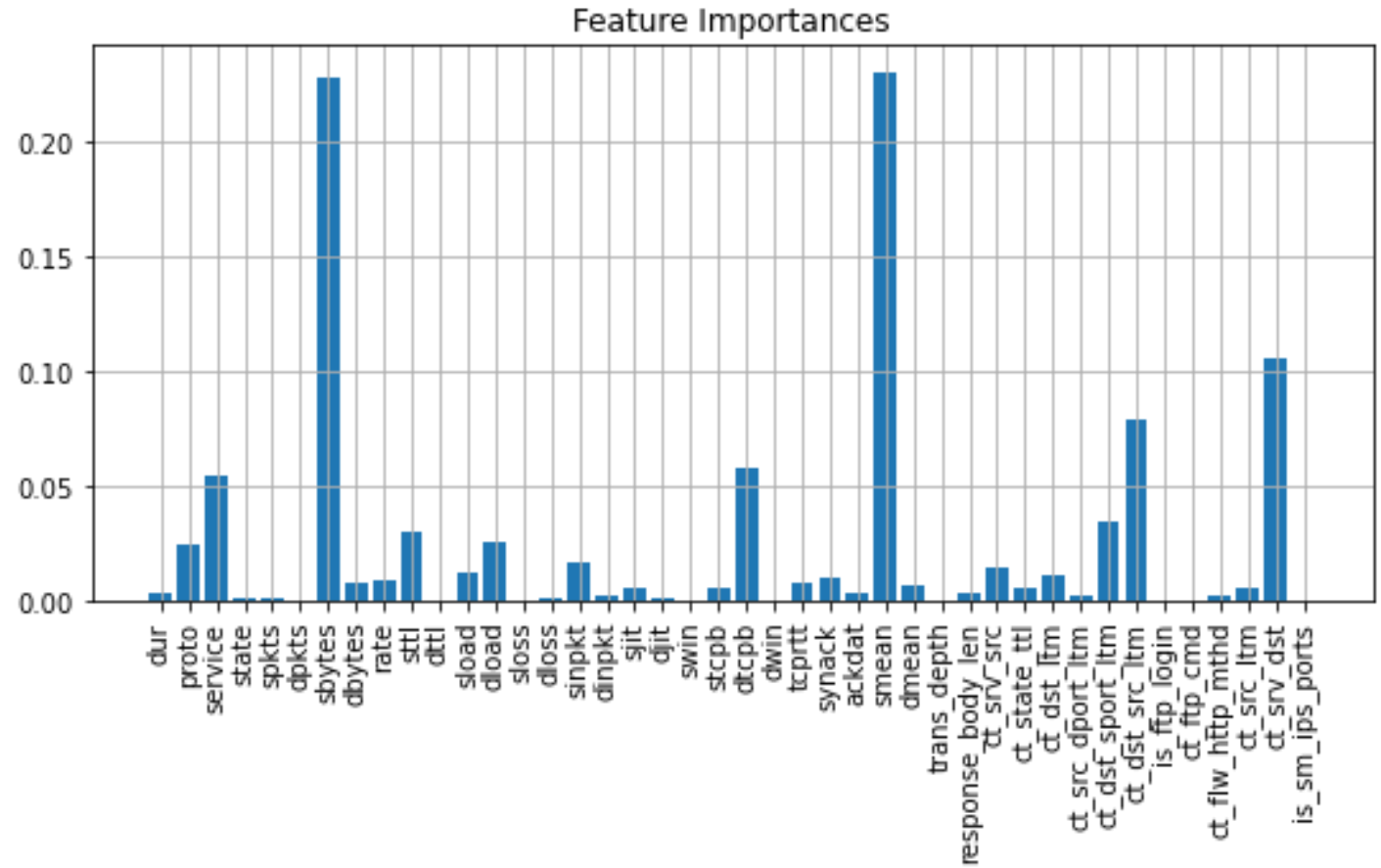

Figura 23 Importanza delle features per AdaBoost su attack_cat

Conclusione dell'analisi sulla features importance

Analizzando le features più importanti per ognuno di questi algoritmi si può notare come la features più utilizzata guardando nel complesso sia sttl, questa osservazione rispecchia l'analisi fatta sulle varie features inizialmente.
Altre features molto utilizzate dagli algoritmi che predicono attack e attack_cat sono: ct_srv_dst, sbytes, smean.
In conclusione alla fase di training anche pel l'algoritmo Random Forest si può già fare una classifica in base allo scoring dei vari algoritmi che è scaturito tramite la cross_validation, per la predizione di attack la lista dallo scoring più alto a quello più basso è:
Random Forest (0.9218547589293102), Decision Tree (0.9132831483909658), Bagging (0.9130607209486925), AdaBoost (0.9119714697407811), K-nearest neighbors (0.8835809576703).
Mentre la lista degli algoritmi per la predizione di attack_cat dallo scoring più alto a quello più basso è:
Bagging (0.7859368398670895), Decision Tree (0.7837582468674095), Random Forest (0.779783118534071), AdaBoost (0.7316541017640782), K-nearest neighbors (0.6569710210986133).

Complessivamente considerando entrambe le predizioni l'algoritmo migliore per ora sembra essere Decision Tree che tecnicamente è l'algoritmo più semplice tra i 5, questo risultato è peculiare, ma potrebbe essere dovuto alle features presentate dove solo alcune sono veramente significative e molte sono anche correlate.
L'applicare Random Forest su un'unica features per predire attack ha portato il risultato migliore rispetto a tutti, cosa che suggerisce ancora una volta l'importanza di questa variabile sulla predizione, che tuttavia potrebbe non ottenere risultati così buoni applicando l'algoritmo sul Dataset Test.
Un altro risultato interessante è l'andamento dell'algoritmo di Bagging sulla predizione di attack_cat dove ha ottenuto risultati migliori rispetto agli altri, questo potrebbe essere dovuto alla partenza già buona di base dell'algoritmo Decision Tree e all'iper-parametro basso che ha permesso di migliorare anziché diminuire il risultato in questo caso, al contrario dell'AdaBoost dove sembra che in entrambe le predizione l'applicare di altri alberi sugli errori fatti dai precedenti abbia portato ad un peggioramento rispetto al modello preso come base.

Risultati

Per analizzare i vari risultati degli algoritmi si sono prodotti vari bar chart e infine si sono presentati vari risultati attraverso l'utilizzo di diversi tipi di scoring.

Risultati sulle predizioni corrette di attack

Si è creato un bar chart che presenta il numero di occorrenze predette in maniera corretta per ognuno dei 5 algoritmi allenati per le predizioni della variabile attack, creando così 6 barre per ogni tipologia d'attacco dove una di queste non rappresenta un algoritmo ma il numero totale di attacchi di quel tipo, mostrando visibilmente la quantità di attacchi predetti in maniera corretta per ogni algoritmo.
'attack' rappresenta il numero originale di attacchi dentro il Dataset,
'dt' rappresenta l'algoritmo Decision Tree,
'knn' rappresenta l'algoritmo K-nearest neighbors,
'rf' rappresenta l'algoritmo Random Forest,
'ada' rappresenta l'algoritmo AdaBoost,
'bag' rappresenta l'algoritmo Bagging.

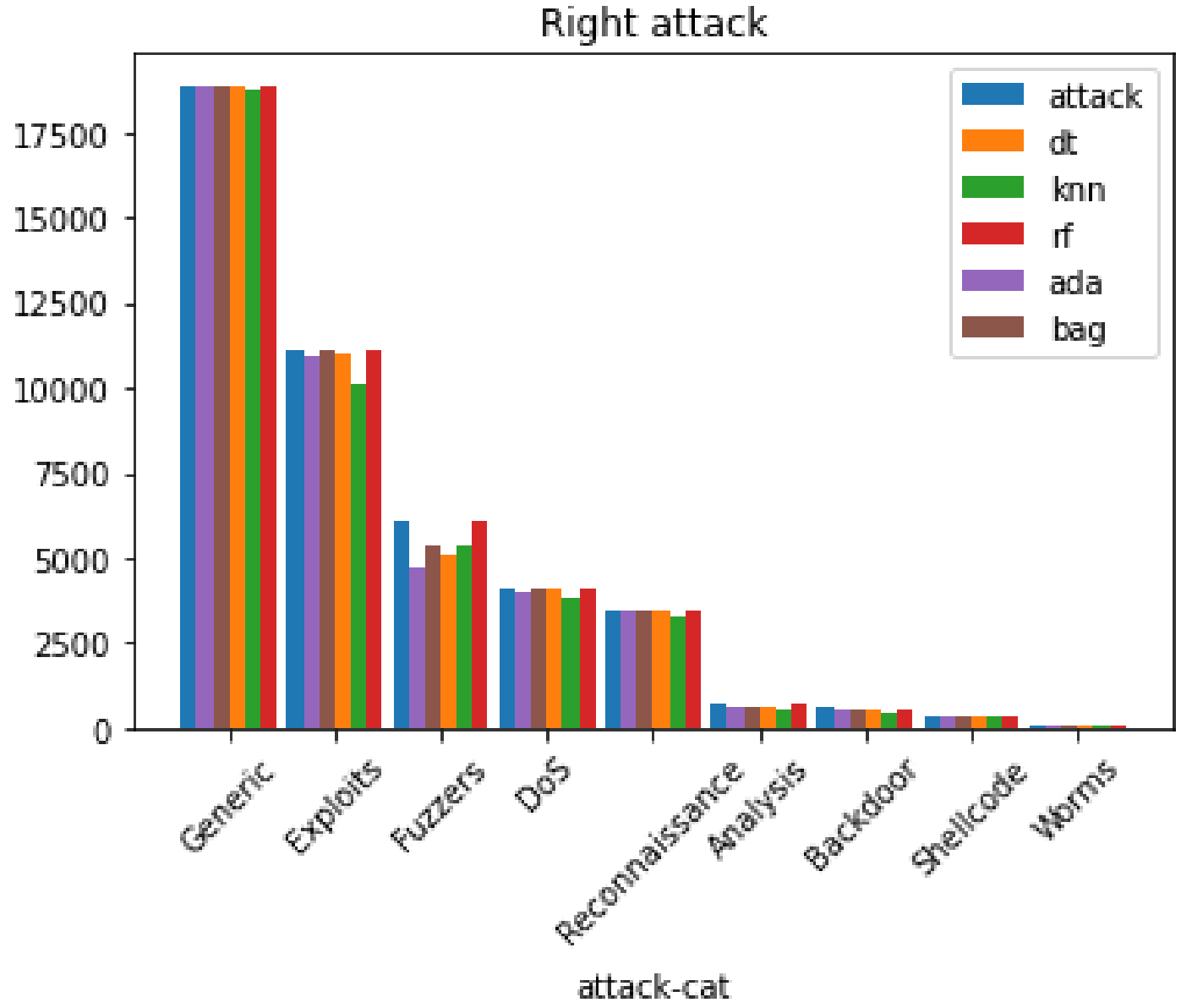

Figura 24 Occorrenze predizioni corrette dei vari algoritmi per tipo d'attacco su attack

Le seguenti liste presentano i numeri per ognuna delle barre per colore in ordine per tipologie di attacchi presentati in figura e successivamente il numero totale dato dalla somma di quest'ultimi.
attack: [18871, 11132, 6062, 4089, 3496, 677, 583, 378, 44]
somma = 45332

dt: [18858, 11008, 5092, 4055, 3483, 624, 580, 372, 43]
somma = 44115

knn: [18800, 10120, 5402, 3845, 3288, 579, 486, 359, 37]
somma = 42916

rf: [18864, 11119, 6062, 4053, 3496, 677, 577, 378, 44]
somma = 45270

ada: [18857, 10970, 4721, 4043, 3475, 642, 581, 367, 43]
somma = 43699

bag: [18868, 11063, 5357, 4068, 3492, 641, 580, 376, 43]
somma = 44488

- Su 18871 attacchi di tipo Generic nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo di Bagging è quello che ci si è avvicinato di più non rilevandone 3 occorrenze, invece il peggiore è stato K-nearest neighbors con 71 attacchi non rilevati.
- Su 11132 attacchi di tipo Exploits nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo Random Forest è quello che ci si è avvicinato di più non rilevandone 13 occorrenze, invece il peggiore è stato K-nearest neighbors con 1012 attacchi non rilevati.
- Su 6062 attacchi di tipo Fuzzers solo l'algoritmo Random Forest è riuscito a rilevarne tutte le occorrenze, mentre il peggiore è stato AdaBoost con 1341 attacchi non rilevati.
- Su 4089 attacchi di tipo DoS nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo di Bagging è quello che ci si è avvicinato di più non rilevandone 21 occorrenze, invece il peggiore è stato K-nearest neighbors con 244 attacchi non rilevati.
- Su 3496 attacchi di tipo Reconnaissance solo l'algoritmo Random Forest è riuscito a rilevarne tutte le occorrenze, mentre il peggiore è stato K-nearest neighbors con 208 attacchi non rilevati.
- Su 677 attacchi di tipo Analysis solo l'algoritmo Random Forest è riuscito a rilevarne tutte le occorrenze, mentre il peggiore è stato K-nearest neighbors con 98 attacchi non rilevati.
- Su 583 attacchi di tipo Backdoor nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo AdaBoost è quello che ci si è avvicinato di più non rilevandone 2 occorrenze, invece il peggiore è stato K-nearest neighbors con 97 attacchi non rilevati.
- Su 378 attacchi di tipo Shellcode solo l'algoritmo Random Forest è riuscito a rilevarne tutte le occorrenze, mentre il peggiore è stato K-nearest neighbors con 19 attacchi non rilevati.
- Su 44 attacchi di tipo Worms solo l'algoritmo Random Forest è riuscito a rilevarne tutte le occorrenze, mentre il peggiore è stato K-nearest neighbors con 7 attacchi non rilevati.

L'algoritmo che in generale ha rilevato più attacchi è Random Forest con 45270 record classificati come attacchi rilevati su 45332, mentre l'algoritmo generalmente peggiore è K-nearest neighbors con 42916.

L'algoritmo Random Forest che guarda unicamente sttl è riuscito comunque ad ottenere il risultato migliore, mostrando però che non tutti i record degli attacchi sono rilevabili tramite la features, ma comunque è una variabile molto importante per la predizione. K-nearest neighbors ha maggiori difficoltà probabilmente per la natura dei dati molto complessi e di difficile interpretazione e trovare dei record simili tra loro non sempre corrisponde ad una predizione corretta. L'algorimo AdaBoost è stato l'unico a fare peggio di K-nearest neighbors nella rilevazione degli record classificati come attacchi di tipo Fuzzers dimostrando come in realtà seppur creando un Decision Tree su molti di questi record comunque sia difficile da predire.

Risultati sulle predizioni corrette di attack_cat

Similmente come si è fatto precedentemente si è creato lo stesso bar chat con gli stessi acronimi e lo stesso significato con le seguenti liste, ma rispetto agli algoritmi allenati per predire attack_cat, quindi delle istanze per essere considerate corrette devono essere stato predetto il valore corretto che rappresenta le varie tipologie di attacchi o il traffico normale.

Università
Ca'Foscari
Venezia

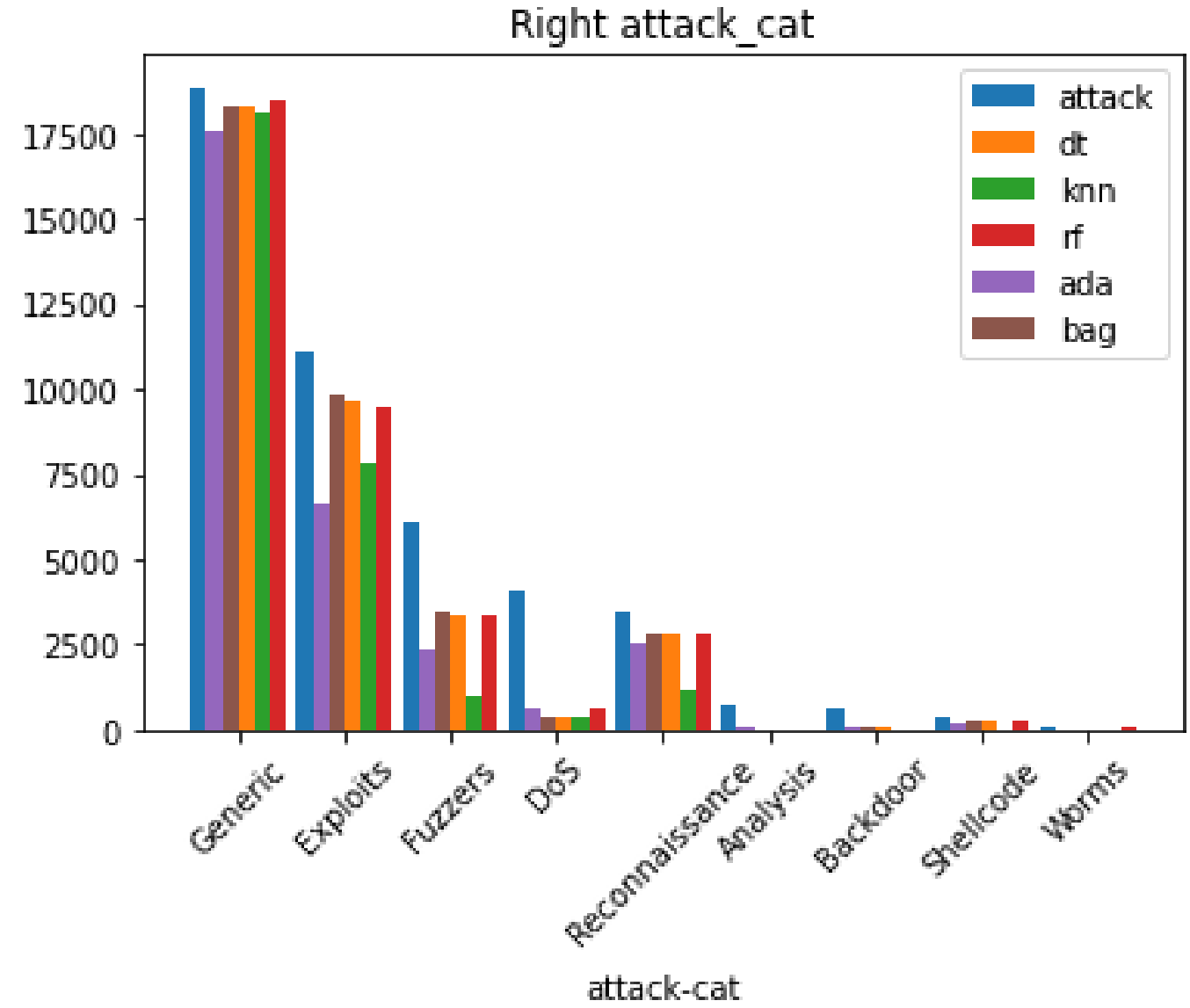

Figura 25 Occorrenze predizioni corrette dei vari algoritmi per tipo d'attacco su attack_cat

attack: [18871, 11132, 6062, 4089, 3496, 677, 583, 378, 44]
somma = 45332

dt: [18351, 9681, 3319, 354, 2808, 20, 76, 294, 26]
somma = 34929

knn: [18143, 7840, 1011, 339, 1190, 2, 2, 5, 0]
somma = 28532

rf: [18467, 9458, 3346, 634, 2843, 4, 25, 244, 37]
somma = 35058

ada: [17542, 6634, 2402, 626, 2584, 52, 112, 199, 2]

somma = 30153

bag: [18330, 9822, 3426, 332, 2811, 0, 103, 284, 23]
somma = 35131

- Su 18871 attacchi di tipo Generic nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo Random Forest è quello che ci si è avvicinato di più non rilevandone 404 occorrenze, invece il peggiore è stato AdaBoost con 1329 attacchi non rilevati.
- Su 11132 attacchi di tipo Exploits nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo di Bagging è quello che ci si è avvicinato di più non rilevandone 1310 occorrenze, invece il peggiore è stato AdaBoost con 4498 attacchi non rilevati.
- Su 6062 attacchi di tipo Fuzzers nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo di Bagging è quello che ci si è avvicinato di più non rilevandone 2636 occorrenze, invece il peggiore è stato K-nearest neighbors con 5051 attacchi non rilevati.
- Su 4089 attacchi di tipo DoS nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo Random Forest è quello che ci si è avvicinato di più non rilevandone 3455 occorrenze, invece il peggiore è stato quello di Bagging con 3757 attacchi non rilevati.
- Su 3496 attacchi di tipo Reconnaissance nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo di Random Forest è quello che ci si è avvicinato di più non rilevandone 653 occorrenze, invece il peggiore è stato K-nearest neighbors con 2306 attacchi non rilevati.
- Su 677 attacchi di tipo Analysis nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo di AdaBoost è quello che ci si è avvicinato di più non rilevandone 625 occorrenze, invece il peggiore è stato quello di Bagging con nessun attacco rilevato.
- Su 583 attacchi di tipo Backdoor nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo AdaBoost è quello che ci si è avvicinato di più non rilevandone 471 occorrenze, invece il peggiore è stato K-nearest neighbors con 581 attacchi non rilevati.
- Su 378 attacchi di tipo Shellcode nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo Decision Tree è quello che ci si è avvicinato di più non rilevandone 84 occorrenze, invece il peggiore è stato K-nearest neighbors con 373 attacchi non rilevati.
- Su 44 attacchi di tipo Worms nessun algoritmo è riuscito a rivelarli tutti correttamente, l'algoritmo di Random Forest è quello che ci si è avvicinato di più non rilevandone 7 occorrenze, invece il peggiore è stato K-nearest neighbors con nessun attacco rilevato.

L'algoritmo che in generale ha rilevato più attacchi è quello di Bagging con 3 5131 record classificati come attacchi rilevati su 45332, mentre l'algoritmo g eneralmente peggiore è K-nearest neighbors con 28532.

Come era intuibile si ottengono risultati peggiori nella rilevazione degli errori se si vuole anche predire la tipologia di attacco. Un interessante risultato è come l'algoritmo di Bagging sia il migliore grazie al suo fattore casuale di creazione delle parti dei Dataset per crare il Decision Tree da costruire, nonostante questo non ha classificato attacchi di tipo Analysis, forse perché sono attacchi facilmente scambiabili con altre tipologie visto che nella predizione precedente sulla variabile attack non ci sono state troppe difficoltà nell'individuazione, escludendo K-nearest neighbors.

Risultati sulle predizioni errate di attack_cat

Si è creato un bar chart che presenta il numero di occorrenze predette in maniera errata per ognuno dei 5 algoritmi allenati per le predizioni della variabile attack_cat, quindi la predizione non rappresenta il tipo classificato per quel record, creando così 5 barre per ogni tipologia d'attacco. Gli acronimi sono gli stessi con l'aggiunta di 'Normal' come elemento nell'asse delle x che indica i record del traffico normale; quindi, la barra rappresenta il numero di record normali classificati però come minaccia dagli algoritmi.

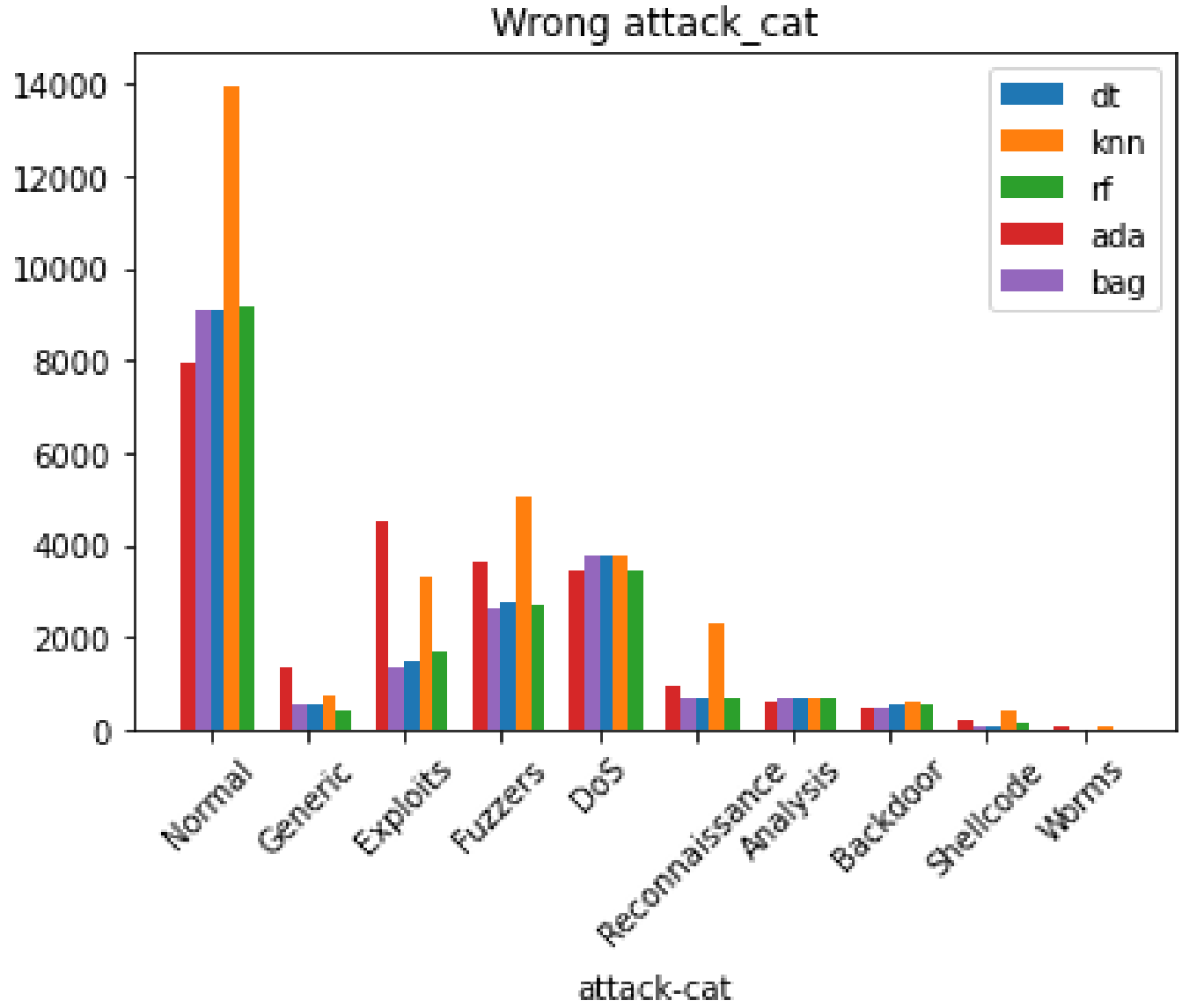

Figura 26 Occorrenze predizioni errate dei vari algoritmi per tipo d'attacco su attack_cat

Le seguenti liste presentano i numeri per ognuna delle barre per colore in ordine per tipologie possibili dei vari record presentati in figura e successivamente il numero totale dato dalla somma di quest'ultimi.
dt: [9113, 520, 1451, 2743, 3735, 688, 657, 507, 84, 18]
somma = 19516

knn: [13958, 728, 3292, 5051, 3750, 2306, 675, 581, 373, 44]
somma = 30758

rf: [9184, 404, 1674, 2716, 3455, 653, 673, 558, 134, 7]
somma = 19458

ada: [7953, 1329, 4498, 3660, 3463, 912, 625, 471, 179, 42]
somma = 23132

bag: [9097, 541, 1310, 2636, 3757, 685, 677, 480, 94, 21]
somma = 19298

- I record del traffico normale sono stati classificati erroneamente attacchi maggiormente dall'algoritmo K-nearest neighbors con 13958 errori, mentre quello che ne ha fatti di meno è l'algoritmo AdaBoost con 7953 errori.
- Gli attacchi di tipo Generic sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo AdaBoost con 1329 errori, mentre quello che ne ha fatti di meno è l'algoritmo Random Forest con 404 errori.
- Gli attacchi di tipo Exploits sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo AdaBoost con 4498 errori, mentre quello che ne ha fatti di meno è l'algoritmo di Bagging con 1310 errori.
- Gli attacchi di tipo Fuzzers sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo K-nearest neighbors con 5051 errori, mentre quello che ne ha fatti di meno è l'algoritmo di Bagging con 2636 errori.
- Gli attacchi di tipo DoS sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo di Bagging con 3757 errori, mentre quello che ne ha fatti di meno è l'algoritmo Random Forest con 3455 errori.
- Gli attacchi di tipo Reconnaissance sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo K-nearest neighbors con 2306 errori, mentre quello che ne ha fatti di meno è l'algoritmo Random Forest con 653 errori.
- Gli attacchi di tipo Analysis sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo di Bagging con 677 errori, mentre quello che ne ha fatti di meno è l'algoritmo AdaBoost con 625 errori.
- Gli attacchi di tipo Backdoor sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo K-nearest neighbors con 581 errori, mentre quello che ne ha fatti di meno è l'algoritmo AdaBoost con 471 errori.
- Gli attacchi di tipo Shellcode sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo K-nearest neighbors con 373 errori, mentre quello che ne ha fatti di meno è l'algoritmo Decision Tree con 84 errori.
- Gli attacchi di tipo Worms sono stati classificati erroneamente traffico normale o altre tipologie di attacco maggiormente dall'algoritmo K-nearest neighbors con 44 errori, mentre quello che ne ha fatti di meno è l'algoritmo

Random Forest con 7 errori.

In più rispetto al grafico precedente si può notare come l'algoritmo K-nearest neighbors ha fatto molte più classificazioni errate per i record del traffico normale, mostrando quindi un numero molto più elevato di falsi negativi senza però migliorare nella predizione degli altri attacchi.

Falsi negativi sulle predizioni di attack

Similmente come si è fatto precedentemente con le predizioni errate per la variabile attack_cat si è creato lo stesso bar chat con gli stessi acronimi ma sia le liste che il grafico rappresentano i numeri di falsi negativi per i vari algoritmi creati per la predizione della variabile attack. Essendo falsi negativi la parte rappresentante il traffico normale non è presente.

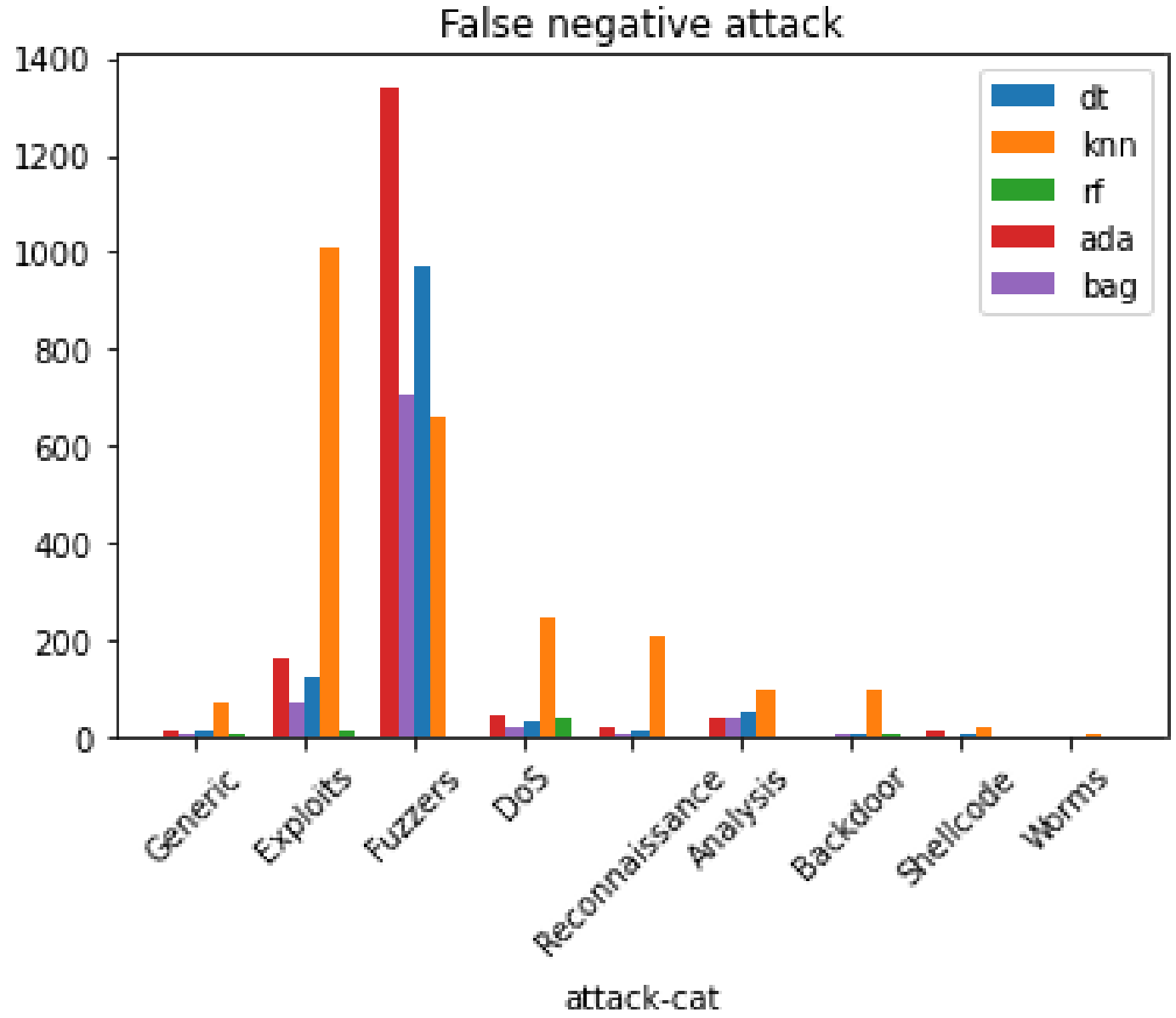

Figura 27 Occorrenze falsi negativi dei vari algoritmi per tipo d'attacco su attack

dt: [13, 124, 970, 34, 13, 53, 3, 6, 1]
somma = 1217

knn: [71, 1012, 660, 244, 208, 98, 97, 19, 7]
somma = 2416

rf: [7, 13, 0, 36, 0, 0, 6, 0, 0]
somma = 62

ada: [14, 162, 1341, 46, 21, 35, 2, 11, 1]
somma = 1633

bag: [3, 69, 705, 21, 4, 36, 3, 2, 1]
somma = 844

- Gli attacchi di tipo Generic sono quelli in proporzione soggetti al minor errore di predizione, tuttavia l'algoritmo con maggiori errori, K-nearest neighbors ha classificato 71 record come traffico normale, mentre quello più corretto è quello di Bagging che ha errato solo su 3 record.
- Gli attacchi di tipo Exploits sono stati classificati traffico normale maggiormente dall'algoritmo K-nearest neighbors che ha classificato 1012 record come traffico normale, mentre quello più corretto Random Forest ha errato solo su 13 record.
- Gli attacchi di tipo Fuzzers sono stati classificati traffico normale maggiormente dall'algoritmo AdaBoost che ha classificato 1341 record come traffico normale, mentre quello più corretto Random Forest non ha fatto errori.
- Gli attacchi di tipo DoS sono stati classificati traffico normale maggiormente dall'algoritmo K-nearest neighbors che ha classificato 244 record come traffico normale, mentre quello più corretto di Bagging ha errato solo su 21 record.
- Gli attacchi di tipo Reconnaissance sono stati classificati traffico normale maggiormente dall'algoritmo K-nearest neighbors che ha classificato 208 record come traffico normale, mentre quello più corretto Random Forest non ha fatto errori.
- Gli attacchi di tipo Analysis sono stati classificati traffico normale maggiormente dall'algoritmo K-nearest neighbors che ha classificato 98 record come traffico normale, mentre quello più corretto Random Forest non ha fatto errori.
- Gli attacchi di tipo Backdoor sono stati classificati traffico normale maggiormente dall'algoritmo K-nearest neighbors che ha classificato 97 record come traffico normale, mentre quello più corretto Random Forest ha errato solo su 6 record.
- Gli attacchi di tipo Shellcode sono stati classificati traffico normale maggiormente dall'algoritmo K-nearest neighbors che ha classificato 19 record come traffico normale, mentre quello più corretto Random Forest non ha fatto errori.
- Gli attacchi di tipo Worms sono stati classificati traffico normale maggiormente dall'algoritmo K-nearest neighbors che ha classificato 7 record come traffico normale, mentre quello più corretto Random Forest non ha fatto errori.

Il numero di falsi negativi maggiore è dell'algoritmo K-nearest neighbors con 2416, mentre quello minore è dell'algoritmo Random Forest con 62.

Come per le analisi precedenti si può notare come K-nearest neighbors tende ad avere generalmente più problemi nella predizione corretta dei vari record, mentre Random Forest si dimostra avere una precisione maggiore, nonostante all'utilizzo solo di sttl, rispetto anche all'AdaBoost che crea Decision Tree sugli errori degli alberi precedenti ma con scarsi risultati.

Falsi negativi sulle predizioni di attack_cat

Similmente a come si è fatto precedentemente si è fatto un bar chart con gli stessi significati e acronimi, solamente che per gli algoritmi allenati per predire attack_cat, quindi i falsi negativi sono rappresentati dal fatto che attacchi sono stati classificati come record normali, senza considerare sbagli di predizione sulla tipologia dello specifico attacco.

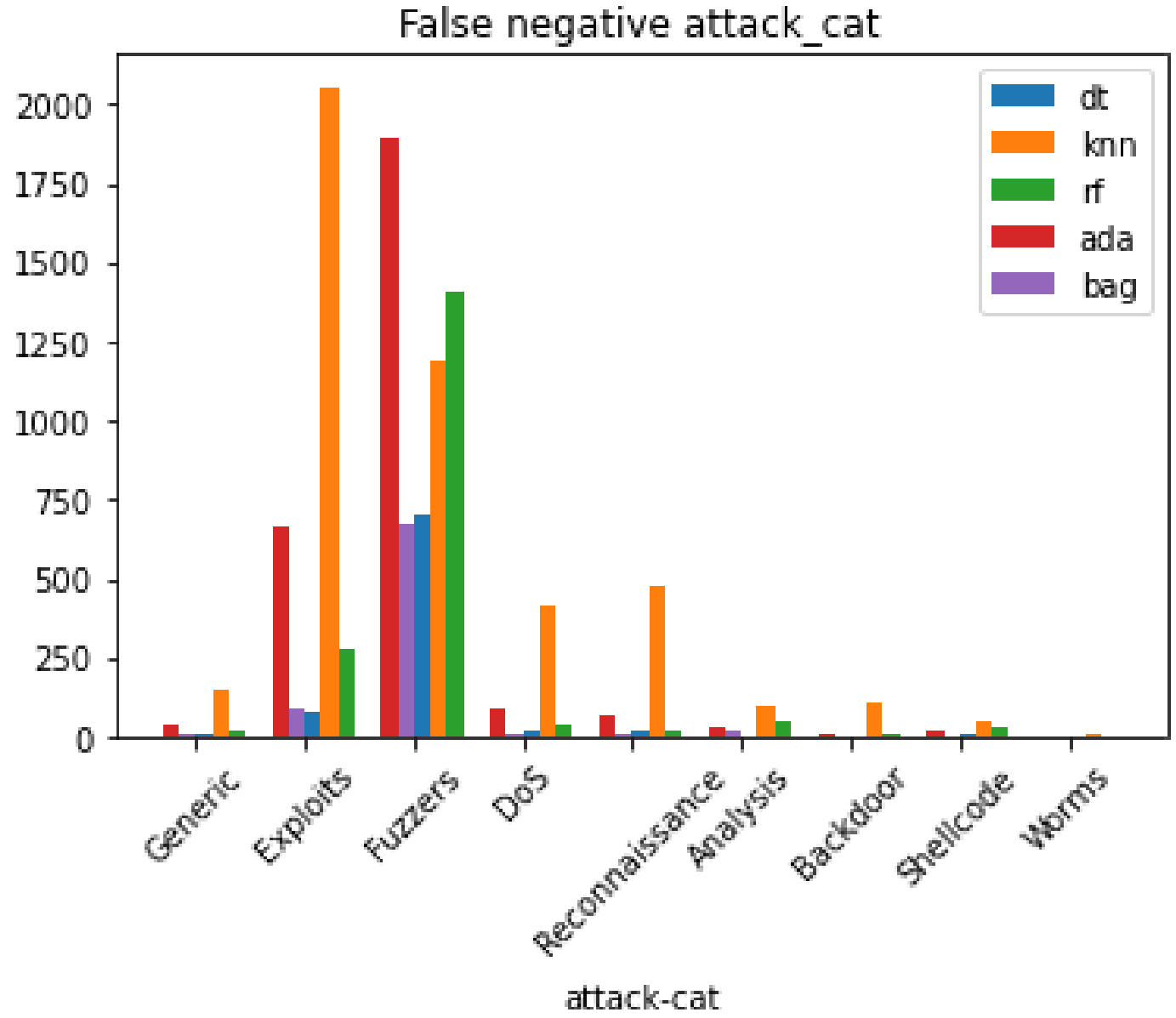

Figura 28 Occorrenze falsi negativi dei vari algoritmi per tipo d'attacco su attack_cat

dt: [8, 74, 702, 18, 16, 1, 1, 4, 0]
len = 824

knn: [149, 2055, 1193, 415, 476, 102, 105, 47, 10]
len = 4552

rf: [22, 274, 1408, 40, 19, 50, 5, 24, 1]
len = 1843

ada: [38, 668, 1893, 91, 64, 29, 4, 18, 2]
len = 2807

bag: [4, 89, 674, 13, 5, 18, 0, 3, 1]
len = 807

- Gli attacchi di tipo Generic sono quelli in proporzione soggetti al minor errore di predizione; tuttavia, l'algoritmo con maggior numero di attacchi classificati come normali è K-nearest neighbors con 149 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è quello di Bagging con 4 non rilevati.
- Per attacchi di tipo Exploits l'algoritmo con maggior numero di attacchi classificati come normali è K-nearest neighbors con 2055 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è Decision Tree con 74 non rilevati.
- Per attacchi di tipo Fuzzers l'algoritmo con maggior numero di attacchi classificati come normali è AdaBoost con 1893 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è quello di Bagging con 674 non rilevati.
- Per attacchi di tipo DoS l'algoritmo con maggior numero di attacchi classificati come normali è K-nearest neighbors con 415 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è quello di Bagging con 13 non rilevati.
- Per attacchi di tipo Recoinnaissance l'algoritmo con maggior numero di attacchi classificati come normali è K-nearest neighbors con 476 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è quello di Bagging con 5 non rilevati.
- Per attacchi di tipo Analysis l'algoritmo con maggior numero di attacchi classificati come normali è K-nearest neighbors con 102 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è Decision Tree con 1 non rilevati.
- Per attacchi di tipo Backdoor l'algoritmo con maggior numero di attacchi classificati come normali è K-nearest neighbors con 105 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è quello di Bagging che gli ha rilevati tutti.
- Per attacchi di tipo Shellcode l'algoritmo con maggior numero di attacchi classificati come normali è K-nearest neighbors con 47 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è quello di Bagging con 3 non rilevati.
- Per attacchi di tipo Worms l'algoritmo con maggior numero di attacchi classificati come normali è K-nearest neighbors con 10 non rilevati, mentre quello che almeno ha classificato questi record maggiormente come una tipologia di attacchi è quello Decision Tree che gli ha rilevati tutti.

L'algoritmo con il minor numero di falsi negativi è quello di Bagging con 807 attacchi classificati come normali, mentre quello con il numero maggiore è K-nearest neighbors con 4552 attacchi classificati come normali.

Come da risultati precedenti e dalla natura degli algoritmi, predire una variabile più complessa come attack_cat provoca una maggior presenza di falsi negativi anche non considerando se la tipologia predetta degli attacchi sia quella corretta o un'altra. Il numero di falsi negativi è si maggiore ma comunque non troppo distante per i corrisposti algoritmi allenati per predire attack, tranne per Decision Tree che in realtà ha un numero minore di falsi negativi allenandosi per attack_cat.

Falsi positivi sulle predizioni di attack

Si è creato un bar chart che presenta il numero di falsi positivi, record classificati come traffico normale predetti però come attacchi dagli algoritmi allenati per predire attack. Per ognuno dei 5 algoritmi si è presente una barra che ne rappresenta il numero grazie all'altezza di quest'ultima e ai numeri successivi al grafico. Gli acronimi sono gli stessi precedentemente usati.

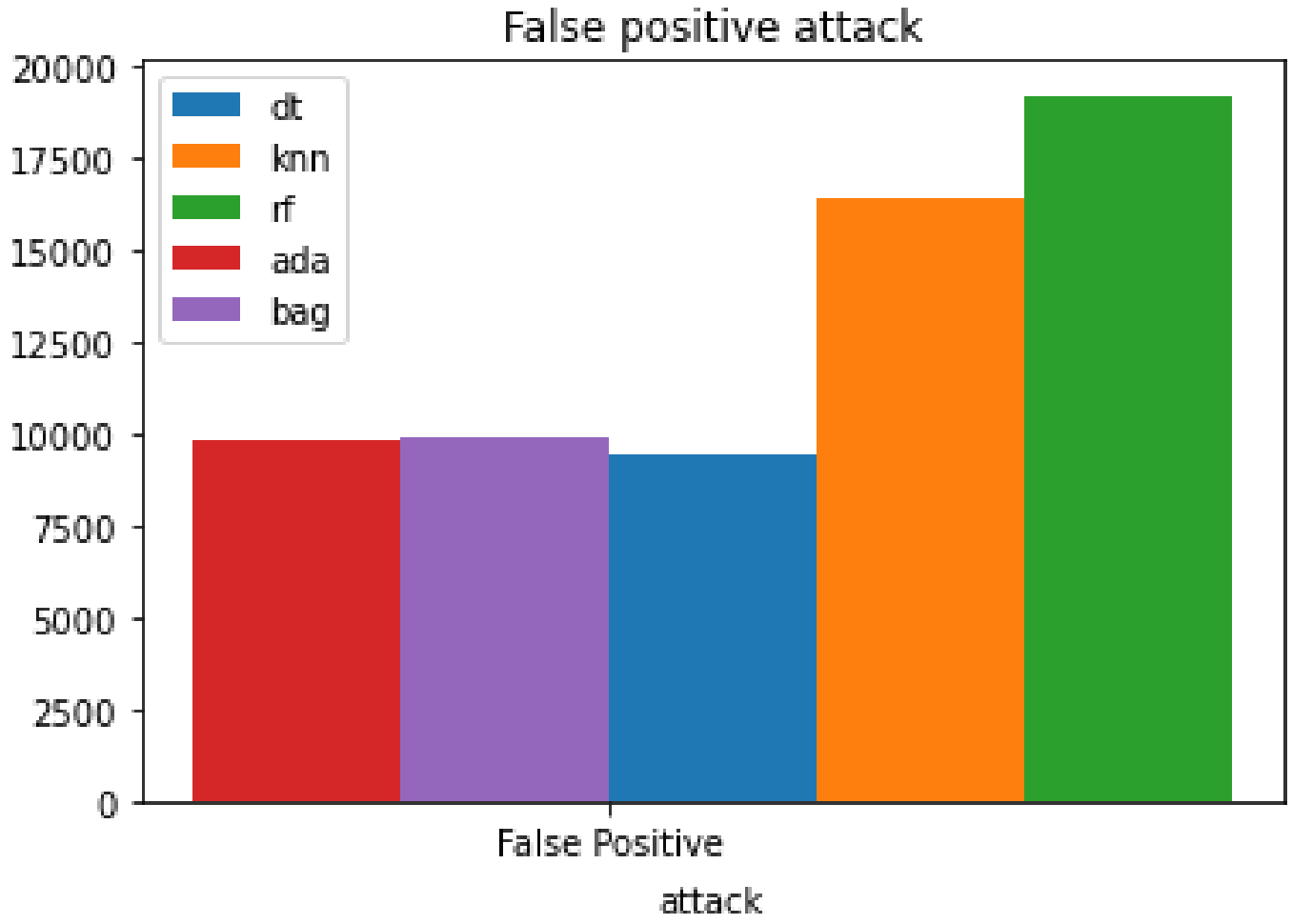

Figura 29 Occorrenze falsi positivi dei vari algoritmi su attack

dt: 9465

knn: 16415

rf: 19178

ada: 9776

bag: 9856

Il numero minore di falsi positivi è stato prodotto dall'algoritmo Decision Tree con 9465, mentre quello con il numero maggiore è Random Forest con 19178.

Questo risultato mostra come il basarsi su un'unica variabile come Random Forest non sia abbastanza significativo per dati così complessi, infatti il numero elevato di falsi positivi è dovuto a record del traffico normale con un sttl particolarmente elevato o comunque sia di simile valore rispetto altri

record di altri attacchi.

Falsi positivi sulle predizioni di attack_cat

Similmente a come si è fatto precedentemente si è fatto un bar chart con gli stessi significati e acronimi e le successive liste, solamente che per gli algoritmi allenati per predire attack_cat, quindi i falsi positivi sono record di traffico normale classificati come una tipologia di attacco, rappresentata in base i valori dell'asse delle x.

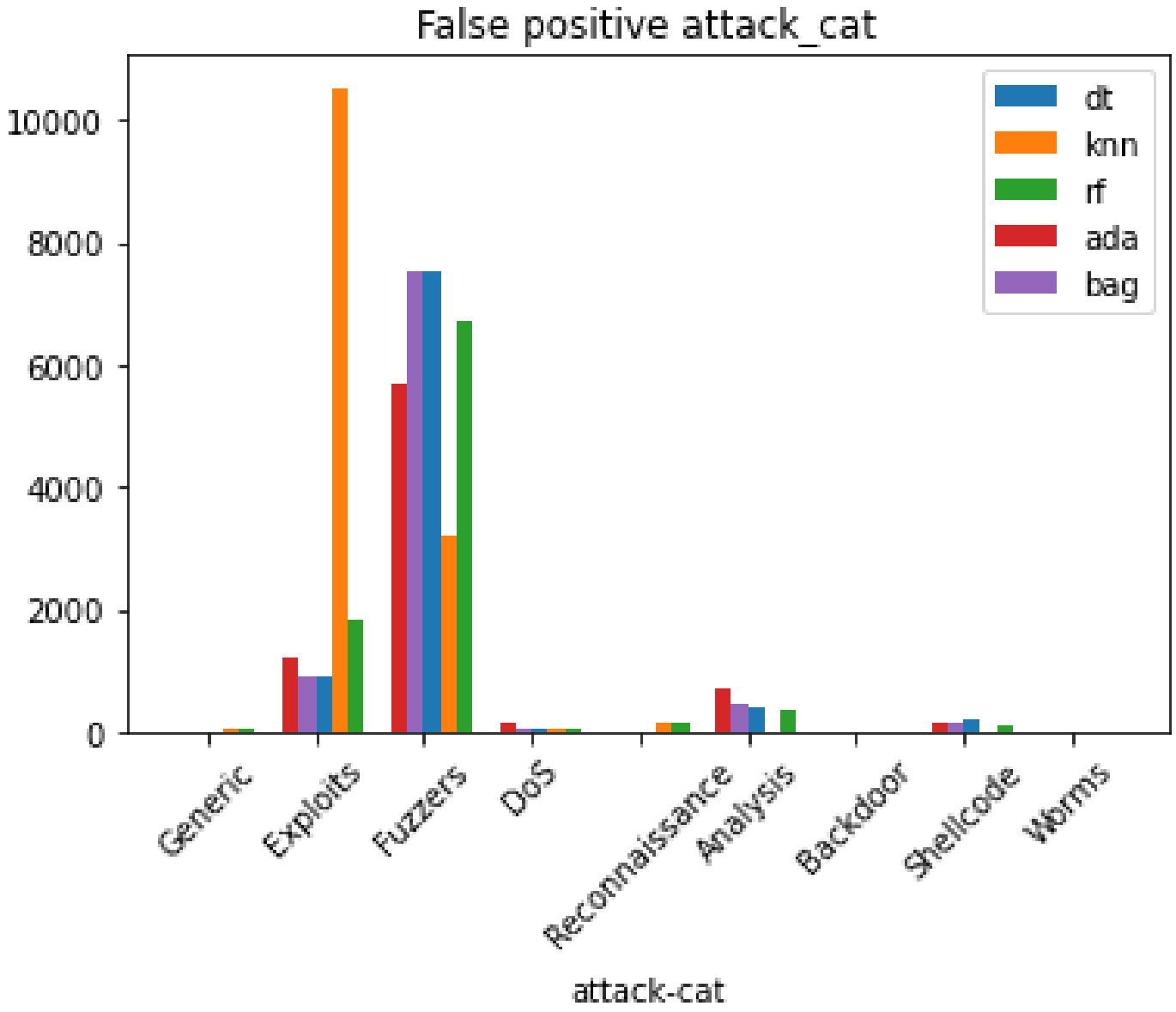

Figura 30 Occorrenze falsi positivi dei vari algoritmi per tipo d'attacco su attack

dt: [3, 901, 7511, 65, 11, 408, 11, 202, 1]
somma = 9113

knn: [55, 10527, 3187, 33, 144, 1, 0, 11, 0]
somma = 13958

rf: [24, 1819, 6689, 55, 135, 335, 2, 120, 5]
somma = 9184

ada: [15, 1227, 5693, 167, 8, 703, 0, 138, 2]
somma = 7953

bag: [3, 906, 7540, 35, 10, 459, 0, 143, 1]
somma = 9097
- Record del traffico normale sono stati classificati attacchi di tipo Generic maggiormente dall'algoritmo K-nearest neighbors 55 volte, mentre quelli con il minor numero di errori di questo tipo sono quello di Bagging e Decision Tree con 3 errori.
- Record del traffico normale sono stati classificati attacchi di tipo Exploits maggiormente dall'algoritmo K-nearest neighbors 10527 volte, mentre quello con il minor numero di errori di questo tipo è Random Forest con 901 errori.
- Record del traffico normale sono stati classificati attacchi di tipo Fuzzers maggiormente dall'algoritmo di Bagging 7540 volte, mentre quello con il minor numero di errori di questo tipo è K-nearest neighbors con 3187 errori.
- Record del traffico normale sono stati classificati attacchi di tipo DoS maggiormente dall'algoritmo AdaBoost 167 volte, mentre quello con il minor numero di errori di questo tipo è K-nearest neighbors con 33 errori.
- Record del traffico normale sono stati classificati attacchi di tipo Reconnaissance maggiormente dall'algoritmo K-nearest neighbors 144 volte, mentre quello con il minor numero di errori di questo tipo è K-nearest neighbors con 33 errori.
- Record del traffico normale sono stati classificati attacchi di tipo Analysis maggiormente dall'algoritmo AdaBoost 703 volte, mentre quello con il minor numero di errori di questo tipo è K-nearest neighbors con 1 errore.
- Record del traffico normale sono stati classificati attacchi di tipo Backdoor maggiormente dall'algoritmo Decision Tree 11 volte, mentre quelli con il minor numero di errori di questo tipo sono quelli di Bagging, AdaBoost e K-nearest neighbors con 0 errori.
- Record del traffico normale sono stati classificati attacchi di tipo Shellcode maggiormente dall'algoritmo Decision Tree 202 volte, mentre quello con il minor numero di errori di questo tipo è K-nearest neighbors con 11 errori.
- Record del traffico normale sono stati classificati attacchi di tipo Worms maggiormente dall'algoritmo Random Forest 5 volte, mentre quello con il minor numero di errori di questo tipo è K-nearest neighbors con 0 errori.

L'algoritmo K-nearest neighbors è quello che ha più falsi positivi, più precisamente 13958, mentre quello che ne ha meno è AdaBoost con 7953.

Si possono notare grazie a questo grafico alcune peculiarità degli algoritmi:
- K-nearest neighbors confonde molto spesso il traffico normale con attacchi di tipo Exploits, questo implica una somiglianza tra le due tipologie di record.
- Fuzzers, escludendo K-nearest neighbors che ha maggiori difficoltà con attacchi Exploits, è la tipologia di attacchi che sembra essere mischiata più spesso con il traffico normale da tutti gli algoritmi.
- AdaBoost grazie alla sua natura di creazione dei vari Decision Tree mostra un numero inferiore di errori di questo tipo, mentre il resto comunque rimane su un numero vicino e particolarmente elevato.

Risultati Finali

Per ottenere dei risultati finali si sono utilizzati diversi tipi di scoring:
- Accuracy = $\frac{Numero\ di\ predizioni\ corrette}{Numero\ totale\ di\ predizioni}$
- Precision = $\frac{Numero\ di\ predizioni\ corrette}{Numero\ predizioni\ corrette + Numero\ falsi\ positivi}$ (per le predizioni attack_cat il Numero di falsi positivi è dato dalla somma di tutti i record erroneamente classificati di una tipologia specifica, questo sommato alla stessa operazione prendendo però in considerazione una per volta tutte le varie tipologie classificabili)
- Recall = $\frac{Numero\ di\ predizioni\ corrette}{Numero\ predizioni\ corrette + Numero\ falsi\ negativi}$ (per le predizioni attack_cat il Numero di falsi negativi è dato dalla somma di tutti i record erroneamente non classificati di una tipologia specifica, questo sommato alla stessa operazione prendendo però in considerazione una per volta tutte le varie tipologie classificabili)
- F1 score = $2\frac{precision * recall}{precision + recall}$
- AUC-ROC = area sotto la curva ROC (presente solo per variabili binarie)

Algoritmi	Accuracy per attack	Accuracy per attack_cat	Precision per attack	Precision per attack_cat	Recall per attack	Recall per attack_cat	F1 score per attack	F1 score per attack_cat	AUC-ROC
Decision Tree	0.870257	0.762960	0.893195	0.775481	0.870257	0.762960	0.872985	0.750287	0.890510
KNN	0.771280	0.626415	0.837652	0.680718	0.771280	0.626415	0.782665	0.615801	0.809146

Random Forest	0.766312	0.763664	0.886340	0.781424	0.766312	0.763664	0.786663	0.756853	0.849480
AdaBoost	0.861427	0.719040	0.883996	0.680932	0.861427	0.719040	0.864284	0.693601	0.880298
Bagging	0.870038	0.765608	0.897158	0.781974	0.870038	0.765608	0.873174	0.753830	0.894241

Tabella 31 Risultati finali per vari score

Per i vari scoring ci sono diverse classifiche presentate dalle seguenti liste:
- Accuracy per algoritmi allenati per predire attack sul Dataset Test
[DecisionTree, Bagging, AdaBoost, K-nearest neighbors, Random Forest]

- Accuracy per algoritmi allenati per predire attack_cat sul Dataset Test
[Bagging, Random Forest, DecisionTree, AdaBoost, K-nearest neighbors]

- Precision per algoritmi allenati per predire attack sul Dataset Test
[Bagging, Decision Tree, Random Forest, AdaBoost, K-nearest neighbors]

- Precision per algoritmi allenati per predire attack_cat sul Dataset Test
[Bagging, Random Forest, Decision Tree, AdaBoost, K-nearest neighbors]

- Recall per algoritmi allenati per predire attack sul Dataset Test
[Decision Tree, Bagging, AdaBoost, K-nearest neighbors, Random Forest]

- Recall per algoritmi allenati per predire attack_cat sul Dataset Test
[Bagging, Random Forest, Decision Tree, AdaBoost, K-nearest neighbors]

- F1 score per algoritmi allenati per predire attack sul Dataset Test
[Bagging, Decision Tree, AdaBoost, Random Forest, K-nearest neighbors]

- F1 score per algoritmi allenati per predire attack_cat sul Dataset Test
[Random Forest, Bagging, Decision Tree, AdaBoost, K-nearest neighbors]

- AUC-ROC per algoritmi allenati per predire attack sul Dataset Test
[Bagging, Decision Tree, AdaBoost, Random Forest, K-nearest neighbors]

Conclusioni finali sullo studio

A seguito di questo studio si può affermare che la natura di questi dati rende complesso il lavoro di questi algoritmi se applicati come IDS, quindi per

identificare traffico considerabile come minaccia. Diverse tipologie di attacco differiscono l'una dalle altre per diverse variabili, nonostante siano presenti alcune features più importanti di altre visto la selezione di quest'ultime nell'algoritmo Random Forest e il risultato ottenuto da quello allenato per predire la variabile target attack, cioè quello di tenere unicamente la variabile sttl in considerazione, selezione che ha portato a risultati sicuramente rilevanti anche se penalizzati comunque molto da record del traffico normale con quel valore leggermente fuori scala rispetto al resto, infatti è il peggior algoritmo considerando lo score Recall che non considera i falsi positivi, quindi traffico normale che invece viene classificato come attacchi. L'algoritmo, infine, migliore per la predizione di attack sembra essere stato quello di Bagging grazie alla sua capacità di evitare l'overfitting, che è presente anche per la Random Forest ma in questo caso utilizzando un'unica variabile ha peggiorato questa sua abilità. Il risultato migliore invece per le predizione di attack_cat è sempre ottenuto dall'algoritmo di Bagging tranne per lo F1 score che in realtà è leggermente più rilevante rispetto agli altri, che è stato ottenuto da Random Forest, cosa sensata visto che quest'ultimo è un'evoluzione dell'algoritmo di Bagging.
Il risultato peggiore ottenuto quasi su tutti gli score, visto l'ultimo posto di Random Forest su attack con recall come score, è stato ottenuto da K-nearest neighbors che visto anche le osservazioni delle analisi precedenti si può capire come questa tipologia di dati, con le numerose presenze di variabili correlati tra loro e talvolta poco significative per la predizione, hanno reso inefficace questo algoritmo.
Confrontando i risultati degli algoritmi con quelli di uno studio simile Network Attack Detection and Classification Using Machine Learning Models Based on UNSW-NB15 Data-Set di Rackshit Punjari (https://i-rakshitpujari.medium.com/network-attack-detection-and-classification-using-machine-learning-models-based-on-unsw-nb15-a645bba73987) si può notare che una maggior lavorazione dei dati e un maggiore pool di iper-parametri testati hanno portato a dei risultati differenti. Più nello specifico si sono presi per un confronto gli algoritmi in comune (Decision Tree, K-nearest neighbors, Random Forest) e con lo stesso tipo di scoring, cioè l'accuracy.

Algoritmi	Accuracy per attack	Accuracy per attack_cat
Decision Tree	86.4 %	76.19 %
KNN	83.63 %	69.55 %
Random Forest	87.65 %	76.2 %

Tabella 32 Risultati altro studio

Nello studio sopracitato si ottengono i seguenti risultati rispetto allo studio di questo documento:

- K-nearest neighbors ottiene risultati migliori sia nella predizione di attack che in quella di attack_cat dimostrandosi un algoritmo che necessita una lavorazione più completa sugli iper-parametri e sui dati.
- Random Forest ottiene risultati migliori solo nella predizione di attack dove non si utilizza solamente una variabile, ma nella predizione di attack_cat si ottengono risultati molto simili.
- Decision Tree ottiene risultati peggiori nella predizione di attack e simili in quella di attack_cat, avendo potuto testare anche con la strumentazione sopracitata in maniera adeguata gli iper-parametri di questo algoritmo essendo molto semplice.

In conclusione si può dire che comunque sia, come era intuibile, non sembra convenire realizzare algoritmi che cercano non solo se è presente un attacco nella rete, ma anche classificarne la sua tipologia, perché rende sia più lento il controllo che più impreciso a livello pratico.

Citazioni

Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." *Military Communications and Information Systems Conference (MilCIS)*, 2015. IEEE, 2015.

Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." *Information Security Journal: A Global Perspective* (2016): 1-14.

Moustafa, Nour, et al. "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." *IEEE Transactions on Big Data (2017).*

Moustafa, Nour, et al. "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." *Data Analytics and Decision Support for Cybersecurity. Springer, Cham, 2017.* 127-156.

Sarhan, Mohanad, Siamak Layeghy, Nour Moustafa, and Marius Portmann. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings* (p. 117). Springer Nature.

Università
Ca'Foscari
Venezia

Rackshit Punjari. Network Attack Detection and Classification Using Machine Learning Models Based on UNSW-NB15 Data-Set. In https://i-rakshitpujari.medium.com/network-attack-detection-and-classification-using-machine-learning-models-based-on-unsw-nb15-a645bba73987.

Indice Tabelle

Indice figure