Università Politecnica delle Marche

Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea in Ingegneria Informatica e dell'Automazione

Sistema per malware detection basato sull'analisi del traffico DNS

A System for malware detection based on the DNS traffic analysis

Relatore:
Prof. Alessandro Cucchiarelli

Candidato:
Agostino Dati

Correlatore:
Prof. Christian Morbidoni

Anno Accademico 2019/2020

Abstract

Tra le minacce presenti sulla rete *Internet,* quella degli attacchi informatici mediante l'utilizzo di *botnet* è tra le più pericolose. Data la loro pericolosità, sono oggetto di studio nell'ambito della sicurezza informatica. Una *botnet* è una rete di computer, detti *bot*, controllati da un *cyber* criminale, il *bot master*, che può essere utilizzata per azioni illegali e fraudolente come attacchi *DDoS* e furto di dati. Il controllo sulla botnet da parte del bot master viene esercitato mediante l'utilizzo di un server (*Command & Control Server*) raggiungibile mediante un nome di dominio. Quest'ultimo è generato da specifici algoritmi condivisi tra i bot e bot master, i *Domain Generation Algorithms* (DGA), che sono suddivisi in famiglie in base al loro funzionamento. In particolare, il DGA genera un numero elevato di nomi di dominio e una volta che il bot master registra nel *DNS* (*Domain Name Service*) globale uno dei nomi di dominio generati il bot, cercando di contattare tutti i nomi generati dall'algoritmo, raggiunge il C&C Server. In questo modo si instaura un canale di comunicazione tra dispositivi infetti e cyber criminale molto robusto. Per poter rendere innocua una botnet, si può usare la tecnica di *sink-holing*: una volta individuati i nomi di domini malevoli che conducono al C&C server, le richieste indirizzate verso essi vengono reindirizzate verso un *host* non malevolo e slegato dal C&C server. In questa tesi ci si pone l'obiettivo di realizzare un sistema che possa individuare tali nomi di dominio malevoli e classificarli in base alla famiglia di DGA che li ha generati utilizzando tecniche di *machine learning supervisionato* (*Support Vector Machine, Random Forest* e *MLP*), partendo da un dataset di nomi di dominio creato in precedenza da altri tesisti. Dal dataset vengono estratte le *features lessicali* dei nomi di dominio benevoli e malevoli, che indicano

la loro divergenza e similarità. Ciò permette di far emergere le differenze tra i nomi di dominio generati automaticamente dagli algoritmi (DGA) e quelli creati dall'uomo. Il dataset è stato suddiviso in training set e test set. Il modello per la classificazione multi-classe ottenuto produce, per ogni nome di dominio fornitogli, un numero intero compreso tra 0 e 25. Questo valore indica se il nome di dominio è benevolo (0) o se è generato da una delle 25 famiglie di DGA conosciute dal classificatore (1-25). Mediante l'utilizzo di metriche di valutazione (*Accuracy*, Precision*, F1-score, Recall)* si valuta la bontà del classificatore. Attraverso questi parametri si può confrontare il classificatore multi-classe con quello binario, che produce in output due valori interi: 0 se il nome di dominio è benevolo, 1 altrimenti. Da questo confronto si osserva come il classificatore multi classe produce ottimi risultati, a fronte di un lieve peggioramento delle prestazioni rispetto al classificatore binario.

Indice

1. Introduzione

Per la società odierna la rete di telecomunicazione *Internet* costituisce un tassello fondamentale. Attualmente le possibilità offerte da Internet sono molteplici: comunicare in tempi brevissimi con altri utenti, che hanno accesso alla rete, distanti migliaia di chilometri, accedere ad un'enorme quantità di informazioni oppure eseguire transazioni economiche e commerciali.

Queste ed altre operazioni sono possibili grazie al continuo scambio di informazioni e dati, che risultano essere una risorsa molto ambita non solo da aziende e governi, ma anche dalla criminalità. Infatti, la criminalità informatica è cresciuta fortemente negli ultimi anni provocando danni rilevanti sia sul piano sociale che su quello economico. [1]

Esistono diversi modi con cui un *cyber* criminale può appropriarsi indebitamente dei dati degli utenti, a partire da truffe telematiche fino all'utilizzo di *software* dannosi. Questa tipologia di programmi, denominati *malware*, contengono precise istruzioni volte a compiere operazioni criminose. Nel panorama delle minacce informatiche, ve n'è una particolarmente pericolosa e per questo oggetto di studio: le *Botnet*. Una botnet è una rete di computer, denominati *bot*, collegati ad Internet e controllati da una singola sorgente, chiamata *bot master*. I computer infettati dal malware entrano a far parte della botnet e una volta ricevuti gli ordini dal bot master li eseguono.

Ciò che rende una botnet temibile è la capacità di essere formata da migliaia, se non milioni, di computer pronti ad eseguire istruzioni potenzialmente fraudolente, volte a compiere attacchi informatici di diverso tipo. Tra questi si trovano la diffusione di *ransomware* [1], *trojan* [2] bancari e gli attacchi *DDoS* [3].

Un elevato numero di dispositivi infetti rende le botnet ben distribuite e difficili da tracciare. Inoltre, maggiore è il numero di bot maggiori saranno il numero di attacchi ed il conseguente ritorno economico.

Pertanto, uno dei principali obiettivi del bot master è quello di diffondere il malware quanto più possibile così da massimizzare il numero di bot all'interno della rete.

Un'ulteriore precauzione presa dai cyber criminali è quella di utilizzare domini *HTTP* che identificano il server *C&C (Command and Control Channel)*. Questi domini sono difficili da identificare poiché sono generati automaticamente e in grandi quantità da specifici algoritmi, i *Domain Generation Algorithm* (*DGA*). Usando i DGA, i bot generano i loro nomi di dominio in maniera pseudo-casuale attraverso un apposito *seme*. DGA e seme sono condivisi tra bot e botmaster. Una volta che il botmaster registra nel DNS globale uno dei nomi di dominio generati dal DGA il bot, cercando di contattare tutti i nomi generati dall'algoritmo, raggiunge il C&C Server. Questa tecnica garantisce un canale di comunicazione molto affidabile tra i bot i il bot master.

Dunque, lo studio dei DGA risulta essere di primaria importanza per poter identificare i bot e bloccare il collegamento con il server C&C. In tal modo il dispositivo infettato non può ricevere ordini dal bot master, indebolendo l'intera botnet.

Lo scopo della tesi è quello di realizzare un sistema che permette di individuare i domini utilizzati dai bot per collegarsi con il server C&C e di identificare da quale famiglia di DGA è stato generato. Infatti, i DGA possono essere classificati in famiglie in base al tipo di algoritmo utilizzato e ad altri fattori. Queste operazioni vengono svolte attraverso l'analisi del traffico DNS e l'uso di algoritmi di *Machine Learning*.

2. Contesto applicativo

Una Botnet è una rete di computer, denominati bot, collegati ad Internet e controllati da una singola sorgente, chiamata bot master. I bot eseguono le istruzioni fornite dal bot master.

In questo capitolo verrà analizzata la struttura, il funzionamento e le tecniche di comunicazione delle botnet, utilizzata per fini illegali.

2.1 Struttura e funzionamento

Una botnet è costituita da diversi elementi:

- **Malware**: è un programma malevolo che, eseguito su un dispositivo infetto, opera per compromettere la sicurezza del sistema che lo ospita;
- **Bot**: sono i dispositivi, di utenti ignari, infettati dal malware e che hanno stabilito il collegamento con il C&C;
- **Bot master**: è il cybercriminale che governa la botnet, fornisce ai bot le istruzioni affinché essi si propaghino o eseguano codici malevoli;
- **Command and Control Server (C&C)**: è il server tramite il quale il botmaster controlla l'intera botnet. Consente lo scambio di messaggi tra

il malware in esecuzione sul dispositivo infetto e il server C&C controllato dal cybercriminale.

Il ciclo di vita di una botnet può essere suddiviso in quattro fasi:

1. **Infezione**: i dispositivi vengono infettati da un software malevolo, il *loader*. Il loader eseguirà allora il download dei codici binari del malware e lo eseguirà;
2. **Comunicazione con il C&C**: il bot cerca di instaurare un punto di rendez-vous per collegarsi al server C&C. Una volta stabilita la connessione il bot master potrà comunicare con il dispositivo infettato e avviare la procedura di mantenimento e aggiornamento del bot, scambiando diverse informazioni;
3. **Attacchi da eseguire**: il bot master (tramite il C&C Server) invia le informazioni necessarie ai bot per poter eseguire degli attacchi;
4. **Propagazione**: vengono attivati i meccanismi di propagazione del malware dal bot master affinché aumenti il numero di bot all'interno della rete.

La diffusione del malware è messa in atto dal bot master e può essere effettuata in diversi modi:

- E-mail: il malware viene diffuso come allegato o tramite un collegamento cliccabile che reindirizza a siti web progettati appositamente per indurre l'utente a scaricare ed eseguire il malware sul proprio dispositivo;

- Falle di sicurezza: vengono sfruttate le vulnerabilità del sistema operativo per permettere l'esecuzione automatica del malware;
- Applicazioni di largo utilizzo non originali: vengono diffuse in rete applicazioni modificate simili ad applicazioni note e ben conosciute che svolgono le stesse funzionalità dell'originale ma che nascondono al loro interno delle istruzioni malevole, che verranno eseguite una volta avviato il programma.

Il malware viene creato attraverso l'uso di appositi strumenti, tra cui *Rootkit*, collezioni di software solitamente malevoli realizzati per ottenere l'accesso ad un computer o alle sue componenti. Questi software si occupano anche di mascherare sé stessi o altri programmi utili a raggiungere l'obiettivo per il quale sono stati creati. L'invisibilità del malware risulta essere fondamentale, in tal modo può non essere rilevato dai sistemi di sicurezza del dispositivo infettato e dunque evitare la propria identificazione e rilevazione.

Il malware, dopo aver infettato il dispositivo, tenta di instaurare un collegamento con il C&C. Completata anche questa operazione, il bot master potrà fornire le istruzioni desiderato al nuovo bot della rete.

Il collegamento può essere realizzato usando diverse tecniche:

- **HTTP** *(Hypertext Tranfer Protocol):* la botnet viene controllata dal bot master attraverso l'uso di un sito web basato sul protocollo HTTP. In

questo modo il *firewall* del dispositivo infettato non distinguerà il traffico *DNS* (*Domain Name Services*) malevolo da quello non-malevolo. Dunque, il protocollo HTTP garantisce la comunicazione tra bot e C&C, poiché viene considerato normale traffico su siti web.

- **IRC** *(Internet Relay Chat):* questo protocollo di messaggistica istantanea online consente sia la comunicazione diretta fra due utenti che lo scambio di messaggi tra un insieme di individui raggruppati in "stanze", chiamate *canali*. Molte botnet sfruttano questi canali usando i dispositivi infettati come utenti della *chat*. Essi rimangono in attesa di ricevere messaggi che contengono le istruzioni comunicate dal bot master.
- **P2P** *(Peer-to-peer):* in una rete P2P i nodi (*peer*) sono dispositivi informatici collegati via Internet. I file vengono condivisi direttamente tra i nodi della rete senza dover usare un server centrale. Dunque, ogni computer che opera all'interno di una rete P2P funge sia da server che da client. Il bot master utilizza una rete già esistente per diffondere il malware sui dispositivi vulnerabili, con il rischio di essere eseguiti dagli utenti stessi, inconsapevoli della malignità di tali software.

Instaurando il collegamento C&C tra bot e bot master, viene introdotto un server tra i due che ha il compito di schermare il master, così che quest'ultimo non possa essere individuato dall'esterno. Il bot master, dunque, invia le istruzioni al server intermediario che le distribuisce ai bot della rete.

Risulta evidente che tale server è di vitale importanza all'interno della botnet. Allo stesso tempo può essere l'anello debole della rete. Solitamente il server è costituito da una sola macchina e se venisse danneggiato o neutralizzato, ne risentirebbe l'intera botnet poiché verrebbe interrotta la comunicazione con i bot e l'intera rete collasserebbe. Una botnet così organizzata viene definita botnet *centralizzata* (Figura 1).

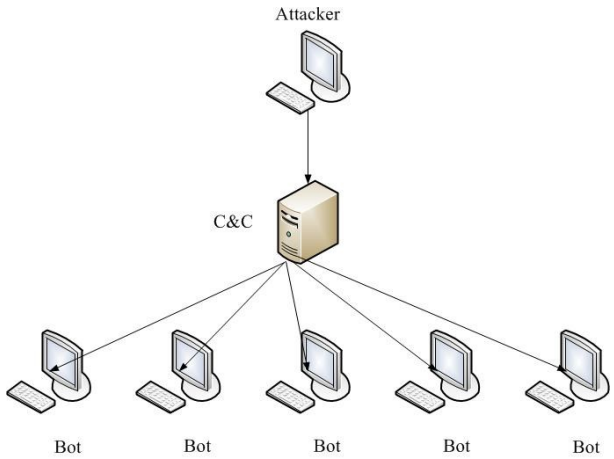

Figura 1 - Botnet Centralizzata

Per minimizzare i danni in caso di un malfunzionamento del server C&C, se ne possono usare molteplici che comunichino reciprocamente. In tal modo, se si dovesse perdere un server, il master può continuare a comunicare con i bot attraverso i C&C server rimanenti. Una configurazione con questa topologia è detta rete *centralizzata a stella estesa* (Figura 2).

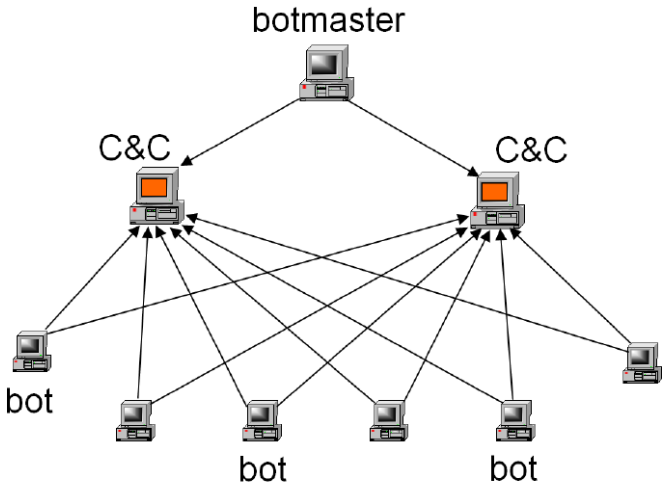

Figura 2 - Botnet centralizzata a stella estesa

Lo svantaggio principale di un'architettura centralizzata risulta essere il legame con il server principale. Se il C&C server viene neutralizzato, l'intera botnet viene compromessa e potrebbe risultare molto semplice rilevare il server principale.

Una soluzione a questo problema viene ovviata utilizzando una diversa topologia della rete, quella *decentralizzata* (Figura 3). In tale configurazione non esiste un server centrale ma ogni bot può agire come server C&C. Infatti, essa è realizzata attraverso una rete P2P. In questo modo la botnet è più sicura e risulta essere più difficile da smantellare.

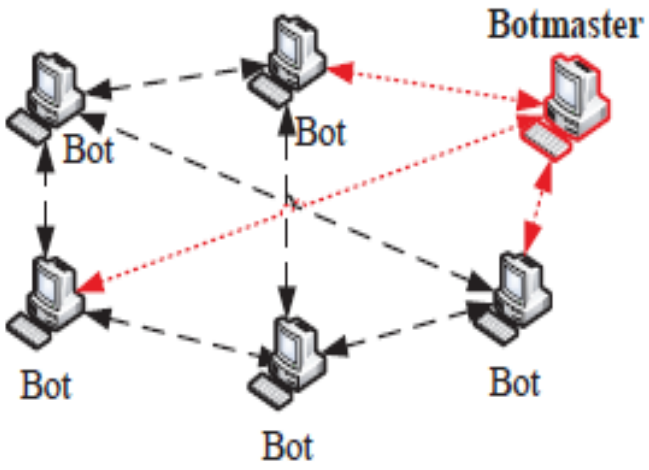

Figura 3 - Botnet decentralizzata

Vi è un'ultima tipologia di botnet caratterizzata da una struttura non del tutto definita; la botnet *non strutturata* (Figura 4). Data la complessità molto elevata di tale rete, non è molto utilizzata poiché vi sono tempi di latenza molto elevati e incertezza nella consegna dei pacchetti.

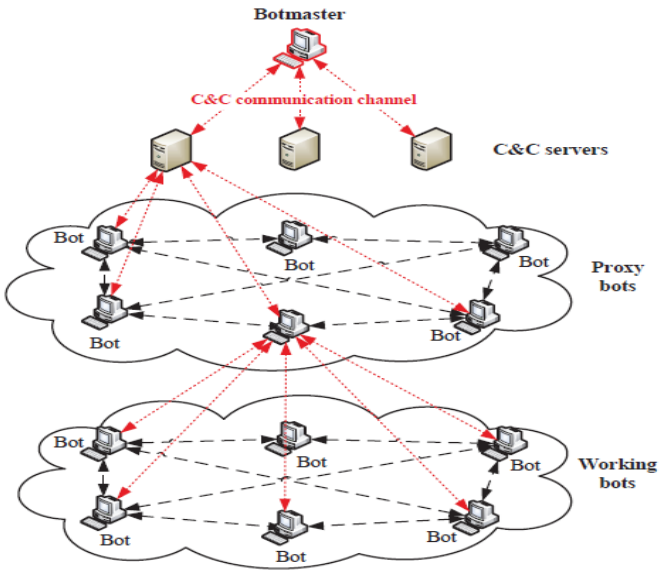

Figura 4 - Botnet non strutturata

2.2 Tipologie di attacchi tramite botnet

Le tipologie di attacchi per cui le botnet sono utilizzate risultano essere diretti sia a siti web che ad utenti specifici. Tra gli attacchi più comuni rientrano:

- **DDoS** (Distributed Denial of Service): è una tipologia di attacco che consiste nel sovraccaricare di richieste un servizio Internet. Dato il numero elevato di cui è costituita generalmente una botnet, questo enorme quantitativo di richieste sfocia nell'impossibilità di erogare dei servizi di cui si fa richiesta. Tutto ciò porta ad un blocco o ad un riavvio forzato dei server del servizio web vittima, creando così un disservizio che si risolve in un danno economico e di immagine per la vittima.
- **Spyware**: il bot master impartisce l'ordine di installare un software, lo *spyware*, che raccoglie informazioni sull'attività online di un utente senza il suo consenso. L'installazione avviene invisibilmente. I dati raccolti dagli spyware installati su diversi dispositivi vengono usati dal bot master in diversi modi, ad esempio possono essere venduti in cambio di un compenso.

- **Adware**: gli *adware*, che vengono installati sui dispositivi designati, son un particolare tipo di malware, progettato per mostrare annunci pubblicitari, inviare e ricevere dati.
- **E-mail spam**: i bot tentano il recupero della lista di indirizzi e-mail dell'utente affinché possano inviare una notevole quantità di messaggi di posta elettronica. Solitamente queste e-mail contengono messaggi ingannevoli o hanno in allegato dei malware. Questa tipologia di attacco viene largamente usata per ingrandire la botnet, propagando il malware per l'infezione.

Come detto precedentemente, per neutralizzare una botnet si può procedere cercando di impedire ai bot di comunicare con il bot master in modo che non avvenga più lo scambio di informazioni, dati e ordini. In questo modo, il dispositivo infettato risulta essere escluso dalla botnet.

Affinché ciò avvenga, si può usare la tecnica di ***sink-holing*** attraverso la quale, una volta individuati, i nomi di domini malevoli che conducono al server C&C sono reindirizzati verso un *host* specifico. Tale host risulterà essere slegato dal server C&C originale così che i bot non possano comunicare con il bot master.

I cyber criminali hanno ideato nuove tecniche per aggirare anche il sink-holing rendendo più arduo il rilevamento dei domini malevoli. Tra queste tecniche vi è l'utilizzo di algoritmi che generano un elevato numero di nomi di dominio, i *Domain Generation Algorithms*.

2.3 Domain Generation Algorithms

I *Domain Generation Algorithms* sono utilizzati per generare dinamicamente un elevato numero di nomi di domini apparentemente casuali in un breve periodo di tempo. Tra questi, viene selezionato un sottoinsieme per la comunicazione C&C. La generazione dei domini avviene calcolandoli a partire da un **seme**, da un **elemento che varia col tempo** e dal **dominio di primo livello** (TLD). Il seme può essere una qualsiasi cosa decida il bot master: costanti numeriche, frasi e così via. Il seme risulta essere di vitale importanza poiché permette di creare i punti di *rendez-vous* tra il bot master e i bot stessi e può essere cambiato a piacimento. L'elemento che cambia col tempo, invece, non deve essere necessariamente una data (o un'ora), ma qualsiasi cosa che vari col tempo come un argomento di tendenza sui social network o la parola più cercata su Google in un dato paese al momento della connessione. Più questa variabile è difficile da prevedere, più è difficile per i "difensori" intercettare i domini. Infatti, il seme e l'elemento variabile col tempo combinati da un algoritmo generano il nome del dominio che verrà combinato con uno dei TLD disponibili.

L'utilizzo di questi algoritmi di generazione di domini pone i cyber criminali in una situazione di vantaggio rispetto agli esperti di sicurezza informatica e alle forze dell'ordine. Questi ultimi, infatti, devono controllare tutti i domini per

garantire una rimozione efficace mentre il bot master ha bisogno di accedere ad un singolo dominio per controllare i bot o far migrare il canale di comunicazione.

2.3.1 Classificazione dei DGA in famiglie

I DGA possono essere classificati in famiglie in base alla sorgente del seme e al tipo di algoritmo utilizzato.

Classificazione in base alla sorgente del seme:

- **Tempo dipendente**: l'algoritmo utilizza come seme una risorsa temporale;
- **Deterministico**: il seme è di tipo deterministico.

La dipendenza dal tempo e il determinismo consentono le seguenti quattro combinazioni: Tempo indipendente e deterministico (**TID**), dipendente dal tempo e deterministico (**TDD**), dipendente dal tempo e non deterministico (**TDN**), indipendente dal tempo e non deterministico (**TIN**).

Inoltre, sono differenziati quattro schemi di generazione:

- **Arithmetic-based DGA (A)**: calcolano una sequenza di valori rappresentati mediante la codifica ASCII che sono utilizzati direttamente per il nome di dominio oppure per creare un dizionario che viene usato per definire i nomi di dominio;

- **Hash-based DGA (H):** utilizzano la rappresentazione esadecimale di una tabella *hash* per produrre un nome di dominio;
- **Wordlist-based DGA (W)**: concatenano sequenze di parole provenienti da una o più liste predefinite, creando nomi di dominio meno casuali e più camuffabili;
- **Permutation-based DGA (P):** derivano tutti i possibili nomi di dominio attraverso la permutazione di un nome di dominio iniziale.

Queste classificazioni possono essere combinate per caratterizzare al meglio le famiglie di DGA. Quelle più diffuse sono le TDD-A e la TID-A. [5]

3. Obiettivo

L'obiettivo di questa tesi è quello di realizzare un sistema che analizza il traffico di rete al fine di rilevare gli indirizzi generati da DGA e individuarne anche la famiglia di appartenenza.

A questo scopo, verranno utilizzate tecniche di *machine learning*.

3.1 Machine Learning

Il *machine learning*, o apprendimento automatico, è una branca dell'intelligenza artificiale in cui si studiano algoritmi informatici che migliorano automaticamente se stessi attraverso l'esperienza e l'apprendimento. Analizzando dei dati di addestramento, questi sistemi creano modelli matematici che effettuano previsioni per dati mai analizzati prima. Per gli algoritmi di machine learning risulta essere fondamentale l'addestramento su quanti più dati possibile, in questa maniera essi possono effettuare previsioni sempre più precise.

Le tecniche di apprendimento più utilizzate sono l'apprendimento supervisionato e l'apprendimento non supervisionato.

Nell'apprendimento supervisionato gli algoritmi sono addestrati usando dati di cui si conosce la natura, espressa, in genere, dalla classe di appartenenza. L'algoritmo che usa l'apprendimento supervisionato, in fase di addestramento, impara ad abbinare ad un dato in input una specifica *classe* che verrà confrontata con quella nota del dato stesso. In caso di errore, il modello prodotto viene modificato di conseguenza. Questo modello viene utilizzato per prevedere i risultati fornendo dati non ancora classificati.

La classificazione è una categoria dell'apprendimento supervisionato in cui si cerca di prevedere le classi associate agli elementi di un nuovo data set, utilizzando l'esperienza acquisita in precedenza. L'etichetta è un valore che viene assegnato ad un dato e indica a quale classe appartiene. Un algoritmo di classificazione si divide in due fasi principali:

- Fase di addestramento (training): si fornisce un dataset di addestramento (training set), dei cui elementi si conoscono a priori le categorie di appartenenza, che viene usato dall'algoritmo per studiare i dati al fine di trovare un pattern e sviluppare un modello delle categorie (o classi).
- Test: dopo aver completato l'addestramento, all'algoritmo viene dato un dataset di test, con elementi distinti da quelli usati per il training, dei cui elementi si conoscono le classi di appartenenza, per valutare l'efficacia del modello creato dalla fase di training.

Solitamente, il dataset di addestramento e quello di test fanno parte di uno stesso dataset iniziale che viene suddiviso in maniera proporzionale in sottoinsiemi ad intersezione nulla.

L'apprendimento non supervisionato viene utilizzato su dati non classificati. L'algoritmo ricerca caratteristiche e pattern comuni per effettuare previsioni su input successivi.

Nel corso di questa tesi vengono utilizzati algoritmi che sfruttano l'apprendimento supervisionato per classificare singoli nomi di dominio identificando se sono generati da un DGA, è in caso lo fosse cercando di individuare la relativa famiglia di DGA, attraverso un processo di classificazione multi-classe.

Di seguito verranno introdotti i tre algoritmi utilizzati: Support-vector machine, Random Forest e MLP.

3.1.1 Support Vector Machine

Il *Support Vector Machine* è un metodo di apprendimento supervisionato utilizzato per problemi di classificazione e/o regressione. Il modello più semplice di SVM è utilizzato per la classificazione lineare e consiste nella rappresentazione degli esempi come punti nello spazio, mappati in modo tale che gli esempi appartenenti alle diverse classi (due, nel caso più frequente) siano

separati da uno spazio il più ampio possibile. I nuovi dati forniti in input sono mappati nello stesso spazio e la predizione della classe a cui appartengono viene fatta sulla base del lato in cui ricadono (Figura 5). Nel caso della classificazione multi-classe, solitamente si riduce il problema in più problemi di classificazione binaria. [6]

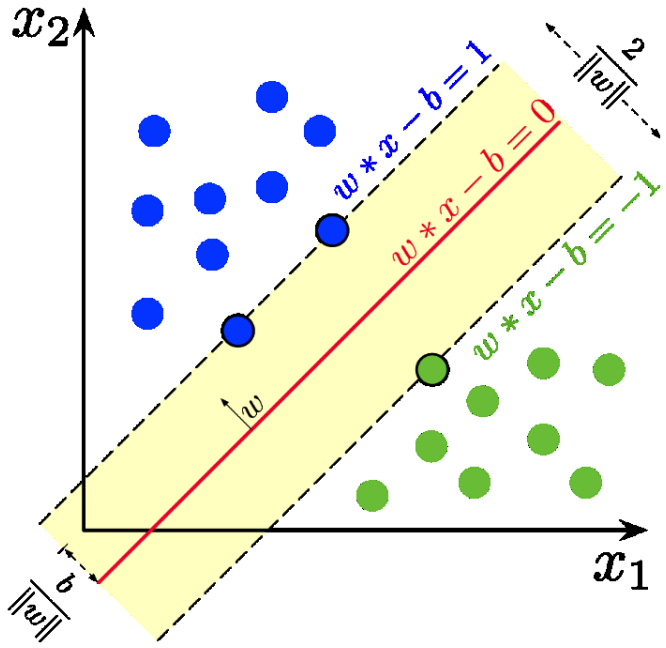

Figura 5 - SVM binario

3.1.2 Random Forest

Il *Random Forest* è un modello di classificazione basato sugli alberi decisionali. Gli alberi decisionali (*Decision Trees,* Figura 6) sono tecniche per la creazione di modelli predittivi, dove ogni variabile del modello è rappresentata da un *nodo*, le singole proprietà di ogni variabile sono identificate da un *arco* verso un nodo figlio e il valore predetto per la variabile, a partire dai valori delle proprietà, è rappresentato da una *foglia.* Il processo decisionale è rappresentato con un albero logico rovesciato dove ogni nodo è una funzione condizione. [7]

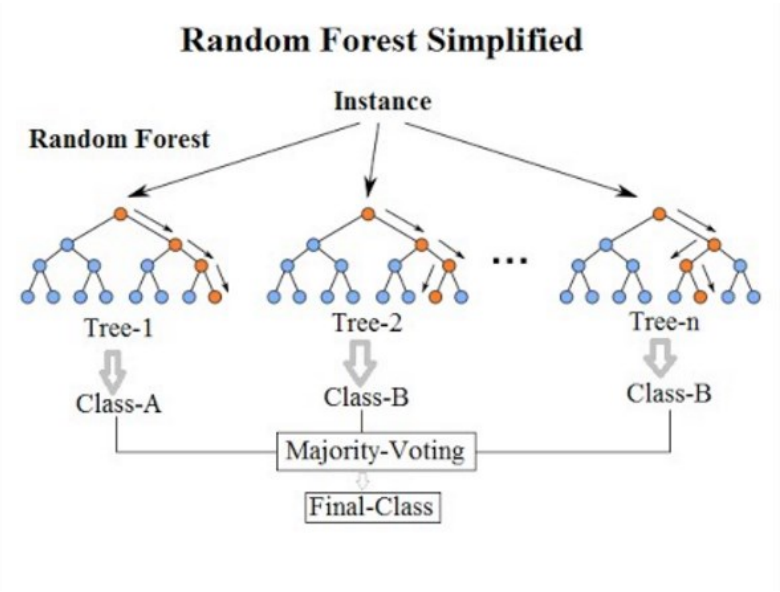

Figura 6 - Random Forest

3.1.3 Multi-Layer Perceptron

Multi-Layer Perceptron (MLP): è una classe di rete neurale artificiale ed è una tecnica di apprendimento supervisionato. Una MLP (Figura 8) è costituita da almeno tre livelli di nodi: un livello di input, un livello nascosto e un livello di output. Ad eccezione dei nodi di input, ogni nodo è un neurone che usa una funziona di attivazione non lineare. È in grado di distinguere dati che non sono separabili linearmente. [8]

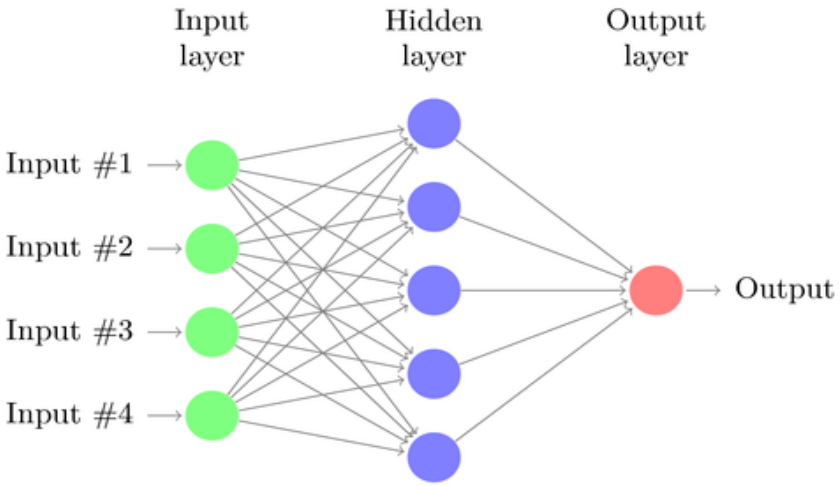

Figura 7 – MLP

3.2 Metriche di valutazione

Per valutare la bontà degli algoritmi utilizzati, si usano una serie di parametri.

In un problema di classificazione binaria, il caso più semplice, l'insieme dei dati si può classificare in due gruppi, solitamente in positivi (P) e negativi (N). I risultati ottenuti dalle previsioni del classificatore allora possono essere indicati con le diciture precedentemente enunciate. Gli esiti possibili sono quindi quattro:

- Vero Positivo (TP): a partire da un dato appartenente alla classe P, il classificatore predice il valore P;
- Falso Positivo (FP): a partire da un dato appartenente alla classe N, il classificatore predice il valore P;
- Vero Negativo (TN): a partire da un dato appartenente alla classe N, il classificatore predice il valore N;
- Falso Negativo (FN): a partire da un dato appartenente alla classe P, il classificatore predice il valore N.

Le metriche utilizzate sono:

- Accuracy: è il rapporto tra il numero di predizioni corrette e il numero totale di predizioni effettuate

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

- Errore:

$$Errore = 1 - Accuracy$$

- Precision: indica in che misura le predizioni ritenute positive sono positive

$$Precisoin = \frac{TP}{TP + FP}$$

- Recall: indica in che misura il classificatore riesce ad individuare tutti campioni positivi

$$Recall = \frac{TP}{TP + FN}$$

- F1-score: è la media armonica tra Precision e Recall

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

3.4 Feature

Le *feature* sono le caratteristiche dei dati da classificare che sono mappate negli attributi utilizzati dai classificatori per effettuare l'addestramento per creare il modello con il quale vengono realizzate le predizioni.

In questa tesi, che si basa su lavori sviluppati precedentemente [9], sono utilizzate come feature le caratteristiche lessicali dei nomi di dominio. In particolare, si usano due metriche: la *Kullback - Leibler divergence* e la *Jaccard Index*. Attraverso le feature si evidenziano le differenze tra i nomi di dominio legittimi e quelli creati dai DGA.

3.4.1 Kullback-Leibler Divergence

La Kullback-Leibler Divergence è una misura non simmetrica della differenza tra due distribuzioni di probabilità P e Q. In particolare, la divergenza di Kullback–Leibler di Q da P, indicata con $D_{\mathrm{KL}}(P \mid\mid Q)$, è la misura dell'informazione persa quando Q è usata per approssimare P.

Per due distribuzioni discrete P e Q, la divergenza KL di Q da P è definita come:

$$\sum_{i=1}^{n} P(\mathrm{i})\, log_2 \left(\frac{P(i)}{Q(i)}\right)$$

[10] con n che è il numero dei valori possibili delle variabili, la distribuzione di probabilità P rappresenta la distribuzione di test e Q è la distribuzione di base da cui viene calcolata la metrica [11].

Come distribuzioni di riferimento Q, nel presente lavoro, sono prese le distribuzioni degli *1-gram, 2-gram* e *3-gram* dei nomi di dominio legittimi (il cui insieme costituisce la *Whitelist)* e le distribuzioni degli *1-gram, 2-gram* e *3-gram* della lista dei nomi di dominio creati attraverso i DGA (il cui insieme costituisce la *Blacklist*).

Mediante la K-L Divergence si calcola la "distanza" tra la distribuzione degli n-grammi di ogni nome di dominio presente nel dataset e la distribuzione dei corrispondenti n-grammi della whitelist e della blacklist, andando così ad ottenere per ogni dominio sei features. Successivamente si calcolano valori della K-L Divergence, considerando le distribuzioni dei bigrammi e trigrammi di ogni singola famiglia di DGA delle 25 che compongono il dataset usato per gli esperimenti (Tabella 1).

KULLBACK-LEIBLER DIVERGENCE	
$\boldsymbol{D_{KL}(P\|\|B)_{1-gram}}$	$D_{KL}(P\|\|W)_{1-gram}$
$\boldsymbol{D_{KL}(P\|\|B)_{2-gram}}$	$D_{KL}(P\|\|W)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|B)_{3-gram}}$	$D_{KL}(P\|\|W)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Conficker)_{2-gram}}$	$D_{KL}(P\|\|Conficker)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Corebot)_{2-gram}}$	$D_{KL}(P\|\|Corebot)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Cryptolocker)_{2-gram}}$	$D_{KL}(P\|\|Cryptolocker)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Dircrypt)_{2-gram}}$	$D_{KL}(P\|\|Dircrypt)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Emotet)_{2-gram}}$	$D_{KL}(P\|\|Emotet)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Fobber)_{2-gram}}$	$D_{KL}(P\|\|Fobber)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Gozi)_{2-gram}}$	$D_{KL}(P\|\|Gozi)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Kraken)_{2-gram}}$	$D_{KL}(P\|\|Kraken)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Matsnu)_{2-gram}}$	$D_{KL}(P\|\|Matsnu)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Murofet)_{2-gram}}$	$D_{KL}(P\|\|Murofet)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Necurs)_{2-gram}}$	$D_{KL}(P\|\|Necurs)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Nymain)_{2-gram}}$	$D_{KL}(P\|\|Nymain)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Padcrypt)_{2-gram}}$	$D_{KL}(P\|\|Padcrypt)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Pushdo)_{2-gram}}$	$D_{KL}(P\|\|Pushdo)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Pykspa)_{2-gram}}$	$D_{KL}(P\|\|Pykspa)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Quadars)_{2-gram}}$	$D_{KL}(P\|\|Quadars)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Ramdo)_{2-gram}}$	$D_{KL}(P\|\|Ramdo)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Ramnit)_{2-gram}}$	$D_{KL}(P\|\|Ramnit)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Ranbyus)_{2-gram}}$	$D_{KL}(P\|\|Ranbyus)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Rovnix)_{2-gram}}$	$D_{KL}(P\|\|Rovnix)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Simda)_{2-gram}}$	$D_{KL}(P\|\|Simda)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Suppobox)_{2-gram}}$	$D_{KL}(P\|\|Suppobox)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Symmi)_{2-gram}}$	$D_{KL}(P\|\|Symmi)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Timba)_{2-gram}}$	$D_{KL}(P\|\|Timba)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Vawtrak)_{2-gram}}$	$D_{KL}(P\|\|Vawtrak)_{3-gram}$

Tabella 1 - Kullback-Leibler Divergence su Whitelist, Blacklist e le famiglie di DGA

3.4.2 Jaccard Index

La *Jaccard Index* misura la somiglianza tra due insiemi di elementi. È così definita:

$$JI = \frac{A \cap B}{A \cup B}$$

Con A e B insiemi che, nel nostro caso, rappresentano due set di n-gram di due nomi di dominio. [12]

Una volta estratte i *2-gram* e *3-gram* dei nomi di dominio viene calcolata la Jaccard Index sulle diverse distribuzioni (Tabella 2).

JACCARD INDEX	
$\boldsymbol{JI(P, Whitelist)_{2-gram}}$	$JI(P, Whitelist)_{3-gram}$
$\boldsymbol{JI(P, Conficker)_{2-gram}}$	$JI(P, Conficker)_{3-gram}$
$\boldsymbol{JI(P, Corebot)_{2-gram}}$	$JI(P, Corebot)_{3-gram}$
$\boldsymbol{JI(P, Cryptolocker)_{2-gram}}$	$JI(P, Cryptolocker)_{3-gram}$
$\boldsymbol{JI(P, Dircrypt)_{2-gram}}$	$JI(P, Dircrypt)_{3-gram}$
$\boldsymbol{JI(P, Emotet)_{2-gram}}$	$JI(P, Emotet)_{3-gram}$
$\boldsymbol{JI(P, Fobber)_{2-gram}}$	$JI(P, Fobber)_{3-gram}$
$\boldsymbol{JI(P, Gozi)_{2-gram}}$	$JI(P, Gozi)_{3-gram}$
$\boldsymbol{JI(P, Kraken)_{2-gram}}$	$JI(P, Kraken)_{3-gram}$
$\boldsymbol{JI(P, Matsnu)_{2-gram}}$	$JI(P, Matsnu)_{3-gram}$
$\boldsymbol{JI(P, Murofet)_{2-gram}}$	$JI(P, Murofet)_{3-gram}$
$\boldsymbol{JI(P, Necurs)_{2-gram}}$	$JI(P, Necurs)_{3-gram}$
$\boldsymbol{JI(P, Nymain)_{2-gram}}$	$JI(P, Nymain)_{3-gram}$
$\boldsymbol{JI(P, Padcrypt)_{2-gram}}$	$JI(P, Padcrypt)_{3-gram}$
$\boldsymbol{JI(P, Pushdo)_{2-gram}}$	$JI(P, Pushdo)_{3-gram}$
$\boldsymbol{JI(P, Pykspa)_{2-gram}}$	$JI(P, Pykspa)_{3-gram}$
$\boldsymbol{JI(P, Quadars)_{2-gram}}$	$JI(P, Quadars)_{3-gram}$
$\boldsymbol{JI(P, Ramdo)_{2-gram}}$	$JI(P, Ramdo)_{3-gram}$
$\boldsymbol{JI(P, Ramnit)_{2-gram}}$	$JI(P, Ramnit)_{3-gram}$
$\boldsymbol{JI(P, Ranbyus)_{2-gram}}$	$JI(P, Ranbyus)_{3-gram}$
$\boldsymbol{JI(P, Rovnix)_{2-gram}}$	$JI(P, Rovnix)_{3-gram}$
$\boldsymbol{JI(P, Simda)_{2-gram}}$	$JI(P, Simda)_{3-gram}$
$\boldsymbol{JI(P, Suppobox)_{2-gram}}$	$JI(P, Suppobox)_{3-gram}$
$\boldsymbol{JI(P, Symmi)_{2-gram}}$	$JI(P, Symmi)_{3-gram}$
$\boldsymbol{JI(P, Tinba)_{2-gram}}$	$JI(P, Tinba)_{3-gram}$
$\boldsymbol{JI(P, Vawtrak)_{2-gram}}$	$JI(P, Vawtrak)_{3-gram}$

Tabella 2 - Jaccard Index su Whitelist e le famiglie di DGA

4. Dataset

Di seguito viene descritta, basandosi su lavori di tesisti precedenti, la creazione del dataset utilizzato.

4.1 Creazione del dataset e bilanciamento dei dati

L'insieme dei nomi di dominio "benevoli", ovvero nomi di dominio associati a siti effettivamente presenti nel Web con i loro contenuti informativi, è denominato **whitelist.**

Per popolare la whitelist sono state utilizzate due liste ottenute dai siti *alexa.com* e *statvoo.com*.

Per la composizione della **blacklist**, ovvero l'insieme dei nomi di dominio generati dai DGA, è stato sfruttato l'archivio DGA di Netlab 360 (Tabella 3). Nonostante le famiglie presenti sul suddetto sito siano più di quaranta, ne sono state scelte un numero limitato. Infatti, vi era un forte sbilanciamento nel numero di nomi di dominio disponibili per famiglia di DGA. Nello specifico sono state mantenute ventidue famiglie di DGA, che risultano essere quelle con il

maggior numero di nomi o delle quali si conosce l'algoritmo di generazione, in modo da produrre i nomi di dominio necessari.

L'obiettivo prefissato è quello di ottenere un dataset bilanciato, con un numero di nomi di dominio, suddivisi equamente tra whitelist e blacklist, pari a 840.000 (Figura 8). Quindi essendo 420.000 nomi di dominio per la blacklist, ogni famiglia deve avere 15.000 nomi di dominio. A tal proposito, sono state aggiunte ulteriori sei famiglie, ad altre sedici sono stati aggiunti ulteriori nomi di dominio usando gli algoritmi di generazione e le dimensioni di altre sei sono state ridotte a 15000 elementi. Dunque, si ha un totale si 28 famiglie di DGA e 420.000 nomi di dominio "malevoli".

Si è cercato inoltre di bilanciare anche per tipologia di DGA: delle 28 famiglie prese in considerazione per la blacklist, 15 famiglie sono algoritmi tempo dipendenti-deterministici (TDD) e le restanti 13 sono algoritmi tempo indipendenti-deterministici (TID).

Il dataset viene suddiviso in due set chiamati **set A** e **set B**, dove il primo è il 80% del totale e il secondo è il rimanente 20%. Entrambi contengono un numero uguale di nomi di dominio malevoli e appartenenti alla whitelist. Il set A (Tabella 5) serve per la fase di addestramento del classificatore mentre il set B (Tabella 4) è utilizzato esclusivamente per la fase di test e contiene famiglie di DGA sconosciute all'algoritmo. Infatti, vi sono famiglie che il classificatore non ha mai incontrato in fase di addestramento.

Di 28 famiglie di DGA, il set A ne contiene 25 più la classe di nomi domini della whitelist, per un totale di 26 famiglie. Il set B è formato dalle 3 famiglie rimanenti. Per una migliore comprensione fare riferimento alle Figura 9 e 10.

NOME DGA	NUMEROSITA'	NOME DGA	NUMEROSITA'
ccleaner	1	cryptolocker	1.000
madmax	1	chinad	1.000
xshellghost	1	locky	1.150
blackhole	2	qadars	1.800
mirai	2	suppobox	2.369
tofsee	20	shifu	2.552
tinynuke	32	symmi	4.256
bedep	45	murofet	7.359
vidro	100	necurs	8.192
proslikefan	100	ranbyus	8.560
gspy	100	virut	9914
bamital	104	gameover	12.002
padcrypt	168	ramnit	17.851
tempedreve	204	simda	22.102
nymaim	255	pykspa	45.701
vawtrak	299	tinba	90.024
dircrypt	469	rovnix	179.995
fobber	600	emotet	244.795
matsnu	873	banjori	439.213
dyre	1.000	conficker	445.595
TOTALE 1.549.806			

Tabella 3 - Archivio DGA Netlab360

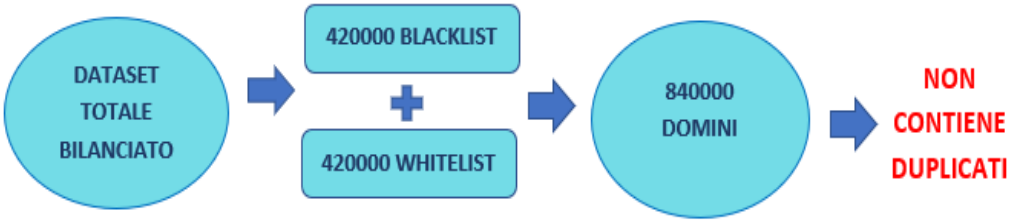

Figura 8 - Suddivisone del Dataset

Set B	
Classe	**Numero domini**
whitelist	45000
banjori	15000
chinad	15000
qakbot	15000

Tabella 4 - Set B

NOME DGA	NUMERO DOMINI	NOME DGA	NUMERO DOMINI
whitelist (alexa)	375000	padcrypt	15000
conficker	15000	pushdo	15000
corebot	15000	pykspa	15000
cryptolocker	15000	qadars	15000
dircrypt	15000	ramdo	15000
emotet	15000	ramnit	15000
fobber	15000	ranbyus	15000
gozi	15000	rovnix	15000
kraken	15000	simda	15000
matsn	15000	suppobox	15000
murofet	15000	symmi	15000
necurs	15000	tinba	15000
nymaim	15000	vawtrak	15000
banjori	15000	chinad	15000
qakbot	15000		

Tabella 5 - Dataset bilanciato

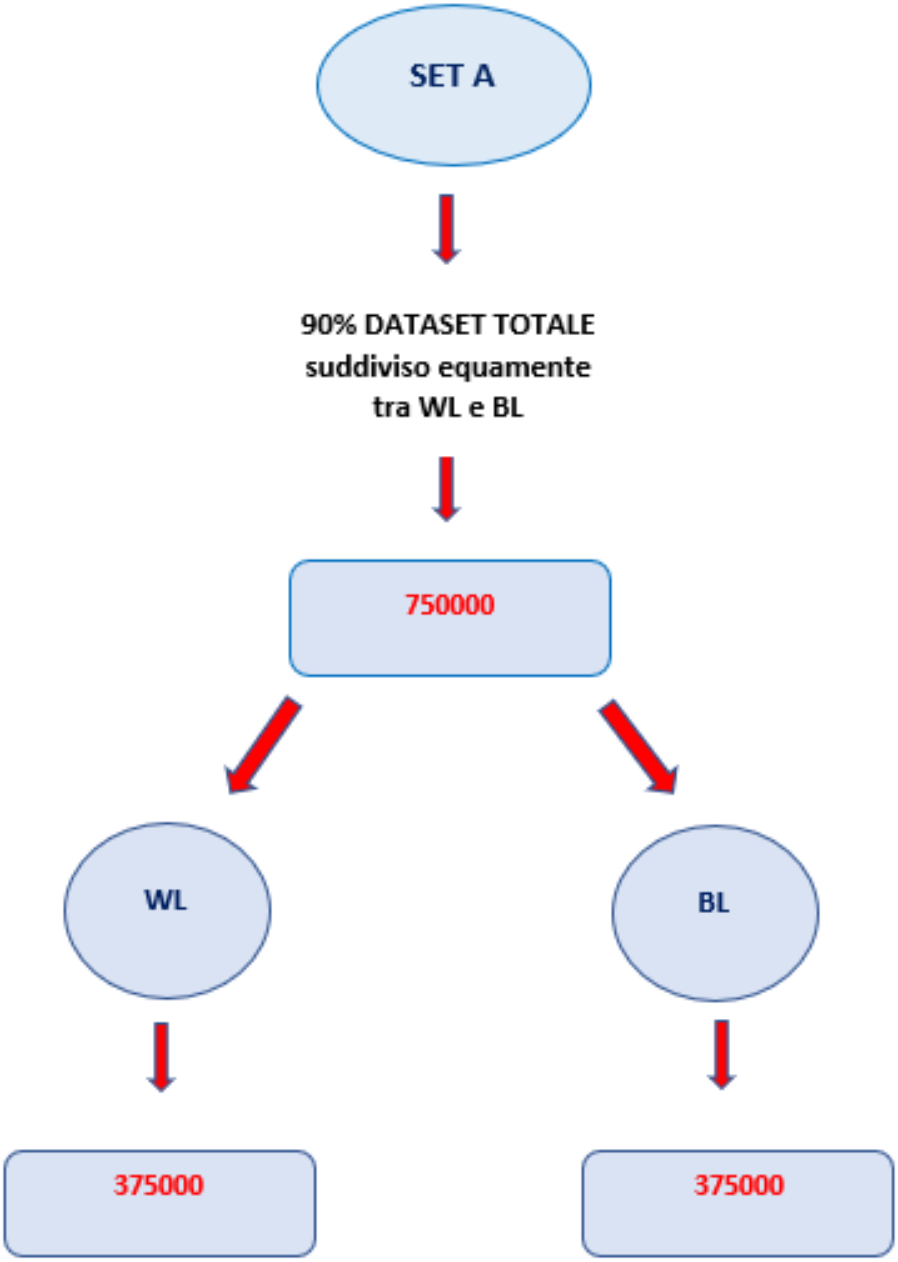

Figura 9 - Suddivisione Set A

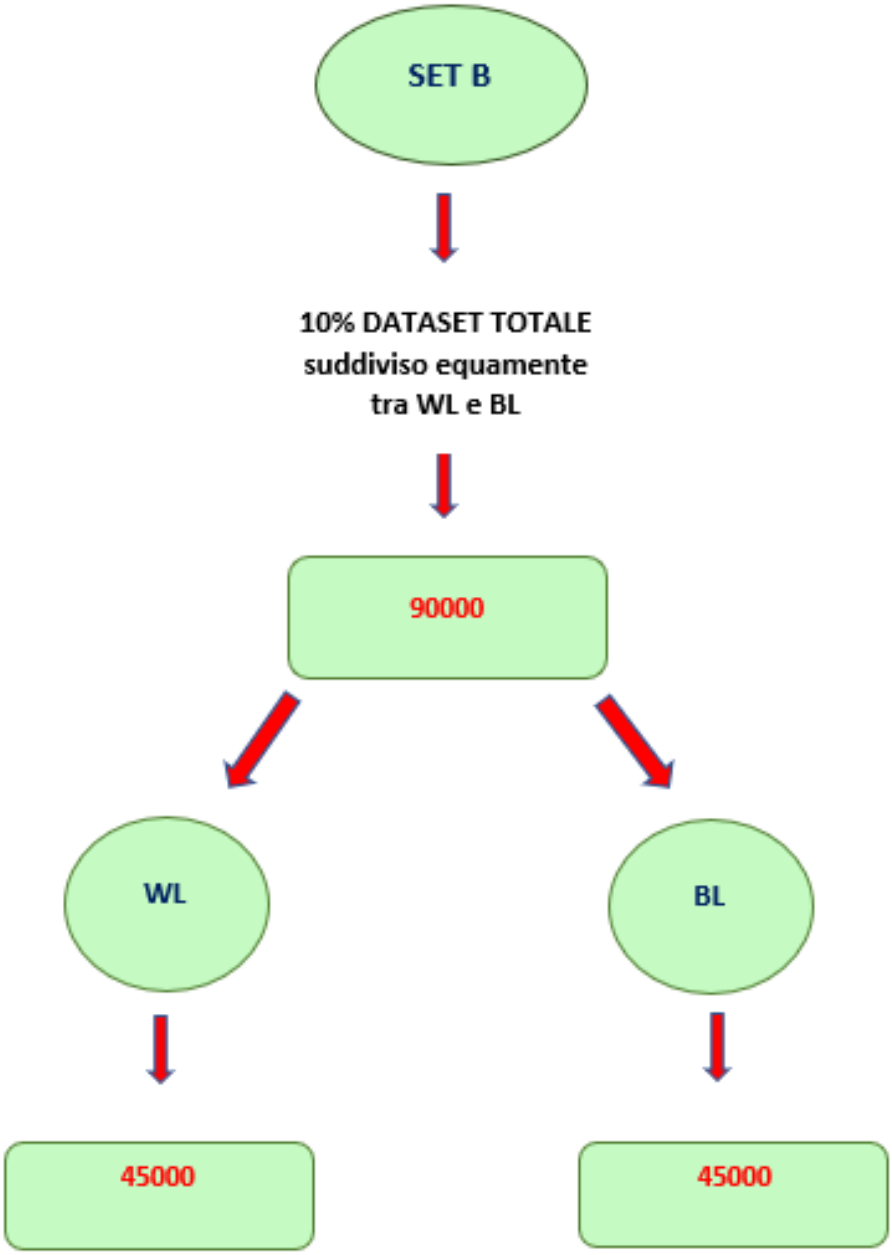

Figura 10 - Suddivisione Set B

4.2 Set di addestramento e set di test

Per quanto detto sugli algoritmi di machine learning con apprendimento supervisionato, si necessitano di due dataset, che derivano da quello principale: il dataset che viene usato per l'addestramento dell'algoritmo (***A-train***) e il dataset per testare il modello prodotto (**A-test**). Sono rispettivamente il 90% (675000 nomi di dominio) e il 10% (75000 nomi di dominio) del set A (Figura 11). Il numero di nomi di dominio malevoli risulta essere uguale a quello dei benevoli. Anche in questa suddivisione i dati risultano essere bilanciati, come osservabile nelle Tabelle 6 e 7.

Sia A-train che A-test hanno le stesse famiglie di DGA per poter addestrare il classificatore e valutarne l'efficacia su tipologie di nomi di dominio delle quali conosce già le famiglie di DGA che le hanno generate.

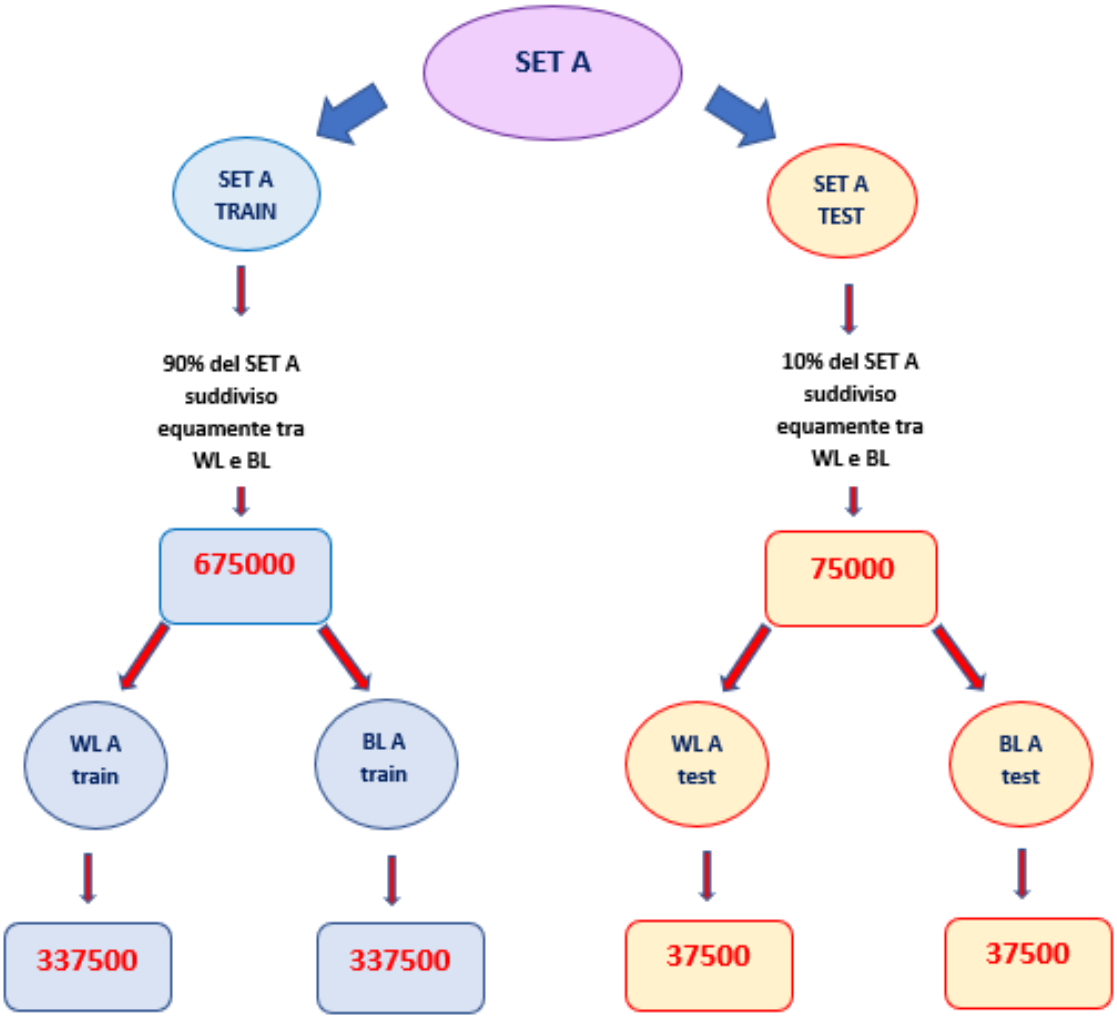

Figura 11 - Suddivisione Set A in A train e A test

A - TRAIN			
CLASSE	**NUMERO DOMINI**	**CLASSE**	**NUMERO DOMINI**
whitelist	337500	padcrypt	13500
conficker	13500	pushdo	13500
corebot	13500	pykspa	13500
cryptolocker	13500	qadars	13500
dircrypt	13500	ramdo	13500
emotet	13500	ramnit	13500
fobber	13500	ranbyus	13500
gozi	13500	rovnix	13500
kraken	13500	simda	13500
matsn	13500	suppobox	13500
murofet	13500	symmi	13500
necurs	13500	tinba	13500
nymaim	13500	vawtrak	13500
TOTALE 675000			

Tabella 6 - Composizione A train

A - TEST			
CLASSE	**NUMERO DOMINI**	**CLASSE**	**NUMERO DOMINI**
whitelist	37500	padcrypt	1500
conficker	1500	pushdo	1500
corebot	1500	pykspa	1500
cryptolocker	1500	qadars	1500
dircrypt	1500	ramdo	1500
emotet	1500	ramnit	1500
fobber	1500	ranbyus	1500
gozi	1500	rovnix	1500
kraken	1500	simda	1500
matsn	1500	suppobox	1500
murofet	1500	symmi	1500
necurs	1500	tinba	1500
nymaim	1500	vawtrak	1500
TOTALE 75000			

Tabella 7 - Composizione A test

4.3 Analisi dei nomi di dominio

Un nome di dominio è una sequenza gerarchica di stringhe separate da punti. Sono costruiti mediante le regole e le procedure del *Domain Name System* (DNS) e sono associati dal servizio DNS a IP (*Internet Protocol*). Un IP, che consnte di identificare in modo univoco un nodo della rete Internet, a differenza di un nome di dominio, risulta essere difficile da memorizzare poiché è una sequenza di numeri separati da punti.

I nomi di dominio sono organizzati come livelli subordinati al dominio radice del DNS (*DNS root domain*), che è senza nome. Si suddividono, considerando le stringhe che li compongono (separate da punti) da sinistra verso destra come segue:

1. *Top Level Domain* (TLD);
2. *Second Level Domain* (2LD);
3. *Third Level Domain* (3LD);
4. …;
5. Sub-dominio "*www*", non sempre presente.

Consideriamo il nome di dominio:

www.google.com

Partendo da sinistra si ha "*com*" che è il Top Level Domain, "*google*" è il Secondary Level Domain e "www" è il sub-dominio. Questo è solo un esempio. Infatti, non tutti i nomi di dominio sono composti da solo due livelli di dominio.

Nel dataset preso in esame, si è deciso di non considerare il sub-dominio poiché non risulta essere rilevante ai fini del lavoro svolto.

Dai lavori svolti precedentemente, si evince che l'alfabeto utilizzato per i nomi di dominio è costituito esclusivamente da numeri, lettere, punti, *underscore* e trattini:

"*0123456789abcdefghijklmnopqrstuvwxyz_-.*"

Analizzando la struttura e la distribuzione di *1-gram*, *2-gram* e *3-gram* all'interno della whitelist e della blacklist risulta che l'andamento della distribuzione dei caratteri alfanumerici (*1-gram*) è molto simile tra i nomi di dominio benevoli che malevoli. Ci si ritrova nella stessa situazione anche per i bigrammi, nonostante vi siano dei picchi su "co", "om", "ne", "et" e così via. Questi picchi sono dovuti alla presenza dei Top Level Domain. Infatti, analizzando la distribuzione dei trigrammi, si osserva chiaramente come i TLD sono responsabili di tali picchi (Figura 12).

Nei precedenti lavori sullo stesso tema, questo rumore provocato dai Top Level Domain ha portato a non prenderli in esame e a selezionare, per ciascun nome di dominio, solamente il Secondary Level Domain. [9]

In questa tesi è stata operata una scelta diversa. Basandosi sul lavoro di recupero dei TLD di un altro tesista [13], ad ogni nome di dominio è stato assegnato un indice numerico che identifica il Top Level Domain corrispondente e viene utilizzato come feature dal classificatore.

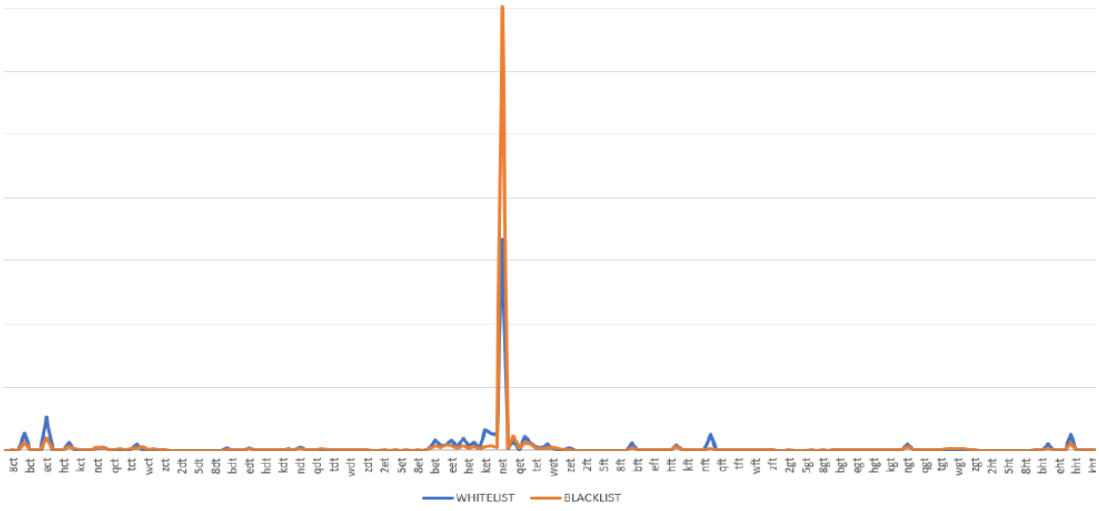

Figura 12 - Distribuzione 3-gram

4.4 Codice DGA

Per permettere ai classificatori utilizzati non solo di prevedere se il nome di dominio fornito è malevolo ma di prevederne anche la famiglia di appartenenza, bisogna considerare il problema di classificazione non binario ma multi-classe.

Poiché gli algoritmi presi in esame si basano sull'apprendimento supervisionato, in fase di addestramento necessitano di confrontare il risultato predetto con il valore vero. In questo modo possono correggere il modello che creano.

Fino ad ora, il problema è stato trattato in maniera tale che gli algoritmi generassero un modello che potesse predire se un nome di dominio fornito fosse benevolo o fosse stato generato da un DGA. Quindi in fase di addestramento, al classificatore veniva fornita una *label* che indicava, mediante un valore numerico, se il nome di dominio analizzato in fase di addestramento fosse benevolo ("0") o malevolo ("1").

Per poter addestrare i classificatori affinché predicano anche la famiglia di DGA a cui dovrebbe appartenere il nome di dominio, la label non assumerà solo due valori, ma ne può assumere 26. Infatti, è stato creato un dizionario che associa ad ogni classe di cui può far parte un nome di dominio un valore numerico, a partire dalla classe "white", associata allo "0", che indica i nomi benevoli.

Dunque, il modello creato dagli algoritmi fornisce in uscita un valore compreso tra 0 e 25.

4.4.1 Incompatibilità tra codici DGA e B-set

Utilizzando come risultato di confronto i codici dei DGA appena discussi, il risultato che il modello può produrre per ogni nome di dominio è un numero compreso tra 0 e 25. Ogni valore compreso in tale intervallo, escluso lo 0,

corrisponde ad una famiglia di DGA, in particolare quelle che il classificatore ha osservato in fase di addestramento.

Il B-set, però, è composto da (oltre che da quelli benevoli) nomi di dominio di famiglie di DGA mai osservate dal classificatore, ovvero:

- Banjori;
- Chinad;
- Qakbot.

Questo significa che nel dizionario dei codici dei DGA non vi è il codice relativo alle suddette famiglie. Oltre le problematiche implementative, risulterebbe impossibile classificare correttamente i nomi di dominio malevoli.

Per quanto detto, i modelli verranno testati solo sull'A-test.

5. Risultati

I risultati della fase di test sono stati ottenuti mediante l'utilizzo di algoritmi di classificazione con apprendimento supervisionato.

È stata utilizzata la libreria *open source* di machine learning. Gli algoritmi di apprendimento supervisionato utilizzati sono:

- SVM (Support Vector Machine);
- Random Forest (Decision Tree);
- MLP (Multi-layer Perceptron classifier - Rete neurale).

Sui risultati sperimentali ottenuti dai diversi classificatori viene effettuata anche una "validazione incrociata" (*Cross-validation*) a partire dal dataset di addestramento, l'A-train. La cross validation consiste nella suddivisione del dataset in *k* parti (*k-fold*) di uguale numerosità, nell'addestramento ripetuto k volte del classificatore con i contenuti di k-1 parti seguito da un test sulla parte non usata in addestramento. La media dei valori di performance ottenuti in ognuno dei k test costituisce una misura delle preformance del classificatore più attendibile di una singola sequenza addestramento-test. Così facendo, inoltre, si riducono i problemi tipoci dei processi di addestramento, come l'*overfitting* [14]. Negli esperimenti effettuati, si è utilizzata una cross validation con K=10 (*10-fold)*.

I test, invece, sono stati effettuati sul set A-test che comprende 75000 nomi di dominio. Come detto precedentemente, poiché si utilizzano i codici delle famiglie di DGA come label per il confronto dei risultati, non è possibili effettuare il test finale sul B-set.

Le feature utilizzate sono 105, la KL-divergence e la Jaccard index calcolate su bigrammi e trigrammi (Tabella 4) e l'indice del Top Level Domain.

FEATURES	
$\boldsymbol{D_{KL}(P\|\|W)_{2-gram}}$	$D_{KL}(P\|\|W)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Conficker)_{2-gram}}$	$D_{KL}(P\|\|Corebot)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Cryptolocker)_{2-gram}}$	$D_{KL}(P\|\|Dircrypt)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Emotet)_{2-gram}}$	$D_{KL}(P\|\|Fobber)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Gozi)_{2-gram}}$	$D_{KL}(P\|\|Kraken)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Matsnu)_{2-gram}}$	$D_{KL}(P\|\|Murofet)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Necurs)_{2-gram}}$	$D_{KL}(P\|\|Nymain)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Padcrypt)_{2-gram}}$	$D_{KL}(P\|\|Pushdo)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Pykspa)_{2-gram}}$	$D_{KL}(P\|\|Quadars)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Ramdo)_{2-gram}}$	$D_{KL}(P\|\|Ramnit)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Ranbyus)_{2-gram}}$	$D_{KL}(P\|\|Rovnix)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Simda)_{2-gram}}$	$D_{KL}(P\|\|Suppobox)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Symmi)_{2-gram}}$	$D_{KL}(P\|\|Timba)_{2-gram}$
$\boldsymbol{D_{KL}(P\|\|Vawtrak)_{2-gram}}$	$D_{KL}(P\|\|Conficker)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Corebot)_{3-gram}}$	$D_{KL}(P\|\|Cryptolocker)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Dircrypt)_{3-gram}}$	$D_{KL}(P\|\|Emotet)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Fobber)_{3-gram}}$	$D_{KL}(P\|\|Gozi)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Kraken)_{3-gram}}$	$D_{KL}(P\|\|Matsnu)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Murofet)_{3-gram}}$	$D_{KL}(P\|\|Necurs)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Nymain)_{3-gram}}$	$D_{KL}(P\|\|Padcrypt)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Pushdo)_{3-gram}}$	$D_{KL}(P\|\|Pykspa)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Quadars)_{3-gram}}$	$D_{KL}(P\|\|Ramdo)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Ramnit)_{3-gram}}$	$D_{KL}(P\|\|Ranbyus)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Rovnix)_{3-gram}}$	$D_{KL}(P\|\|Simda)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Suppobox)_{3-gram}}$	$D_{KL}(P\|\|Symmi)_{3-gram}$
$\boldsymbol{D_{KL}(P\|\|Timba)_{3-gram}}$	$D_{KL}(P\|\|Vawtrak)_{3-gram}$
$\boldsymbol{JI(P,Whitelist)_{2-gram}}$	$JI(P,Whitelist)_{3-gram}$
$\boldsymbol{JI(P,Conficker)_{2-gram}}$	$JI(P,Corebot)_{2-gram}$
$\boldsymbol{JI(P,Cryptolocker)_{2-gram}}$	$JI(P,Dircrypt)_{2-gram}$
$\boldsymbol{JI(P,Emotet)_{2-gram}}$	$JI(P,Fobber)_{2-gram}$

$\boldsymbol{JI(P, Gozi)_{2-gram}}$	$JI(P, Kraken)_{2-gram}$
$\boldsymbol{JI(P, Matsnu)_{2-gram}}$	$JI(P, Murofet)_{2-gram}$
$\boldsymbol{JI(P, Necurs)_{2-gram}}$	$JI(P, Nymain)_{2-gram}$
$\boldsymbol{JI(P, Padcrypt)_{2-gram}}$	$JI(P, Pushdo)_{2-gram}$
$\boldsymbol{JI(P, Pykspa)_{2-gram}}$	$JI(P, Quadars)_{2-gram}$
$\boldsymbol{JI(P, Ramdo)_{2-gram}}$	$JI(P, Ramnit)_{2-gram}$
$\boldsymbol{JI(P, Ranbyus)_{2-gram}}$	$JI(P, Rovnix)_{2-gram}$
$\boldsymbol{JI(P, Simda)_{2-gram}}$	$JI(P, Suppobox)_{2-gram}$
$\boldsymbol{JI(P, Symmi)_{2-gram}}$	$JI(P, Tinba)_{2-gram}$
$\boldsymbol{JI(P, Vawtrak)_{2-gram}}$	$JI(P, Conficker)_{3-gram}$
$\boldsymbol{JI(P, Corebot)_{3-gram}}$	$JI(P, Cryptolocker)_{3-gram}$
$\boldsymbol{JI(P, Dircrypt)_{3-gram}}$	$JI(P, Emotet)_{3-gram}$
$\boldsymbol{JI(P, Fobber)_{3-gram}}$	$JI(P, Gozi)_{3-gram}$
$\boldsymbol{JI(P, Kraken)_{3-gram}}$	$JI(P, Murofet)_{3-gram}$
$\boldsymbol{JI(P, Necurs)_{3-gram}}$	$JI(P, Matsnu)_{3-gram}$
$\boldsymbol{JI(P, Padcrypt)_{3-gram}}$	$JI(P, Nymain)_{3-gram}$
$\boldsymbol{JI(P, Pykspa)_{3-gram}}$	$JI(P, Pushdo)_{3-gram}$
$\boldsymbol{JI(P, Ramdo)_{3-gram}}$	$JI(P, Quadars)_{3-gram}$
$\boldsymbol{JI(P, Ranbyus)_{3-gram}}$	$JI(P, Ramnit)_{3-gram}$
$\boldsymbol{JI(P, Simda)_{3-gram}}$	$JI(P, Rovnix)_{3-gram}$
$\boldsymbol{JI(P, Symmi)_{3-gram}}$	$JI(P, Suppobox)_{3-gram}$
$\boldsymbol{JI(P, Vawtrak)_{3-gram}}$	$JI(P, Tinba)_{3-gram}$

Tabella 8 - Feature lessicali utilizzate

Saranno riportati, per effettuare un confronto, i risultati del classificatore multi-classe ricavati dal dataset senza indice del TLD e i risultati del classificatore binario sul dataset con TLD. Dalla fase di addestramento ci si è resi conto che la rete neurale (MLP) è quella che produce i risultati migliori. Pertanto, i risultati esposti sono quelli del classificatore MLP.

Confrontando il classificatore multi-classe col classificatore binario (sul dataset con TLD) (Tabelle 9, 11 e 13) si osserva un prevedibile calo di prestazioni poiché la classificazione multiclasse risulta più complessa. Per esempio, osservando la Tabella 13, la *Macro-Precision* del classificatore binario risulta essere maggiore di quello del multi-classe di circa del 6%. Questo significa che nel caso binario saranno presenti meno falsi positivi, quindi meno siti "benevoli" verranno scambiati come malevoli. Nonostante ciò, i risultati nel caso della classificazione multi-classe, sono promettenti.

Confrontando i risultati multi-classe ottenuti dal dataset con e senza TLD (Tabelle 9, 10, 11 e 12) si può osservare come sia stata una scelta vincente quella di includere il Top Level Domain. Infatti, vi è un generale miglioramento della Accuracy, della Precision e delle altre caratteristiche.

Nello specifico, possiamo osserva come nel caso senza TLD, la classe con cui il classificatore ha avuto più difficoltà è stata Ramni con 0.69884 di Precision, mentre il valore di Recall più basso è in presenza della classe Necurs con 0.65259 e un'Accuracy di 0.653. Mentre con il dataset con il Top Level Domain si nota

come il classificatore abbia avuto problemi con la famiglia Dircrypt con una Precision di 0.73759, mentre per quanto riguarda Recall e Accuracy i risultati peggiori si ottengono con Necurs, rispettivamente 0.77533 e 0.775. Si noti come anche la Accuracy complessiva (*overall*) nel caso con TLD risulta essere migliore, seppur non di molto. Questo significa che mediamente, nel caso con TLD, il modello classifica correttamente le famiglie più di frequente.

CLASSI	PRECISION	RECALL	F1-SCORE	ACCURACY
White	0.99866	0.99653	0.99760	0.997
Conficker	0.96412	0.94933	0.95667	0.997
Corebot	0.99933	1.00000	0.99967	1.0
Cryptolocker	0.85180	0.82000	0.83560	0.82
Dircrypt	0.73759	0.90133	0.81128	0.901
Emotet	0.99359	0.93067	0.96110	0.931
Fobber	0.86534	0.92533	0.89433	0.925
Gozi	0.98935	0.99133	0.99034	0.991
Kraken	0.91352	0.83800	0.87413	0.838
Matsnu	0.98033	0.99667	0.98843	0.997
Murofet	0.91198	0.89800	0.90494	0.898
Necurs	0.83549	0.77533	0.80429	0.775
Nymaim	0.95163	0.97067	0.96106	0.971
Padcrypt	0.99933	1.00000	0.99967	1.0
Pushdo	0.99800	1.00000	0.99900	1.0
Pykspa	0.96498	0.95533	0.96013	0.955
Qadars	0.99933	0.99933	0.99933	0.999
Ramdo	0.99933	1.00000	0.99967	1.0
Ramnit	0.83912	0.78933	0.81347	0.789
Ranbyus	0.78753	0.86733	0.82551	0.867
Rovnix	0.98876	0.99733	0.99303	0.997
Simda	0.99463	0.98733	0.99097	0.987
Suppobox	0.98807	0.99400	0.99103	0.994
Symmi	0.97828	0.99067	0.98443	0.991
Tinba	0.82937	0.79067	0.80956	0.791
Vawtrak	0.99933	0.99933	0.99933	0.999

Tabella 9 - Risultati ottenuti dal classificatore multi-classe sull'A-test con Top Level Domain

CLASSI	PRECISION	RECALL	F1-SCORE	ACCURACY
White	0.99480	0.99843	0.99661	0.998
Conficker	0.86451	0.86963	0.86706	0.87
Corebot	1.00000	1.00000	1.00000	1.0
Cryptolocker	0.87154	0.76889	0.81700	0.769
Dircrypt	0.79578	0.78222	0.78894	0.782
Emotet	0.84027	0.92741	0.88169	0.927
Fobber	0.87712	0.92000	0.89805	0.92
Gozi	0.96945	0.96370	0.96657	0.964
Kraken	0.80182	0.78519	0.79341	0.785
Matsnu	0.99544	0.96963	0.98236	0.97
Murofet	0.87700	0.93481	0.90498	0.935
Necurs	0.84468	0.65259	0.73631	0.653
Nymaim	0.92981	0.87333	0.90069	0.873
Padcrypt	0.99926	0.99926	0.99926	0.999
Pushdo	0.96312	0.98667	0.97475	0.987
Pykspa	0.93884	0.90963	0.92400	0.91
Qadars	0.99926	1.00000	0.99963	1.0
Ramdo	0.99778	1.00000	0.99889	1.0
Ramnit	0.69884	0.76148	0.72882	0.761
Ranbyus	0.92968	0.84222	0.88379	0.842
Rovnix	0.99926	0.99481	0.99703	0.995
Simda	0.98798	0.97407	0.98098	0.974
Suppobox	0.94225	0.95481	0.94849	0.955
Symmi	0.93022	0.95778	0.94380	0.958
Tinba	0.84770	0.96889	0.90425	0.969
Vawtrak	0.99704	0.99778	0.99741	0.998

Tabella 10 - Risultati ottenuti dal classificatore multi-classe sull'A-test senza Top Level Domain

	PRECISION	RECALL	F1-SCORE
Micro avg	0.96561	0.96561	0.96561
Macro avg	0.93688	0.93707	0.93633
Weighted avg	0.96653	0.96561	0.96573
Overall accuracy	0.96560		

Tabella 11 - Risultati medi ottenuti dal classificatore multi-classe sull'A-test con Top Level Domain

	PRECISION	RECALL	F1-SCORE
Micro avg	0.95510	0.95510	0.95510
Macro avg	0.91898	0.91512	0.91595
Weighted avg	0.95537	0.95510	0.95466
Overall accuracy	0.95510		

Tabella 12 - Risultati medi ottenuti dal classificatore multi-classe sull'A-test senza Top Level Domain

CLASSI	PRECISION	RECALL	F1-SCORE
White	0.99962	0.99080	0.99519
Black	0.99088	0.99963	0.99523
Macro avg	0.99525	0.99521	0.99521

Tabella 13 - Risultati classificatore binario sul set A-test con TLD

6. Conclusione e sviluppi futuri

I risultati ottenuti nella classificazione multi-classe sono molto promettenti, nonostante l'impossibilità di usare i modelli creati con un dataset contente famiglie di DGA non osservate in fase di addestramento. Si evince anche che la Rete Neurale (MLP) è quella che produce il modello migliore in questo contesto, a parità di condizioni, rispetto alla SVM e all'Albero Decisionale (Random Forest). Inoltre, risulta vincente la scelta di utilizzare i Top Level Domain.

L'approccio risulta essere un ottimo esempio di utilizzo delle tecniche di machine learning di apprendimento supervisionato e mostra la loro versatilità in ambito di *cyber security*. Si noti anche come sia fondamentale la raccolta dei dati, il loro utilizzo e le feature che si decidono di utilizzare.

Un passo successivo potrebbe essere quello di rivalutare la suddivisione dei nomi di dominio, per esempio considerando l'intero indirizzo DNS e non solo il Secondary Level Domain.

Potrebbe essere interessare utilizzare il modello binario addestrato sul dataset compreso di Top Level Domain su un dataset formato da nomi di dominio prelevati dal traffico registrato su macchine infette e non infette. In questo modo si può osservare l'effettiva validità di quanto ottenuto.

Bibliografia

[1] *Rapporto Clusit 2020*; Clusit -Associazione Italiana per la Sicurezza Informatica; https://clusit.it/rapporto-clusit/ (ultima consultazione 05/2020)

[2] Wikipedia; *Ransomware*; https://it.wikipedia.org/wiki/Ransomware

[3] Rosita Rijtano; *Trojan Horse: cos'è, come funziona e come rimuoverlo ed evitarlo*; https://www.cybersecurity360.it/nuove-minacce/trojan-i-malware-spia-come-funzionano-e-come-difendersi/ (ultima consultazione 05/2020)

[4] Rosita Rijtano; *Attacco DDoS (Distributed Denial of Service): Cos'è, come fare, come difendersi*; https://www.cybersecurity360.it/nuove-minacce/ddos-cosa-sono-questi-attacchi-hacker-e-come-stanno-evolvendo/ (ultima consultazione 05/2020)

[5] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, Elmar Gerhards-Padilla; *A Comprehensive Measurement Study of Domain Generating Malware*; https://www.usenix.org/system/files/conference/usenixsecurity16/sec16_paper_plohmann.pdf;2020

[6] Wikipedia; *Support Vector Machine*; https://en.wikipedia.org/wiki/Support_vector_machine#Multiclass_SVM (ultima consultazione 05/2020)

[7] Wikipedia; Random Forest https://it.wikipedia.org/wiki/Foresta_casuale; (ultima consultazione 05/2020)

[8] Wikipedia; *Multilayer Perceptron*; https://en.wikipedia.org/wiki/Multilayer_perceptron (ultima consultazione 05/2020)

[9] Federica Mambella; *Uso di features lessicali nell'analisi di nomi di dominio per il rilevamento di malware*; 2018

[10] Wikipedia; *Divergenza di Kullback-Leibler*; https://it.wikipedia.org/wiki/Divergenza_di_Kullback-Leibler (ultima consultazione 05/2020)

[11] Sandeep Yadav, Ashwath K.K. Reddy and A.L. Narashima Reddy, Supranamaya Ranjan; *Detecting Algorithmically Generated Malicious Domain Names;* November 2010

[12] Wikipedia; *Jaccatd Index*; https://it.wikipedia.org/wiki/Indice_di_Jaccard (ultima consultazione 05/2020)

[13] Simone Di Saverio; Rilevazione di Malware tramite analisi di richieste DNS basata su deep learning; 2019

[14] Wikipedia; *Convalida incrociata*; https://it.wikipedia.org/wiki/Convalida_incrociata; (ultima consultazione 05/2020)

Ringraziamenti

Inizio con il ringraziare i miei relatori che mi hanno condotto alla fine di questo viaggio, il professore Alessandro Cucchiarelli e il professore Christian Morbidoni. Li ringrazio per la loro incredibile disponibilità e dedizione.

Ringrazio con tutto me stesso mia madre e mio padre che mi hanno sempre supportato e spronato in questo percorso. Senza di loro non nemmeno sarei qui. Ringrazio anche mio fratello Emilio, per la fiducia che ha avuto in me.

Un ringraziamento speciale va alla mia fidanzata, Giusi. Con lei ho condiviso gioie e dolori, successi e fallimenti. Queste parole non sono sufficienti per esprimere quanto sia stata importante per me durante questi anni e quanto mi ha aiutato.

Ringrazio i miei amici e coinquilini Daniele, Leonardo e Marco. Hanno saputo supportarmi sempre nei modi più originali.

Sono grato anche di aver conosciuto Giacomo e Lorenzo, mi hanno mostrato come le vere amicizie perdurano anche dopo aver intrapreso strade diverse.

Ringrazio i miei amici e colleghi Angelo, Alessandro, Gianluca e Simone, sempre pronti ad aiutarmi e con i quali sono pronto a lavorare nuovamente insieme.

Grazie anche a tutte le altre persone che ho conosciuto durante questo incredibile viaggio.