UNIVERSITÀ
DEGLI STUDI
DI PALERMO

NDS
NETWORKING AND DISTRIBUTED SYSTEMS
RESEARCH GROUP

PROGETTAZIONE DI UN SISTEMA MULTI-CLASSIFICATORE PER IL RILEVAMENTO DI APPLICAZIONI ANDROID MALEVOLE

Tesi di Laurea Magistrale in Ingegneria Informatica

Antonio Santonocito

Relatore: Prof. Alessandra De Paola

PROGETTAZIONE DI UN SISTEMA MULTI-CLASSIFICATORE PER IL RILEVAMENTO DI APPLICAZIONI ANDROID MALEVOLE

TESI DI LAUREA DI

ANTONIO SANTONOCITO

RELATORE

PROF. ALESSANDRA DE PAOLA

SOMMARIO

Al giorno d'oggi la diffusione di software malevoli risulta in costante crescita e mette a repentaglio la confidenzialità, l'integrità e l'accessibilità dei sistemi informatici utilizzati nei più svariativi domini. Gli sviluppatori di malware sono alla continua ricerca di canali e vulnerabilità da sfruttare per diffondere i propri malware, ricercando nuove tecniche di occultamento che riescano ad aggirare le difese di un sistema, cambiando le caratteristiche significative dei malware così da non essere rilevati.

La maggior parte dei rilevatori di malware Android utilizzano modelli di Machine Learning addestrati su campioni appartenenti a ristretti intervalli temporali, rendendoli non inclini alla rilevazione di nuove tendenze in tema malware su un ampio periodo di tempo. I modelli utilizzati soffrono del *concept drift*, fenomeno per cui le proprietà statistiche delle informazioni cambiano nel tempo in un modo non prevedibile.

In questo lavoro di tesi viene proposto un sistema che sfrutta una combinazione di modelli multi-classificatore, in un processo di adattamento e addestramento continuo che consente di avere alte prestazioni in un lungo periodo di tempo. Il sistema utilizza un'analisi ibrida ottenuta dalla combinazione di due modelli multi-classificatore, uno addestrato su caratteristiche statiche ed uno su quelle dinamiche. Le valutazioni sperimentali effettuate hanno consentito di individuare i parametri ed i metodi migliori per la costruzione di un sistema di rilevazione che riesca a gestire il *concept drift*.

SOMMARIO

1. INTRODUZIONE

Il rapporto Clusit di ottobre 2022 [1], descrive la situazione attuale in materia cybercrime. Nel primo semestre del 2022 individua 1.141 cyber attacchi noti, registrando un aumento del +8.4% rispetto al primo semestre del 2021. Questo conferma la tendenza negativa del 2021, anno che era stato nominato il *peggiore di sempre* in tema di minacce informatiche e crescita del numero degli attacchi. Partendo dal primo semestre del 2018 i numeri degli attacchi registrati è aumentato del 53%, passando da 745 a 1.141 mentre la media mensile è aumentata da 171 del 2021 a 190 del 2022 come mostrato in Figura 1 ed in Figura 2.

Sempre nel primo semestre del 2022 analizzato si è aggiunto il conflitto in corso tra Russia e Ucraina, che ha visto l'utilizzo di offensive cibernetiche da parte dei contendenti ed in generale da tutti i principali attori, le azioni messe in atto sono state sia a supporto di attività di *cyber-intelligence* che di *cyber-warfare*.

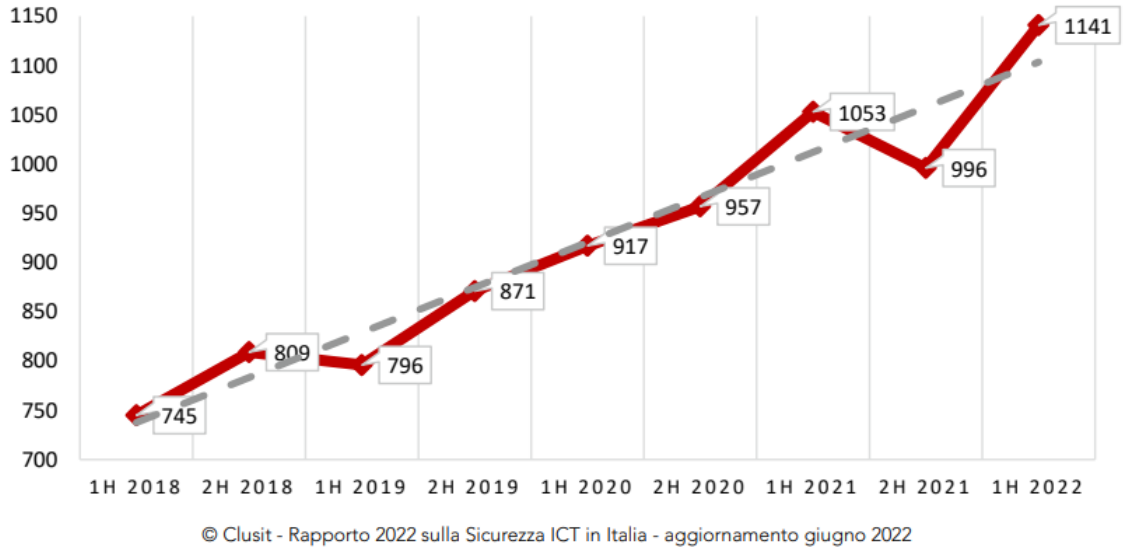

Figura 1 Attacchi per semestre 1H 2018 - 1H 2022

I malware colpiscono più facilmente gli utenti poco consapevoli del mezzo informatico che stanno utilizzando, questi utenti vengono solitamente colpiti attraverso il proprio smartphone, dispositivo di comunicazione più diffuso. Secondo una statistica di GSMA [2], organizzazione che rappresenta gli interessi degli operatori di rete mobile in tutto il mondo, il 53% della popolazione mondiale utilizza lo smartphone ed è connessa alla rete. Smartphone e tablet stanno diventando sempre più l'obiettivo dei cyber-criminali

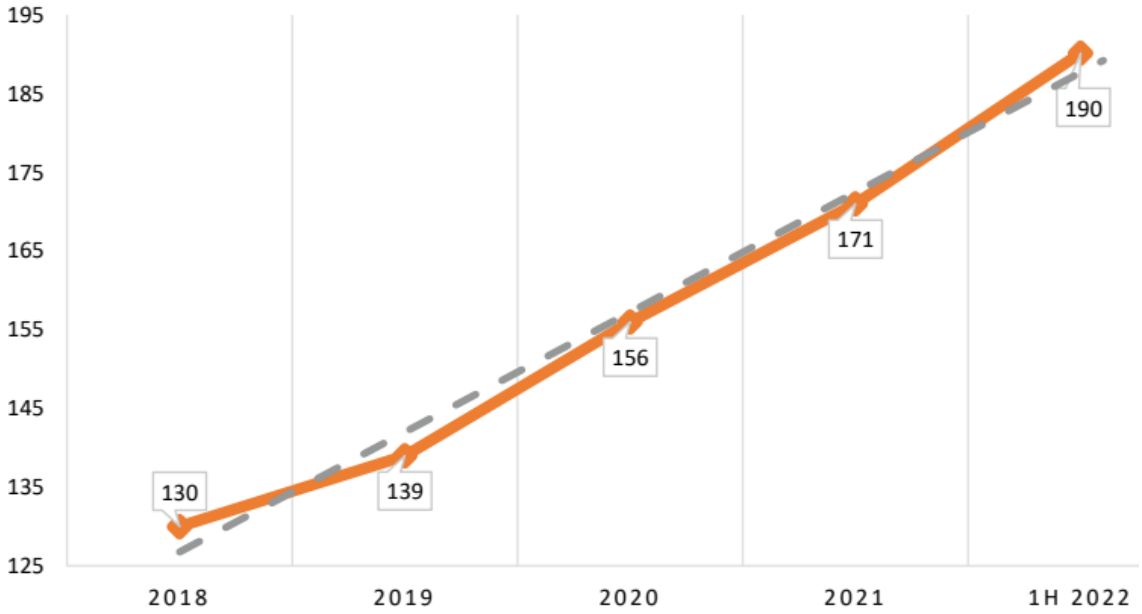

Figura 2 Media mensile degli attacchi 2018 - 1H 2022

grazie alla politica BYOD (*bring your own device*, "porta il tuo dispositivo"), adottata da molte aziende, che permette all'utente di avere accesso alle reti aziendali utilizzando i dispositivi propri. La diffusione di questi dispositivi è accompagnata dalla diffusione di malware mobile, codice malevolo con l'obiettivo di compromettere i dispositivi mobili quali smartphone e tablet.

Le tipologie di malware mobile più conosciute secondo Kaspersky [3] sono:

- Malware Bancario: i cyber-criminali mirano ad attaccare gli utenti che preferiscono effettuare trasferimenti di denaro o pagare le bollette attraverso il loro dispositivo mobile. Il mezzo più utilizzato per l'attacco è quello di ingannare l'utente nell'installazione di Trojan, applicazioni apparentemente innocue ma contenenti codice malevolo.
- Ransomware mobile: malware che corrompe il dispositivo compromettendo l'integrità e l'accessibilità dei dati dell'utente, come documenti, foto e video, criptando le informazioni e chiedendo il pagamento di un riscatto.
- Spyware mobile: lo spyware, nascosto come una normale applicazione, monitora le attività della vittima, registra la sua posizione ed estrae informazioni sensibili quali e-mail e passwords degli account dell'utente. Generalmente lo spyware viene installato all'interno di un pacchetto contenente altre applicazioni apparentemente innocue, raccogliendo informazioni in incognito. Lo spyware è

un malware subdolo, infatti è possibile che non si riesca a notare fino a che le prestazioni del dispositivo non vengono compromesse.

- Adware mobile: gli autori di adware, guadagnano attraverso il numero di clic e di download che le loro pubblicità ricevono. Questi malware infettano il dispositivo, obbligando l'utente a scaricare specifici adware e permettendo agli attaccanti di reindirizzare l'utente su siti pubblicitari.

La diffusione di software malevoli aumenta in modo allarmante, il loro obiettivo è quello di compromettere la confidenzialità, l'integrità e l'accessibilità di un sistema informatico. I malware si classificano in base al modo in cui si diffondono e alle azioni che compiono sul sistema infetto. Con rilevazione dei malware, si intende il processo di analisi del contenuto di un software per determinare se le sue intenzioni sono malevole o legittime.

I primi malware erano molto semplici, non utilizzavano complessi metodi per occultarsi e consistevano di un solo processo, potevano essere facilmente rilevati attraverso dei metodi statici basati sul confronto della struttura e delle caratteristiche di malware già noti. La nuova generazione di malware è più complessa, distruttiva e difficile da rilevare rispetto ai loro predecessori, questo perché possono lavorare ai livelli più bassi del sistema, utilizzando processi già esistenti o nuovi e sfruttando tecniche di occultamento tali da renderli persistenti nel sistema. Le tecniche di occultamento comunemente [4,5] utilizzate sono:

- *Cifratura*: La prima tecnica che i malware utilizzano per non essere rilevati da antivirus basati sulla scansione del codice è l'utilizzo della cifratura. In questi casi il corpo del software malevolo è composto da una parte del codice in chiaro ed una parte cifrata, la parte di codice in chiaro contiene il decifratore che una volta eseguito recupera il codice cifrato ed esegue le istruzioni malevole ottenute. Ad ogni infezione viene utilizzata una chiave differente, così da mantenere la parte cifrata non rilevabile da uno scanner basato su firme di malware noti.
- *Oligomorfismo e Polimorfismo*: nei metodi oligomorfici viene utilizzata una chiave differente per la cifratura e la decifratura del payload, in modo da rendere più difficile la rilevazione del codice malevolo. I metodi polimorfici oltre ad usare diverse chiavi per la cifratura e la decifratura, aggiungono al payload da cifrare

del codice senza alcuno scopo, o effettuano una cifratura a strati così da rendere la loro rilevazione il più complessa possibile.

- *Metamorfismo*: I malware che utilizzano il metamorfismo non utilizzano la cifratura. Utilizzano diverse tecniche di offuscamento per duplicarsi e cambiare dinamicamente la struttura del codice delle nuove generazioni cambiando di fatto aspetto ma mantenendo lo stesso obiettivo.
- *Protezione del codice*: tecniche che interferiscono con i software di analisi, effettuando dei cambiamenti sul sistema stesso e mantengono il codice malevolo nascosto.

La rilevazione di malware è un processo composto da una prima fase di analisi del software, una seconda fase di estrazione delle caratteristiche ed infine una fase di classificazione.

L'analisi dei malware è il processo necessario a determinare le funzionalità del software e capire meglio come questo lavora ed interagisce con l'elaboratore o con altri programmi. L'analisi avviene principalmente in due modalità, una statica e una dinamica. L'analisi statica esamina il software prima che questo venga eseguito, mettendo in evidenza la struttura del codice, le funzioni utilizzate, i permessi richiesti per la sua esecuzione e tutto ciò che si può estrarre dal solo codice sorgente. L'analisi dinamica esamina il software durante la sua esecuzione alla ricerca di comportamenti malevoli, ad esempio, analizzando il traffico di rete generato in esecuzione.

L'acquisizione delle caratteristiche statiche impiega poco tempo e risorse, ma l'utilizzo delle tecniche di offuscamento descritte prima e le tecniche dinamiche di elaborazione del codice sono degli ostacoli importanti dell'analisi statica. L'analisi dinamica, d'altra parte, pur avendo dei vantaggi sull'analisi statica, soffre di tutte quelle tecniche di offuscamento che modificano il comportamento del software in esecuzione ogni volta che viene rilevato che l'esecuzione si sta svolgendo all'interno di un sandbox.

L'obiettivo di questo lavoro di tesi è la costruzione di un sistema che sfrutti i vantaggi delle due tipologie di analisi e che tenga conto del cambiamento delle caratteristiche significative dei malware nel tempo. Questo fenomeno, noto come *concept drift*, dipende dal fatto che le proprietà statistiche delle informazioni possono cambiare nel

tempo in un modo non prevedibile, a detrimento delle capacità di rilevamento del sistema.

I prossimi capitoli sono così descritti:

- Nel capitolo 2, vengono descritti gli approcci più utilizzati per il rilevamento di malware, approfondendo il campo delle applicazioni Android e le tecniche comunemente usate per la rilevazione di Malware Android, approfondendo gli algoritmi di machine learning più utilizzati.
- Nel capitolo 3, vengono approfonditi i sistemi multi-classificatore, il loro funzionamento e le principali scelte di progettazione per la loro costruzione, approfondendo il fenomeno del *concept drift* tenuto in considerazione per la costruzione del sistema proposto.
- Nel capitolo 4, viene descritto il sistema proposto, descrivendone struttura, scelte progettuali e algoritmo di addestramento.
- Nel capitolo 5, viene descritto il data set utilizzato e le valutazioni sperimentali effettuate che hanno guidato le scelte progettuali fatte.
- Nel capitolo 6, sono contenuti i risultati finali ottenuti e i possibili miglioramenti applicabili al sistema per il suo mantenimento.

2. STATO DELL'ARTE

L'analisi di un software è necessaria a conoscere il suo contenuto e comportamento, e consente di determinare le funzionalità del software sotto esame e di rilevare i pattern di interazione con i diversi dispositivi e con altri programmi, nonché le azioni che esso esegue sui dati del sistema.

Generalmente le analisi utilizzate sono due, una statica e una dinamica, l'analisi statica analizza codice e struttura del software senza eseguirlo, mentre quella dinamica analizza il suo comportamento in fase di esecuzione. Durante l'analisi vengono utilizzate tecniche di data mining per estrarre le caratteristiche più significative del software. Per la fase di classificazione vengono utilizzati diversi algoritmi di machine learning con lo scopo di realizzare un sistema in grado di riconoscere la natura di un software sotto esame a partire dai dati ottenuti dalle fasi precedenti.

Risulta necessario, prima di approfondire il campo delle applicazioni Android, fare un approfondimento su due approcci comunemente conosciuti per la rilevazione di software malevoli e che vengono riutilizzati in tecniche più complesse.

Le prime tecniche per la rilevazione di malware sono state quelle basate sulla firma, una caratteristica del software che identifica unicamente sia il malware che la struttura del programma. Metodi di rilevazione basati sulla firma sono ampiamente diffusi, poiché sono metodi veloci ed efficienti per le rilevazioni di malware noti, ma possono essere facilmente aggirati tramite le tecniche di offuscamento descritte in precedenza. I programmi da analizzare vengono elaborati da un software di generazione di firme e queste vengono salvate in un database. Per la classificazione del programma in maligno o benevolo, viene confrontata la sua firma con quelle presenti nel database e in base a questo confronto il programma viene etichettato come malware o meno. Per costruire un generatore di firme performante è necessario che le chiavi

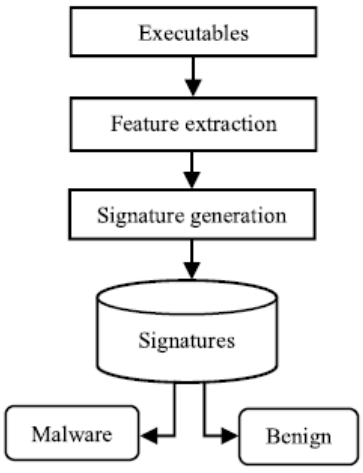

Figura 3 Schema della rilevazione di malware Signature-Based [5]

generate siano più corte possibile, così da caratterizzare più malware possibili e devono essere resistenti a tecniche di offuscamento e impacchettamento.

MalHunter, il sistema di rilevazione proposto da Borojerdi e Abadi [6], è un metodo che genera le firme basandosi sul comportamento polimorfico dei malware. *MalHunter* genera inizialmente un insieme di sequenze di comportamenti ottenuti da campioni di malware polimorfici, ognuno dei quali rappresenta un comportamento caratteristico, generando dei gruppi di malware con sequenze di comportamenti simili e salvandoli in un database. Per la fase di rilevazione, vengono analizzati i comportamenti del malware e confrontati con le sequenze precedentemente generate e salvate in un database.

Un'altra tecnica di rilevazione di malware è quella basata sul comportamento dello stesso. Questa tecnica prevede una prima fase di monitoraggio del software per determinare il suo intento, in modo che, anche se utilizzasse tecniche di offuscamento, sarebbe sempre possibile riconoscere il suo obiettivo che è rimasto immutato. Alcuni malware riescono a cambiare il proprio comportamento se ritengono di essere monitorati, riuscendo ad eludere la scansione. Il comportamento di un software viene ottenuto attraverso l'analisi del software su un sandbox, monitorando le chiamate di sistema, monitorando le modifiche che effettua sui files, monitorando le sue attività di rete e confrontando lo stato dei registri prima e dopo la sua esecuzione. I comportamenti elencati sono utilizzati come caratteristiche del software, queste vengono utilizzate per la costruzione di dataset utilizzati per l'addestramento di algoritmi di Machine Learning che si occuperanno di classificare i software in fase di test.

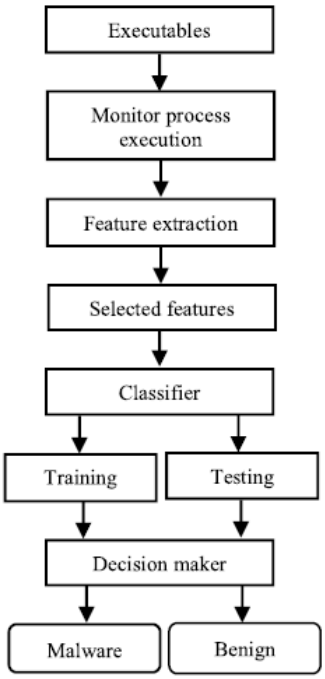

Figura 4 Schema della rilevazione di malware Behavior-Based [5]

Il framework *Bounded Feature Space Behavior Modeling* (BOFM) [7] utilizza le interazioni tra il software e le risorse critiche del sistema come caratteristiche. Le chiamate a sistema vengono usate come comportamenti e caratteristiche di alto livello, utilizzando queste caratteristiche per l'addestramento di algoritmi di ML, applicandoli

poi su nuovi campioni per determinarne il comportamento benevolo o malevolo del software sotto esame.

Il metodo di rilevazione basato sulla firma, precedentemente descritto, è veloce ed efficace nel rilevare software malevoli già noti. Tuttavia, non riesce a rilevare malware di nuova generazione ed è indifeso rispetto alle tecniche di offuscamento e polimorfismo che questi utilizzano.

La tecnica di rilevazione basata sul comportamento del software in fase di esecuzione permette di rilevare intenti malevoli anche se la struttura o le firme del software cambiano. Una sua debolezza è l'alto tasso di falsi positivi, questo è dovuto alla similarità di alcuni comportamenti presenti sia in campioni malevoli che in campioni legittimi.

Ogni metodo di rilevazione ha i suoi vantaggi e ottiene risultati migliori con determinati dataset rispetto ad altri. L'approccio più promettente è quello basato sulla combinazione di modelli e algoritmi di *machine learning* per la classificazione di software malevolo. La combinazione di diversi algoritmi di machine learning si ottiene tramite l'addestramento d'insieme, tramite il quale si riesce ad ottenere prestazioni migliori in fase di generalizzazione rispetto a quelle ottenute dal singolo addestramento. L'apprendimento d'insieme è stato utilizzato per la costruzione dei due rilevatori di malware Android che compongono il sistema ibrido proposto in questo lavoro di tesi e verrà ripreso e descritto successivamente.

2.1 APPLICAZIONI ANDROID

Il Sistema Android è basato sul kernel Linux, il sistema operativo Android utilizza una collezione di componenti indipendenti per la costruzione della propria architettura gerarchica. La classica architettura stratificata è rappresentata in figura ed è composta dal basso verso l'alto da: kernel Linux, livello di astrazione dell'hardware, librerie native C++/C e l'ambiente di Runtime Android, framework Java API ed un ultimo livello per le applicazioni. La parte riservata al kernel si trova alla fine dello stack mentre quella riservata all'utente si trova in cima al sistema Android, kernel e livello utente sono collegati tra loro attraverso chiamate di sistema. Le applicazioni del livello utente sono generalmente scritte in C++ o Java.

Figura 5 Architettura Android [9]

Il sistema Android utilizza le caratteristiche di sicurezza offerte dal kernel Linux. Linux è un sistema operativo multiutente in grado di isolare le risorse di un utente da quelle di un altro utente, ed allo stesso modo ne isola i processi. In un sistema così regolato ogni utente non può accedere ai files di un altro utente ed ogni processo è eseguito con l'identità dell'utente che lo ha lanciato chiamato *UserID*. Il sistema Android utilizza lo stesso principio di sicurezza, ma essendo Android originariamente sviluppato per smartphones, che sono ritenuti dei dispositivi personali, non esisteva il bisogno di registrare differenti utenti nello stesso sistema. Gli *UserID* in sistemi Android vengono utilizzati per distinguere le diverse applicazioni anziché gli utenti e questa è la base del meccanismo di sandbox Android [8].

Android assegna automaticamente un *UserID*, chiamato anche *AppID*, unico ad ogni applicazione sia in fase di installazione che in fase di esecuzione, inoltre gli viene

assegnata una porzione di memoria dedicata, su cui ha i permessi di scrittura e lettura. Queste applicazioni vengono isolate, anche tramite *sandbox*, sia a livello di esecuzione di suoi processi sia a livello di gestione dei files. Sembrerebbe che le applicazioni Android siano molto limitate e che possano accedere solamente ai propri files, per raggiungere funzionalità complesse ed interessanti le applicazioni Android possono richiedere dei diritti di accesso aggiuntivi. Questi diritti vengono chiamati permessi e possono essere relativi alla gestione dei dispositivi hardware, accesso alla rete, files o servizi del sistema. Un'applicazione definisce le richieste di accesso nel file *AndroidManifest.xml.* In fase di installazione, Android esamina le richieste di accesso dell'applicazione e decide se rilasciare o meno i permessi richiesti. Per richieste più delicate, come ad esempio le richieste di accesso ad informazioni sull'utente, viene richiesta esplicitamente la conferma dell'utente per rilasciare i permessi all'applicazione.

Su PC i malware si diffondono principalmente sfruttando le vulnerabilità del web browser, sui dispositivi mobili invece gli sviluppatori di malware sfruttano le vulnerabilità del sistema operativo *iOS* e *Android* del dispositivo. Come difesa, i sistemi operativi iOS, utilizzano un sistema di protezione chiamato "*Walled Garden*". Il metodo di protezione dei sistemi *Apple* crea un ambiente isolato per eseguire le applicazioni su smartphone, così che solo le applicazioni che, dopo l'analisi, risultano benevole possono entrare nel *marketplace Apple*. Il *marketplace Android* non ha meccanismi di difesa del genere avvantaggiando gli sviluppatori di malware ad ingannare gli utenti ad installare le proprie applicazioni malevole.

2.2 TECNICHE PER LA RILEVAZIONE DI MALWARE ANDROID

In questa sezione verranno descritte le principali tecniche di analisi e rilevazione delle caratteristiche e comportamenti malevoli di applicazioni Android seguendo la distinzione fatta in [10], tra analisi locale e analisi del traffico di rete.

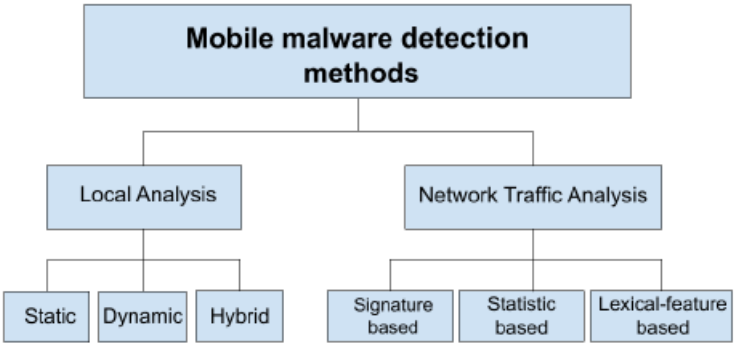

Figura 6 Metodi per la rilevazione di malware Android [10]

ANALISI LOCALE

Per analisi locale si intendono tutti quei metodi statici, dinamici e ibridi che si basano sul codice sorgente, i permessi richiesti ed il comportamento dell'applicazione senza che questa abbia accesso alla rete.

L'analisi statica esamina l'applicazione prima della sua esecuzione, ricercando sequenze di codice riconducibili a comportamenti malevoli o ricercando firme ritenute pericolose tra i files che la compongono. Un Android Package Kit (APK) è un archivio in cui risiede il codice dell'applicazione, le risorse che utilizza ed altre informazioni cruciali al suo funzionamento. Una preventiva analisi statica dei files contenuti nell'APK può mettere in luce gli intenti malevoli dell'applicazione, un APK tipicamente contiene sette componenti rilevanti:

1. *Assets*: cartella opzionale che contiene le risorse utilizzate dall'applicazione.
2. *Lib:* cartella opzionale che contiene codice già compilato relativo a librerie native.
3. *META*-INF: questa cartella contiene i files responsabili dell'integrità e della sicurezza dell'applicazione. Vengono memorizzati qui i metadati dell'applicazione, eventuali certificati digitali, una lista delle risorse, i digest SHA-1 di queste e la firma dell'APK.

4. *Res*: contiene risorse non compilate, come le animazioni, i suoni, le lingue ecc.
5. *AndroidManifest.xml*: file principale di ogni Applicazione Android, qui sono salvate le informazioni vitali come l'identificativo univoco dell'applicazione, i permessi richiesti, la versione, le librerie utilizzate e la descrizione di diverse componenti relative a servizi, attività e trasmissioni necessarie al funzionamento dell'applicazione.
6. *Classes.dex*: il file .dex contiene i meta-dati relativi ai metodi, classi, tipi e prototipi utilizzati dall'eseguibile, nell'ultima sua parte contiene l'intero codice.
7. *Resources.arsc*: contiene l'elenco delle risorse dell'applicazione in formato tabellare.

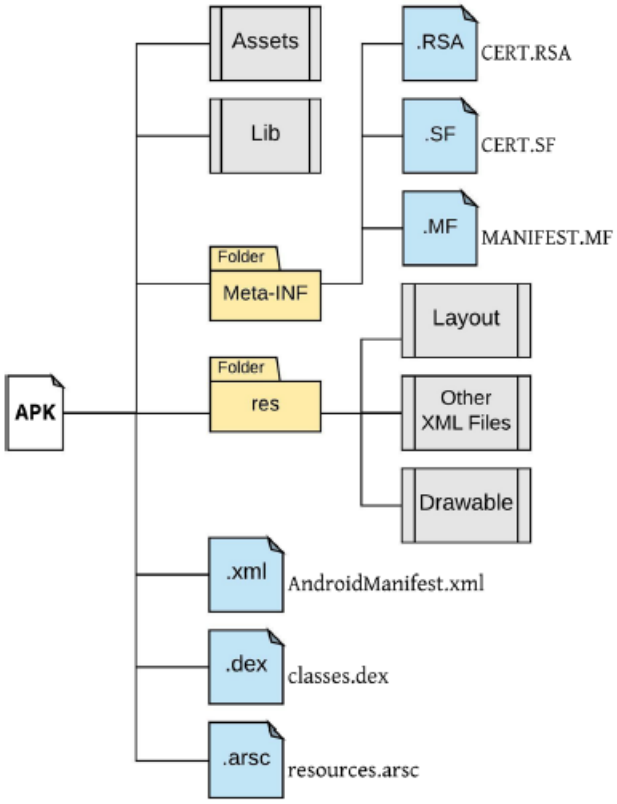

Figura 7 Struttura di una Android Package Kit [16]

Ad esempio, il modello PermPair [11] costruisce e confronta i grafi di campioni benevoli e malevoli, attraverso l'estrazione di coppie di permessi richiesti dall'applicazione ottenuti attraverso il file AndroidManifest.xml descritto prima. Per la rilevazione, vengono confrontati i grafi costruiti attraverso le coppie di permessi richiesti dall'applicazione sospetta con i grafi precedentemente costruiti ed in base al loro confronto determinare se l'applicazione è benevola o malevola.

I metodi dinamici utilizzano il comportamento in fase di esecuzione delle applicazioni per determinare se questa è malevola o benevola. Le caratteristiche dinamiche vengono estratte eseguendo le applicazioni in ambienti reali o ambienti emulati, come l'analisi in *sandbox*, estraendo le caratteristiche comportamentali come caratteristiche dinamiche. I *sandbox* sono utilizzati per avere un ambiente simulato dove analizzare il comportamento dell'applicazione in fase di esecuzione senza eseguirla effettivamente su uno smartphone. Ad esempio, il rilevatore di malware [12] di WT Lin e JY Pan, emula le interazioni dell'utente finale con le applicazioni e registra le risposte e le azioni eseguite dall'applicazione, il metodo di analisi descritto è rappresentato in Figura 8. L'utilizzo di questi strumenti può incentivare i malware che nascondono il proprio comportamento ad esternare la loro vera natura, migliorando l'efficacia dell'analisi.

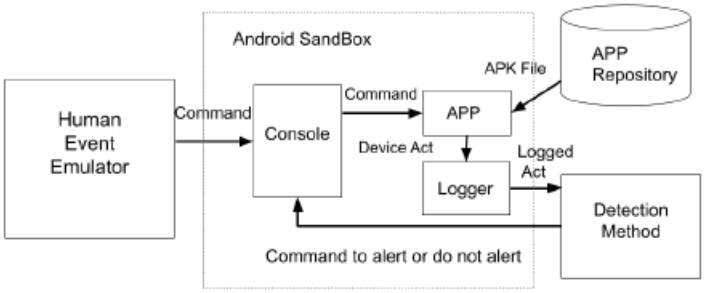

Figura 8 Rilevatore di malware Android via sandbox [12]

I metodi basati sull'analisi statica delle applicazioni non possono rilevare le applicazioni che cambiano il proprio codice in fase di esecuzione, mentre i metodi basati sull'analisi dinamica non posso rilevare i malware che modificano il proprio comportamento se analizzati. I metodi ibridi cercano di minimizzare gli svantaggi dei due metodi e massimizzarne i loro vantaggi combinando i due approcci, come ad esempio il sistema proposto in questo lavoro di tesi. Il sistema proposto si basa su due modelli, un primo sistema multi-classificatore è addestrato sulle sole caratteristiche statiche, permessi richiesti dall'applicazione raccolti durante la sua analisi statica. Quando la decisione presa dal primo sistema non è affidabile l'applicazione viene analizzata dinamicamente e la decisione finale verrà presa dal sistema multi-classificatore addestrato sulle sole caratteristiche dinamiche.

La maggior parte delle applicazioni malevole utilizzano la rete come mezzo per trasmettere i dati sensibili ottenuti. L'analisi del traffico di rete ha come obiettivo quello di tracciare le attività malevole dell'applicazione, le tecniche di rilevazione basate sul comportamento e sul traffico di rete generato in fase di esecuzione, sono due degli approcci più utilizzati per la rilevazione di malware in tempo reale.

Dall'analisi del traffico di rete si possono estrarre caratteristiche significative in diversi modi, l'approccio basato sulle firme punta ad estrarre sequenze di traffico sospetto per confrontarle con flussi di rete malevoli per determinare se il traffico analizzato è malevolo o legittimo. Ad esempio, il modello chiamato FlowIntent di H. Fu et al. [13] riesce a rilevare flussi di dati che non sono necessari al funzionamento dell'applicazione, stabilendo l'obiettivo ed il contesto di ogni trasmissione. Il framework è composto da due moduli: AppInspector e TrafficAnalyzer. Il primo modulo si occupa di definire il contesto delle trasmissioni sensibili così da evidenziare le trasmissioni non funzionali. Per trasmissioni sensibili si intendono le trasmissioni che riguardano dati ottenuti da risorse protette. Le trasmissioni sensibili non funzionali sono tutti i flussi di rete che non sono necessari all'insieme delle funzionalità dell'applicazione indicate tra i file dell'applicazione in esame. Il secondo modulo, TrafficAnalyzer, processa il traffico catturato dal primo modulo e addestra dei classificatori a rilevare se un traffico di rete è funzionale o non funzionale.

Alcuni metodi che utilizzano l'analisi del traffico di rete per rilevare i malware sono basati sulle caratteristiche lessicali del flusso di rete, analizzando il testo e la semantica delle trasmissioni. Solitamente viene analizzata la semantica del flusso di rete HTTP e degli URLs, come il sistema ibrido proposto da S. Kandukuru et al.[14], un modello di rilevazione basato sui permessi richiesti dall'applicazione e sull'analisi del traffico. L'analisi del traffico è incentrata sui pacchetti HTTP, evidenziando come i malware Android trasmettono dati sensibili attraverso semplici messaggi di HTTP POST e HTTP GET. La rilevazione avviene attraverso la ricerca di parole chiave nei messaggi scambiati, come ad esempio informazioni relative al sistema operativo in uso, numeri di serie del dispositivo, dettagli sull'operatore telefonico, informazioni sulla posizione del

dispositivo, e-mail e altre informazioni sensibili. Il numero di parole chiave trovate nel traffico di rete è direttamente proporzionale alla diffusione di informazioni sensibili. Le *Botnets* su dispositivi mobili vengono gestiti attraverso server di comando e controllo dei dispositivi in remoto come rappresentato in Figura 9, il *payload* dei messaggi scambiati contengono l'indirizzo IP cifrato del server oppure direttamente in chiaro. L'analisi del testo dei pacchetti di rete può essere significativa nel rilevare traffico malevolo verso indirizzi sospetti.

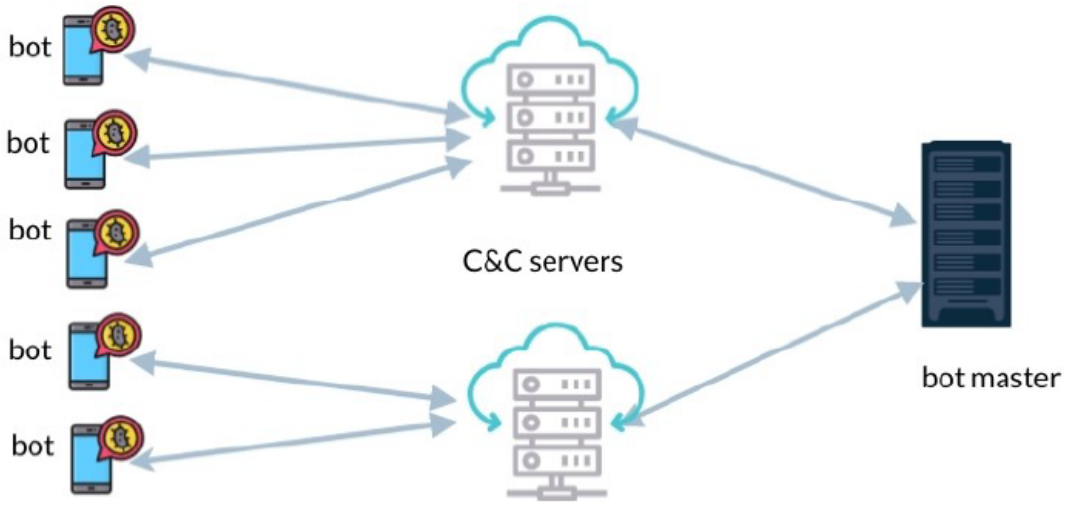

Figura 9 Flusso di rete per il comando ed il controllo remoto di bot smartphone. [10]

L'analisi del traffico di rete passa anche attraverso l'utilizzo delle informazioni contenute nell'intestazione dei pacchetti trasmessi per la rilevazione di malware. I metodi di rilevazione che analizzano le informazioni contenute nell'intestazione dei pacchetti, utilizzano come caratteristiche significative, ad esempio, la dimensione del pacchetto o la durata della trasmissione per addestrare algoritmi di machine learning per rilevare trasmissioni di rete malevole. Un metodo di rilevazione che utilizza l'intestazione dei pacchetti trasmessi è quello proposto da S. Wang et al. [15], il metodo da loro proposto analizza principalmente le richieste HTTP ed i flussi TCP per determinare se un'applicazione è malevola. Vengono analizzate generalmente le intestazioni delle richieste HTTP, nei casi in cui il traffico è cifrato vengono utilizzati i flussi TCP per estrarre caratteristiche significative del comportamento della trasmissione.

2.3 MODELLI E ALGORITMI DI MACHINE LEARNING UTILIZZATI NELLA RILEVAZIONE DI MALWARE ANDROID

Il *machine learning* è una branca dell'intelligenza artificiale che si occupa di sviluppare diverse tecniche per automatizzare la generazione di predizioni basate sulle osservazioni passate. Gli approcci di machine learning possono essere classificati in base al metodo di addestramento utilizzato, questi si dividono in: addestramento supervisionato, addestramento non supervisionato, addestramento semi-supervisionato e addestramento rinforzato.

L'addestramento supervisionato fa uso generalmente di dataset con campioni già etichettati per l'addestramento di modelli predittivi applicabili su diversi tipi di problemi, ad esempio, per problemi di classificazione o regressione. I modelli supervisionati, in fase di addestramento, conoscono per ogni input il rispettivo output che viene utilizzato per addestrare l'algoritmo a riconoscere le regole del modello in esame.
L'addestramento non supervisionato non ha bisogno del dataset con campioni già etichettati per il loro addestramento. L'obiettivo di questi modelli di machine learning è quello di individuare la struttura interna o la distribuzione delle caratteristiche dei campioni nel dataset, generalmente sono utilizzati in problemi relativi al clustering ed alla riduzione della dimensione delle caratteristiche.
L'addestramento semi-supervisionato combina gli elementi dei due approcci di addestramento descritti prima, utilizzando sia campioni etichettati sia campioni non etichettati. I modelli semi-supervisionati in genere utilizzano un modello addestrato sui campioni etichettati per determinare le etichette dei campioni non etichettati. Questo modello viene utilizzato in quei casi in cui la raccolta di campioni etichettati è problematica e la loro disponibilità è ridotta.
L'addestramento rinforzato è caratterizzato da un processo ciclico di predizione e valutazione, i campioni sono utilizzati direttamente come input del modello ed i risultati ottenuti condizionano il cambiamento dei parametri del modello ad ogni ciclo.

I modelli e gli algoritmi di machine learning più utilizzati sono rappresentati in Figura 10, gli algoritmi e modelli rappresentati sono il risultato dell'analisi dello stato attuale della ricerca sui modelli di machine learning utilizzati nella rilevazione di malware Android

effettuata da Liu, Kaijun, et al [17]. Altri metodi comunemente utilizzati utilizzano le reti neurali, l'addestramento di insieme e l'addestramento online. L'addestramento di insieme, ad esempio, è un metodo che combina diversi modelli di machine learning per costruire un unico sistema predittivo, l'addestramento d'insieme non è un metodo di addestramento ma un metodo per combinare diversi modelli predittivi.

Learning Method	Machine Learning Model or Algorithm	
Supervised Learning	1. Decision Trees 2. Naive Bayesian 3. Linear Model (1) Linear Regression: Ordinary Least Squares Regression (2) Logistic Regression (3) Linear Discriminate Analysis (LDA) 4. K-Nearest Neighbor (KNN) 5. Support Vector Machine (SVM)	1. Neural Network (NN) and Deep Learning (DL) 2. Hidden Markov Model (HMM) 3. Transfer Learning 4. Ensemble Learning: Bagging, Boosting, Random Forest 5. Online Learning
Unsupervised Learning	1. Clustering Algorithms: K-means 2. Principal Component Analysis (PCA) 3. Singular Value Decomposition (SVD) 4. Independent Component Analysis (ICA) 5. A-priori Algorithm 6. Expectation-Maximization (EM)	
Semi-supervised Learning	1. Semi-supervised Learning with Nuclear Norm Regularization (SSL-NNR) 2. Graph Inference Learning (GIL) 3. Laplacian SVM	
Reinforcement Learning	1. Q-Learning 2. Deep Q-Learning Network (DQN) 3. Temporal Difference Learning	

Figura 10 Algoritmi di machine learning comunemente utilizzati in problemi di Classificazione.[17]

Gli algoritmi ed i modelli più utilizzati per la classificazione di malware Android che verranno approfonditi sono: Alberi Decisionali, Random Forest, Support Vector Machine e Naive Bayes.

Come rappresentato in Figura 10, l'albero decisionale (DT) è uno degli algoritmi di tipo supervisionato comunemente utilizzato in problemi di classificazione di malware Android. Gli obiettivi di un classificatore DT sono: la classificazione corretta di più campioni di addestramento possibili, avere una capacità di generalizzazione tale da avere buone prestazioni anche su campioni non visti in fase di addestramento, aggiornare facilmente il modello con l'arrivo di nuovi campioni di addestramento ed avere una struttura il più semplice possibile.

Il metodo di classificazione è semplice, l'albero decisionale è costruito attraverso la suddivisione dei campioni del dataset di addestramento in base a regole sui valori delle loro caratteristiche e ottenendo dei sottoinsiemi sempre più piccoli fino ad arrivare ad una situazione in cui gli elementi del sottoinsieme appartengono tutti alla stessa classe, non è possibile suddividere ulteriormente i campioni o si è raggiunta la profondità massima. I nodi foglia ottenuti vengono etichettati con la classe di maggioranza relativa ai campioni contenuti nel nodo foglia, un nuovo campione viene classificato inserendolo nel nodo radice ed etichettandolo con la classe relativa al nodo foglia in cui il campione termina in base al percorso che intraprende nell'albero.

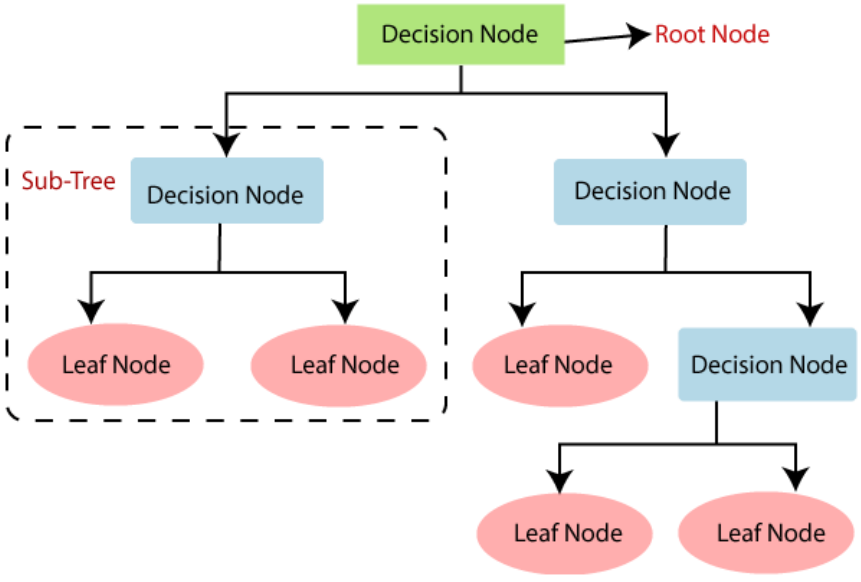

Figura 11 Esempio di Albero Decisionale [18]

La suddivisione dei campioni può avvenire sul valore della singola caratteristica o su un loro sottoinsieme. La scelta significativa per la costruzione di un albero decisionale è quella della tecnica da utilizzare per selezionare le caratteristiche da prendere in considerazione per suddividere i campioni di un nodo interno nei suoi nodi figli; per questo scopo, vengono utilizzate delle opportune misure valutative delle diverse caratteristiche. In problemi in cui la caratteristica da predire è una categoria, le misure valutative comunemente utilizzate per selezionare le caratteristiche che riescono a dividere i campioni di un nodo in base alla loro classe di appartenenza sono l'*Information Gain* e la *Gini Impurity*.

L'information Gain è utilizzata per decidere quali caratteristiche utilizzare per suddividere i campioni contenuti in un nodo. Per misurare l'*Information Gain* si utilizza il concetto di entropia, che è una misura quantificabile dell'impurità presente nella collezione di campioni presente in ogni nodo. Considerato un dataset con N classi, l'entropia può essere calcolata come:

$$E = -\sum_{i=1}^{N} p_i \log_2 p_i \quad ,$$

dove p_i è la probabilità di selezionare casualmente un campione di classe i. La bontà della suddivisione dei campioni in un nodo è calcolata attraverso l'*Information Gain* come

$$IG = E_{genitore} - E_{figlio}.$$

Viene scelta la caratteristica o le caratteristiche con *information gain* maggiore per il nodo radice dell'albero, costruiti i nodi figli per ogni valore delle caratteristiche precedentemente selezionate o in base ad una regola su queste caratteristiche e si ripete il processo per scegliere le caratteristiche necessarie alla divisione dei nodi figli ottenuti. Maggiore è l'entropia rimossa dal nodo genitore e maggiore sarà l'informazione ottenuta, maggiore è l'informazione ottenuta e migliore sarà la divisione.

Un'altra misura comunemente utilizzata per la selezione delle caratteristiche migliori con cui suddividere i campioni di un nodo è la *Gini Impurity* calcolata come

$$Gini\ Impurity = 1 - Gini\ Index,$$

dove l'indice di Gini di un insieme di campioni appartenenti a N classi è calcolato come:

$$Gini\ Index = \sum_{i=1}^{N} p_i^2.$$

Per selezionare la divisione migliore dei campioni presenti in un nodo interno si calcolano gli *Gini Index* dei nodi figli risultanti dalla divisione, successivamente si calcola la *Gini Impurity* dell'operazione di divisione utilizzando come *Gini Index* la somma degli indici dei nodi figli appena calcolati. Questa operazione viene ripetuta per ogni divisione, così da individuare la divisione migliore caratterizzata dal valore di *Gini Impurity* più basso.

Gli alberi decisionali possono creare degli alberi molto complessi in grado di specializzarsi sul riconoscimento dei campioni di addestramento ma non essere in grado di generalizzare per classificare correttamente campioni mai visti, questo fenomeno è chiamato *overfit*. Per evitare l'*overfit* del modello viene scelta una profondità massima in cui l'albero termina la continua divisione dei nodi.

L'algoritmo *Random Forest* è uno degli algoritmi più usati grazie alla sua accuratezza, semplicità e flessibilità. L'algoritmo è basato sulla costruzione di una foresta di alberi decisionali che cerca di aggirare i limiti dell'utilizzo del singolo albero costruendone una foresta ed utilizzando parametri casuali per la loro costruzione così da migliorarne la generalizzazione ed evitare l'*overfit*. A differenza degli alberi decisionali in cui si creano delle regole per decidere come dividere i campioni di un nodo, gli alberi della foresta casuale sono costruiti scegliendo ad ogni passo caratteristiche casuali per la divisione

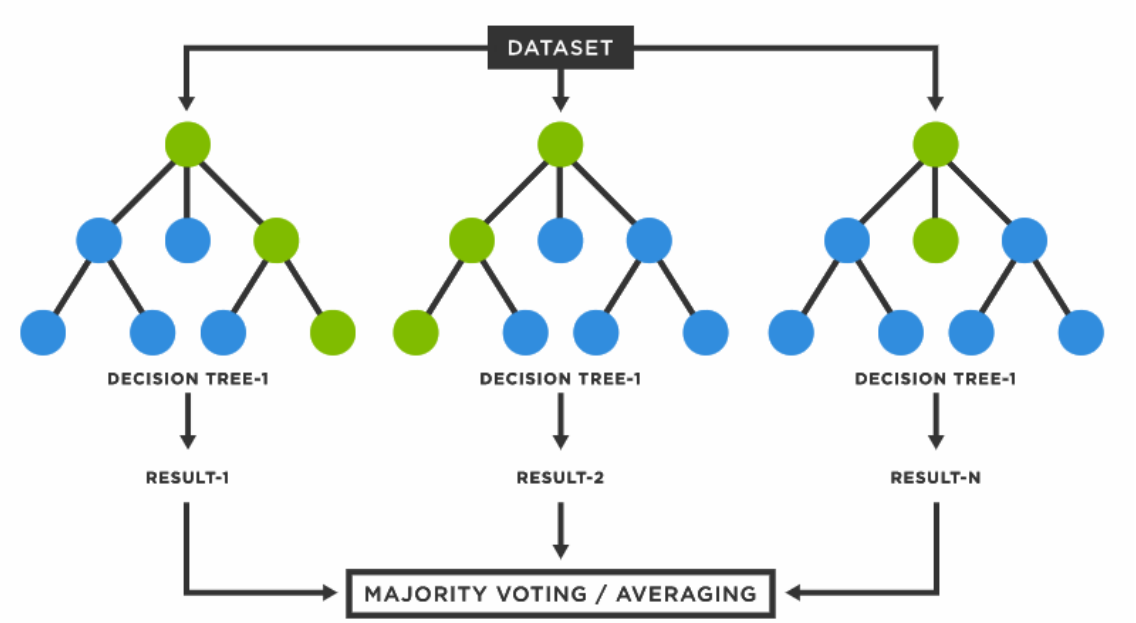

Figura 12 Struttura Random Forest [19]

dei campioni del nodo. Per costruire gli N alberi decisionali della foresta non viene utilizzato l'intero dataset di addestramento, ma una sua partizione. Per addestrare ogni singolo albero della foresta si utilizza un sottoinsieme casuale dell'insieme dei campioni di addestramento ottenuto tramite campionamento con sostituzione. Questo sottoinsieme di campioni viene generato attraverso un metodo casuale di selezione dei campioni di addestramento, in cui uno stesso campione può essere scelto più volte.
Addestrando ogni albero attraverso campioni selezionati casualmente si riduce il problema principale degli alberi decisionali che tendono all'*overfit* sui campioni di addestramento. I modelli che utilizzano insiemi di classificatori come *Random Forest,* sono più efficienti quando i singoli classificatori che li compongono non sono correlati tra loro, la correlazione degli alberi costruiti è già limitata dal diverso insieme di campioni su cui vengono addestrati i singoli alberi decisionali. Un altro metodo per rendere la foresta il più diversificata possibile sfrutta la selezione casuale delle caratteristiche per costruire gli alberi decisionali. Come descritto precedentemente un normale albero decisionale ad ogni nodo valuta una metrica per scegliere le caratteristiche che meglio dividono i campioni contenuti nel nodo in base alla loro classe di appartenenza. In una foresta casuale le caratteristiche utilizzate per la divisione dei nodi sono scelte da un sottoinsieme casuale di caratteristiche, questo rafforza la diversificazione degli alberi appartenenti alla foresta. Come rappresentato in Figura 12, la classificazione di un nuovo campione è data dalla sottomissione del campione ad ogni albero decisionale ed etichettandolo con la classe predetta dalla maggioranza degli alberi della foresta.

SUPPORT VECTOR MACHINE

Le Support Vector Machine o SVM sono modelli di classificazione che hanno come obiettivo quello di ricercare la retta di separazione delle classi che massimizza il margine tra di loro, margine inteso come la distanza minima della retta ai punti della classe. Le SVM sono un metodo di *machine learning* comunemente utilizzato per problemi di classificazione e regressione, ottenendo risultati competitivi su dati linearmente separabili. Le SVM sono in grado di lavorare anche su dati non linearmente separabili attraverso l'utilizzo di funzioni *kernel,* questa famiglia di funzioni sono usate per trasformare dati appartenenti ad uno spazio dimensionale in cui non sono linearmente separabili, in uno spazio a più alta dimensionalità in cui è possibile separare linearmente i dati. Il concetto base delle SVM è quello di separare le diverse classi del dataset fornito utilizzando degli iperpiani.

Se assumiamo che le classi dei dati a disposizione sono linearmente separabili, si ricerca l'iperpiano definito come: $w^T x + b = 0$, dove *w* rappresenta il vettore dei pesi, b rappresenta il *bias* e *x* rappresenta i campioni di addestramento.
L'iperpiano così definito divide lo spazio in due parti, una parte relativa alla classe positiva e una alla classe negativa. L'obiettivo della SVM è trovare i valori di *w* e *b* in modo da posizionare l'iperpiano il più lontano possibile dai campioni più vicini. Vengono costruiti due iperpiani chiamati *support vector* così definiti:

- $H_1 = w^T x_i + b = +1,\ con\ y_i = +1$
- $H_2 = w^T x_i + b = -1,\ \ con\ y_i = -1$

Quando si ha $H_1 \geq +1$ si individua il piano relativo alla classe positiva e con $H_2 \leq -1$ si individua il piano della classe negativa come rappresentato in Figura 13. Le due equazioni possono essere combinate come: $y_i(w^T x_i + b) - 1 \geq 0\ \ \forall i = 1,2, \dots, N$

Il margine è individuato come la somma delle distanze tra l'iperpiano e i *support vector.* Il margine è calcolato come: $m = d_1 + d_2 = \frac{2}{||w||}$, con d_1 e d_2 le distanze con i due *support vector*.

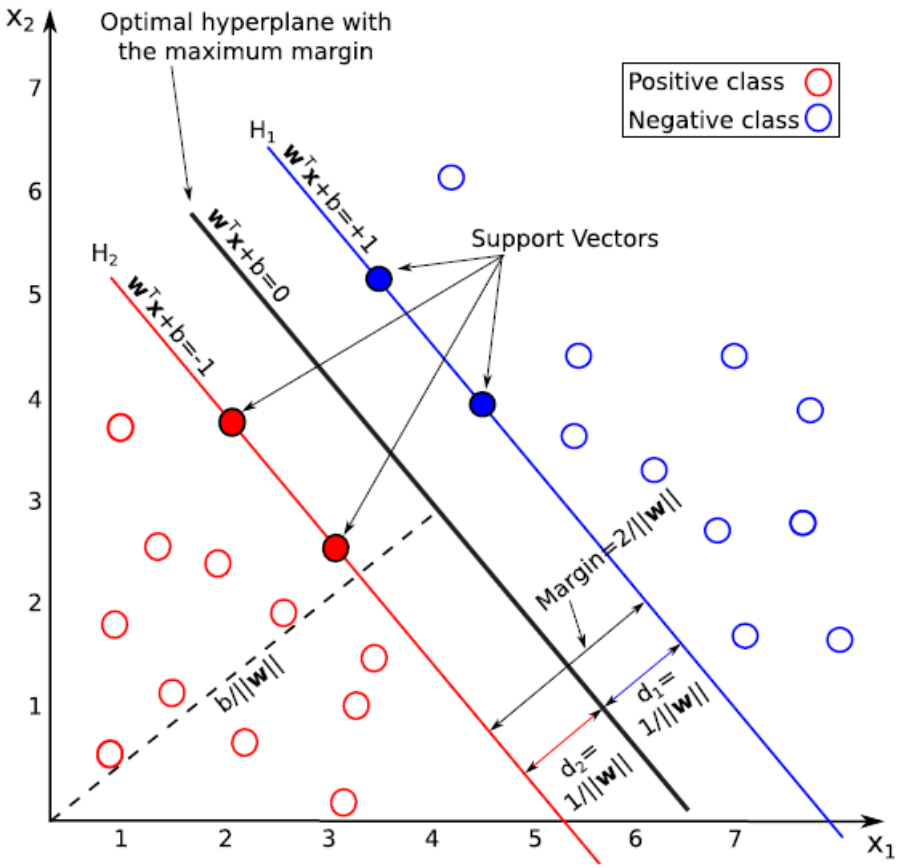

Figura 13 Esempio di Classificazione binaria con dati linearmente separabili [20]

L'obiettivo della SVM è quello di massimizzare il margine, questo può essere formalizzato attraverso la formula di Lagrange ottenuta combinando la funzione obiettivo e il suo vincolo.

$$L_P = \frac{||w||}{2} - \sum_i \alpha_i (y_i (w^T x_i + b) - 1) \, .$$

Il parametro α_i è il moltiplicatore di Lagrange per x_i. Il problema adesso è trovare i valori di *w, b* e α per cui si minimizza L_P, questo è chiamato problema primitivo e può essere rappresentato nella sua forma duale attraverso le derivate differenziali dei tre parametri. La forma duale del problema è la massimizzazione di:

$$max \; L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j \, , con \; \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i y_i = 0 \quad \forall i = 1,2, \dots, N.$$

I parametri α che massimizzano L_D vengono appresi attraverso il metodo di addestramento lungo il gradiente.

Il modello delle SVM può essere applicato anche in casi in cui i dati non sono linearmente separabili attraverso il cosiddetto *kernel trick*. In questi casi possono essere utilizzate le funzioni *kernel* che trasformano i dati in uno spazio a più alta dimensionalità dove è possibile separarli linearmente.

Le funzioni kernel sono definite come: $K(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j)$.

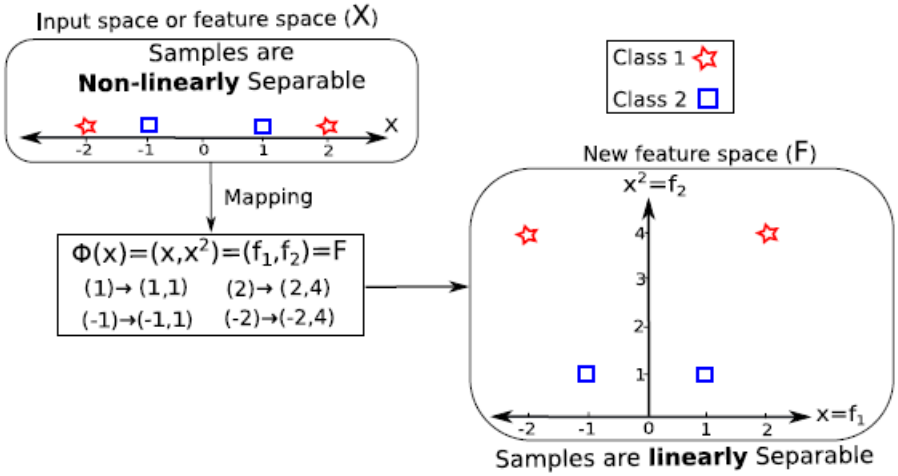

Figura 14 Esempio di come la funzione kernel viene utilizzata per trasformare i dati [20]

NAIVE BAYES

Il modello di classificazione *Naive Bayes* è un altro degli algoritmi di apprendimento supervisionato comunemente utilizzati per la rilevazione di malware come rappresentato in Figura 10. Un classificatore *Naive Bayes* è uno degli algoritmi di classificazione più semplice e per questo viene utilizzato per costruire dei modelli di machine learning ed ottenere delle predizioni nel più breve tempo possibile. *Naive Bayes* è una famiglia di algoritmi così chiamata poiché assume, in maniera semplificata, che ogni caratteristica sia indipendente dalle altre ed ogni caratteristica abbia lo stesso peso nella classificazione. Un classificatore *bayesiano* utilizza il teorema di Bayes per calcolare la probabilità condizionata che un campione appartenga ad una certa classe C osservati i valori delle sue caratteristiche. Preso un campione $X = \{x_1, x_2, \dots, x_n\}$ descritto dalle sue n caratteristiche, l'obiettivo del classificatore *Naive Bayes* è calcolare $P(C \mid X)$, cioè la probabilità a posteriori che il campione appartenga alla classe C condizionata dall'osservazione di X. La probabilità obiettivo può essere riscritta attraverso il teorema di Bayes come segue:

$$P(C \mid X) = \frac{P(X \mid C)\, P(C)}{P(X)}.$$

Il classificatore Naive Bayes è un metodo supervisionato, cioè basato sulla conoscenza di un insieme di campioni di addestramento, ognuno dei quali è descritto da n caratteristiche e dall'etichetta della classe di appartenenza. Dato un nuovo campione *X*, il classificatore vuole predire la sua classe di appartenenza calcolando le probabilità a posteriori per ogni classe esistente ed assegnando ad *X* la classe con la probabilità a posteriori maggiore, in modo che: $P(C_i \mid X) > P(C_j \mid X)$ con $1 \leq \mathrm{j} \leq \mathrm{n}$ $\mathrm{i} \neq \mathrm{j}$. L'obiettivo adesso è trovare la classe C_i che massimizzi la probabilità a posteriori, per il teorema di Bayes visto precedentemente: $P(C_i \mid X) = \frac{P(X \mid C_i)\, P(C_i)}{P(X)}$.

A questo punto, P(X) è costante per ogni classe esistente, $P(X \mid C_i)\, P(C_i)$ è l'unica parte della formula a variare e rappresenta il fattore da massimizzare. Se le probabilità a priori delle classi, $P(C_i)$, non sono note, si può assumere che queste siano equiprobabili e quindi che $P(C_1) = P(C_2) = \cdots = P(C_n)$. Le probabilità a priori delle classi si possono altrimenti calcolare come rapporto tra il numero dei campioni appartenenti ad una

specifica classe ed il numero totale dei campioni di addestramento. L'altro fattore rimasto da massimizzare è $P(\ X = \{x_1, x_2, \dots, x_n\}\ |C_i)$, che per campioni descritti da molte caratteristiche risulta molto oneroso da calcolare. La complessità computazionale di questo calcolo viene ridotta dall'assunzione *naive* che sta alla base dell'algoritmo, cioè che ogni caratteristica sia indipendente dalle altre ed ognuna di esse abbia lo stesso peso nella classificazione. Se le caratteristiche che descrivono un campione sono indipendenti tra loro allora l'ultimo fattore esaminato può essere riscritto come: $P(\ X = \{x_1, x_2, \dots, x_n\}\ |C_i)\ \approx\ \prod_{k=1}^{n} P(x_k|C_i)$.

Le probabilità a posteriori delle caratteristiche x data la classe C possono essere facilmente calcolate dall'insieme di addestramento. Se la caratteristica in esame è una caratteristica categorica, allora $P(x_k|C_i)$ sarà il rapporto tra il numero di campioni appartenenti alla classe C_i dell'insieme di addestramento con valore x_k su quella caratteristica ed il numero di campioni appartenenti alla classe C_i di tutto l'insieme di addestramento. Se la caratteristica in esame è una variabile continua, si assume che questa abbia una distribuzione Gaussiana con una media μ e una deviazione standard σ, ottenendo: $P(x_k|C_i) = \frac{1}{\sqrt{2\pi\sigma_i}}\exp - \frac{(x_k-\mu_i)^2}{2\sigma_i^2}$. I parametri μ_i e σ_i sono la media e la deviazione standard dei valori della caratteristica in esame su tutti i campioni appartenenti alla classe C_i.

Per classificare un nuovo campione *X*, viene calcolato il fattore $P(\ X\ |C_i)\ P(C_i)$ per ogni classe C. L'etichetta che il classificatore assegnerà a *X* sarà la classe C_i che massimizza il fattore $P(\ X\ |C_i)\ P(C_i)$.

3. APPROCCIO MULTI-CLASSIFICATORE

I sistemi intelligenti ibridi sono una famiglia di sistemi che utilizza metodi alternativi per la gestione di problemi reali sempre più complessi, che comportano ambiguità, incertezza ed alta dimensionalità dei dati. Gli approcci ibridi utilizzati hanno come obiettivo quello di sfruttare i punti di forza dei singoli metodi tradizionali e di superare le loro limitazioni attraverso la combinazione di più metodi [21]. I sistemi multi-classificatore (MCS) utilizzano delle combinazioni di classificatori con una base omogenea o eterogenea per prendere una decisione e proprio per questo sono una sottocategoria dei sistemi intelligenti ibridi.

L'apprendimento d'insieme non è un vero e proprio metodo di apprendimento, ma un metodo che combina modelli di apprendimento con specifiche aree di competenza, che nel caso in esame saranno definite da periodi di tempo, in un modello predittivo con l'obiettivo di migliorare la precisione della predizione del singolo modello.

In letteratura si possono trovare diversi vantaggi riconosciuti nell'utilizzo di sistemi multi-classificatore come i seguenti [21]:

- un MCS ha buone prestazioni quando la disponibilità dei dati è problematica, sia nel caso in cui vi è uno scarso numero di campioni per l'apprendimento e sia quando si ha un enorme disponibilità di campioni. In questo caso un MCS ci permette di addestrare i classificatori su porzioni del dataset ed unire le loro decisioni con un metodo opportuno;
- la combinazione di più classificatori può superare le prestazioni del miglior classificatore singolo sotto certe condizioni;
- ogni classificatore ha uno specifico dominio di competenza [22], nel quale supera tutti gli altri algoritmi che competono per lo stesso obiettivo, rendendo impossibile realizzare un singolo classificatore che abbia le prestazioni migliori in ogni problema di classificazione. MCS provano a selezionare il miglior classificatore dell'insieme adatto al problema in esame;
- i MCS possono essere facilmente implementabili su sistemi computazionali efficienti come le architetture multithreading. Un'altra area sono i sistemi

distribuiti, quando ad esempio un database è partizionato per ragioni di privacy, si possono elaborare soluzioni parziali la cui combinazione porta alla decisione finale di classificazione.

La struttura di un sistema multi-classificatore è definita da quattro principali scelte di progettazione: topologia del sistema, modalità di generazione dei classificatori, metodi di gestione dell'insieme di classificatori e tecniche di combinazione delle decisioni.

La topologia del sistema definisce in che modo i classificatori sono connessi, vi sono due topologie canoniche utilizzate per MCS. Nella maggioranza dei MCS, i classificatori sono connessi in parallelo, in cui ognuno di questi riceve gli stessi dati in input per poi combinare le singole decisioni. Questa topologia è adeguata su sistemi che ritengono importante il costo di utilizzo di un classificatore, eseguendo prima il classificatore meno oneroso ed in base alla valutazione della sua classificazione decidere se passare il campione al classificatore più oneroso ma specializzato in casi difficili. In casi speciali si utilizza una topologia sequenziale, principalmente utilizzata in problemi di data mining per migliorare la precisione di algoritmi di apprendimento su sottoinsiemi di dati sequenzialmente più problematici.

Per la generazione dell'insieme di classificatori, bisogna utilizzare basi di addestramento differenti, così da ottenere dei modelli specializzati in diverse aree d'interesse. Esistono diverse strategie per la generazione dell'insieme di classificatori [23], alcune di queste sono:

- *Inizializzazione differente*: quando il processo di addestramento è dipendente dall'inizializzazione, un'inizializzazione differente può portare a classificatori differenti. Ad esempio, nell'utilizzo di reti neurali, dove con la modifica della configurazione iniziale dei pesi si cambia il modello finale.
- *Parametri differenti:* i classificatori vengono generati con una configurazione degli iper-parametri differente che porterà a decisioni differenti.
- *Modelli di classificatori differenti*: questo metodo utilizza combinazioni di diversi modelli di classificazione addestrati sullo stesso spazio delle caratteristiche e sullo stesso insieme di campioni di addestramento. In questo caso la diversità è

intrinseca nelle proprietà di ogni modello, sistemi di questo tipo vengono chiamati modelli eterogenei.

- *Insiemi di addestramento differenti*: ogni classificatore è addestrato su una partizione dell'insieme di addestramento.
- *Insieme delle caratteristiche differente*: questa tecnica è utilizzata per applicazioni in cui i dati possono essere rappresentati in spazi delle caratteristiche distinti. Ad esempio, nel riconoscimento facciale, possono essere applicati diversi metodi di estrazione che producono insiemi distinti di caratteristiche. Ogni classificatore può essere addestrato su caratteristiche estratte con metodi diversi.

Per la gestione dell'insieme di classificatori bisogna definire il metodo con cui generare l'insieme, come opzionalmente aggiornarlo e come selezionare i classificatori migliori da passare al metodo che prenderà la decisione finale. L'insieme può essere generato addestrando i classificatori su partizioni differenti del dataset, su partizioni differenti di caratteristiche o prendendo in considerazione la specializzazione locale di ogni singolo classificatore. Per la selezione dei classificatori migliori vi sono due principali approcci:

- *Selezione statica*: lo spazio delle caratteristiche è partizionato a priori ed un classificatore è assegnato ad ogni partizione. La decisione riguardante un nuovo campione è presa dal classificatore della partizione a cui il campione appartiene. Uno degli algoritmi più noti che utilizza questo approccio è il *Classifier and Selection* [24].
- Selezione dinamica: un singolo o un insieme di classificatori è scelto per classificare ogni nuovo campione, basando la scelta sulle competenze locali di ogni classificatore a cui il nuovo campione appartiene. Solitamente l'area di competenza viene stimata attraverso la competenza dei singoli classificatori nel classificare i k-vicini di un dato campione, diversi metodi di selezione dinamica verranno discussi successivamente.

La decisione finale viene presa attraverso una funzione che combina gli output dei classificatori precedentemente selezionati secondo una regola. Le regole di combinazione non addestrabili sono dei metodi di aggregazione che richiedono alcune

assunzioni sui classificatori per ottenere buone prestazioni. Ad esempio, la votazione con maggioranza in cui si prende la decisione con il maggior numero di voti, risulta efficiente solo se i classificatori votanti risultano indipendenti. Le regole di combinazione addestrabili, non utilizzano delle regole fisse, come la votazione con maggioranza, ma la regola di combinazione è adattata allo specifico problema di classificazione. Le regole addestrabili affrontano il problema combinatorio come un problema di classificazione che presi gli output dei classificatori base li utilizza come caratteristiche per l'addestramento di un altro algoritmo che si occuperà di decidere il modo migliore di aggregare i risultati. Un'altra strategia di combinazione degli outputs è quella basata sull'utilizzo dei pesi, i pesi vengono scelti in base alla competenza locale di ogni classificatore a cui appartiene il campione in input. Le decisioni dei singoli classificatori vengono combinate in modo tale da dare più peso alle decisioni dei classificatori più competenti.

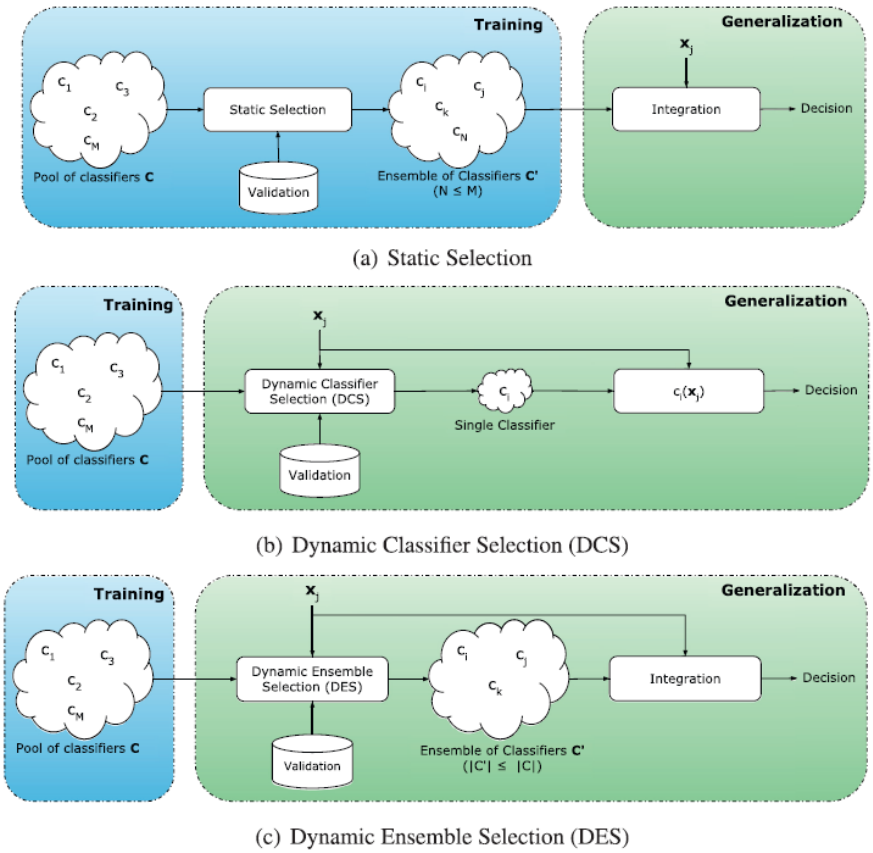

Figura 15 Differenze tra selezione statica e selezione dinamica di uno o più classificatori [25]

3.1 SELEZIONE DINAMICA

La selezione dei classificatori o del classificatore che si occuperà di assegnare la classe a nuovi campioni è parte cruciale del funzionamento di un sistema multi-classificatore. I metodi di selezione dinamica, che selezionano i migliori classificatori per ogni nuovo campione, sono caratterizzati da tre diversi fattori, per prima cosa è necessario definire la regione di competenza, regione che circonda il nuovo campione da classificare, così da stimare i livelli di competenza di ogni classificatore. Per stimare la competenza di ogni classificatore è necessario selezionare un criterio valutativo ed infine bisogna scegliere un metodo che selezioni un singolo classificatore o un insieme di classificatori basandosi sulla loro stima.

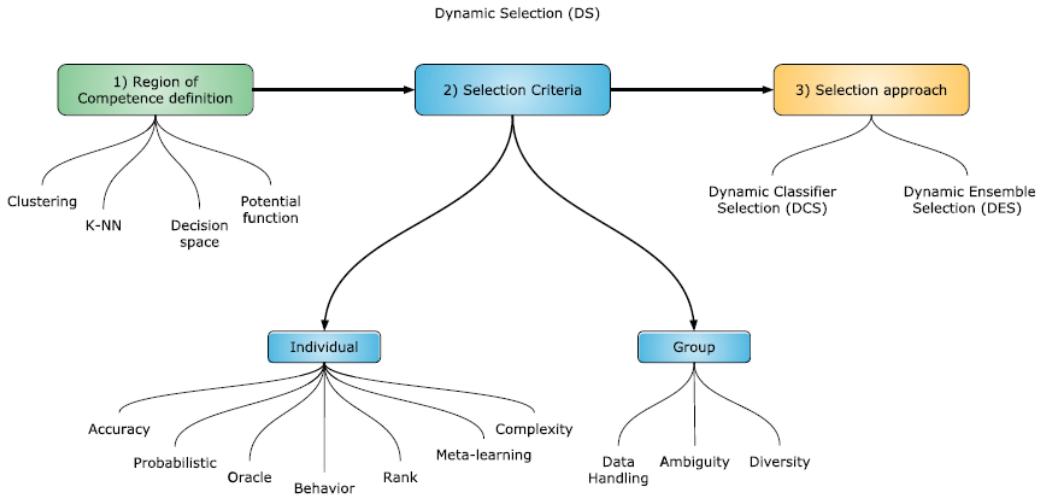

Figura 16 Tassonomia dei sistemi con selezione dinamica [25]

Durante la costruzione del modello finale sono stati valutati diversi metodi:

- *Overall local accuracy* (OLA) [26], metodo dinamico che seleziona un singolo classificatore per ogni nuovo campione, stima la precisione di ogni classificatore in una regione locale dello spazio delle caratteristiche che circonda il campione sconosciuto, scegliendo la decisione del classificatore più preciso in quella regione. La regione locale è definita come i *K-nearest neighbors* tra i campioni dell'insieme di addestramento. Per stimare la precisione locale vi sono due vie, si può prendere in considerazione la percentuale dei campioni nella regione correttamente classificati (OLA) oppure prendere la percentuale dei campioni

appartenenti ad una classe C dell'insieme di addestramento nella regione correttamente etichettati (Local Class Accuracy).

- *K-nearest-oracles* (KNORA) [27], metodo dinamico di selezione di un gruppo di classificatori, per ogni nuovo campione di test trova i *K-nearest neighbors* nell'insieme di valutazione e seleziona i classificatori che correttamente individuano i K vicini, questo metodo può essere applicato in vari modi, quelli presi in considerazione sono due:
    - *KNORA-Eliminate*, selezionati i K vicini di un campione test X e i K classificatori che individuano correttamente tutti i K vicini, Si utilizzano questi K classificatori per prendere una decisione sul test X attraverso un sistema di votazione. Se nessun classificatore riesce a classificare correttamente TUTTI e K i vicini di X, il valore K viene diminuito finché almeno un classificatore individua correttamente tutti i K vicini.
    - *KNORA-Union*, selezionati i K vicini di un campione test X e i K classificatori che individuano correttamente almeno uno dei K vicini, Si utilizzano questi K classificatori per prendere una decisione sul test X attraverso un sistema di votazione. Un classificatore può inviare più di un voto se classifica correttamente più di un vicino.
- *META-DES* [28], la selezione dinamica basata su un solo criterio per misurare la competenza di un classificatore non basta, META-DES prende in considerazione più criteri per avere una selezione migliore attraverso un framework di meta-apprendimento trattando la selezione dinamica come un problema di classificazione in cui i diversi criteri per misurare il livello di competenza di un classificatore fungono da meta-caratteristiche. Caratteristiche che date in input ad un meta-classificatore deciderà se il classificatore è competente o meno per la classificazione di un nuovo campione. META-DES si divide in tre fasi: generazione dell'insieme di classificatori attraverso il set di addestramento, estrazione delle meta-caratteristiche dall'insieme dei classificatori per addestrare il meta-classificatore che lavorerà come selettore ed infine una fase di generalizzazione dove le meta-caratteristiche del nuovo campione da classificare vengono passate al selettore che stimerà il classificatore più competente alla sua classificazione.

3.2 CONCEPT DRIFT

Il concept drift è un fenomeno per il quale le proprietà statistiche delle informazioni cambiano nel tempo in un modo non prevedibile. I metodi di machine learning applicati alla rilevazione di codice malevolo ed intrusioni, hanno la necessità di essere addestrati periodicamente su nuovi dati potenzialmente con proprietà statistiche differenti.

Il *concept drift* sintetizza sotto un unico termine uno dei più grandi problemi dei classificatori di malware, la presenza di dataset-shift [29]. Il *dataset shift* si presenta quando la distribuzione dei campioni di addestramento e di test è differente, viene diviso generalmente in tre diverse categorie:

- *Covariate shift*, fa riferimento al cambiamento della distribuzione degli input dell'insieme di addestramento e di test, si presenta solo in problemi in cui la classe è individuata dalle caratteristiche dei campioni, quando ad esempio il comportamento di un campione rappresentato dalle caratteristiche di X determina l'etichetta Y**.**
- *Prior probability shift,* fa riferimento al cambiamento nella distribuzione della classe obiettivo Y e si presenta solo in problemi in cui l'etichetta Y determina i valori delle caratteristiche di X, quando ad esempio il numero di una determinata classe aumenta o diminuisce drasticamente.
- *Concept shift,* fa rifermento al cambiamento nella relazione tra le variabili di input e la classe obiettivo e rappresenta la sfida più difficile rispetto alle precedenti.

Gli sviluppatori di malware cercano continuamente di evitare che i loro software vengano rilevati o classificati come malevoli dagli store di applicazioni, compagnie di antivirus o dall'utente stesso. Questo li spinge a trovare nuovi metodi per non essere rilevati, attraverso metodi che nascondono le caratteristiche dei loro malware, metodi che sfruttano nuove vulnerabilità e che riescano a celare gli intenti malevoli dell'applicazione. La tecnica più utilizzata per la gestione del concept drift sono gli *Streaming Ensemble Algorithm* (SEA) [30,31], dove i campioni in entrata vengono collezionati in gruppi di dati, che verranno utilizzati per addestrare nuovi modelli da aggiungere all'insieme dei classificatori che verranno valutati su nuovi campioni del

gruppo successivo. La valutazione o classificazione di nuovi campioni viene effettuata da un sottogruppo di classificatori selezionato dinamicamente con uno dei metodi descritti precedentemente.

Esistono diversi fattori che possono causare un *dataset shift*, le due cause principali sono gli ambienti non stazionari e la presenza di *bias* nella selezione dei campioni. La prima causa si riferisce ad un difetto sistematico nel processo di raccolta o etichettatura dei dati che fa sì che gli esempi di addestramento vengano selezionati in modo non uniforme tra la popolazione da modellare. Nella ricerca sulle scienze sociali, ad esempio, ci saranno sottoinsiemi della popolazione che sarà più facile da intervistare e caratterizzare di altri, questa facilità se sovra rappresentata tra i campioni di addestramento porta ad avere una rappresentazione minore dell'altra parte di popolazione portando a problemi di predizione in fase di test. Nelle applicazioni reali, come nei malware, solitamente i dati non sono stazionari nel tempo o nello spazio, creando così un cambiamento delle caratteristiche significative di una classe che renderanno problematiche le decisioni di modelli non aggiornati.

4. SISTEMA PROPOSTO

*** OMISSIS ***

4.1 ALGORITMO DI ADDESTRAMENTO

*** OMISSIS ***

RIFERIMENTI

[1] Rapporto Clusit – Edizione di ottobre 2022, https://clusit.it/pubblicazioni/

[2] Associazione GSM, https://www.gsma.com/

[3] Kaspersky, Minacce alla sicurezza dei dispositivi mobili Android, https://www.kaspersky.it/resource-center/threats/mobile

[4] You, Ilsun, and Kangbin Yim. "Malware obfuscation techniques: A brief survey." 2010 International conference on broadband, wireless computing, communication and applications. IEEE, 2010.

[5] Aslan, Ömer Aslan, and Refik Samet. "A comprehensive review on malware detection approaches." IEEE Access 8 (2020): 6249-6271.

[6] Borojerdi, Haniye Razeghi, and Mahdi Abadi. "MalHunter: Automatic generation of multiple behavioral signatures for polymorphic malware detection." ICCKE 2013. IEEE, 2013.

[7] Chandramohan, Mahinthan, et al. "A scalable approach for malware detection through bounded feature space behavior modeling." 2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2013.

[8] Elenkov, Nikolay. Android security internals: An in-depth guide to Android's security architecture. No Starch Press, 2014.

[9] Platform Architecture, https://developer.android.com/guide/platform

[10] Kambar, Mina Esmail Zadeh Nojoo, et al. "A survey on mobile malware detection methods using machine learning." 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). IEEE, 2022.

[11] Arora, Anshul, Sateesh K. Peddoju, and Mauro Conti. "Permpair: Android malware detection using permission pairs." IEEE Transactions on Information Forensics and Security 15 (2019): 1968-1982.

[12] Lin, Wei-Ting, and Jen-Yi Pan. "Mobile malware detection in sandbox with live event feeding and log pattern analysis." 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2016.

[13] Fu, Hao, et al. "Towards automatic detection of nonfunctional sensitive transmissions in mobile applications." IEEE Transactions on Mobile Computing 20.10 (2020): 3066-3080.

[14] Kandukuru, Satish, and R. M. Sharma. "Android malicious application detection using permission vector and network traffic analysis." 2017 2nd International Conference for Convergence in Technology (I2CT). IEEE, 2017.

[15] Wang, Shanshan, et al. "A mobile malware detection method using behavior features in network traffic." Journal of Network and Computer Applications 133 (2019): 15-25.

[16] Mahdavifar, Samaneh, Dima Alhadidi, and Ali Ghorbani. "Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder." Journal of Network and Systems Management 30.1 (2022): 1-34.

[17] Liu, Kaijun, et al. "A review of android malware detection approaches based on machine learning." IEEE Access 8 (2020): 124579-124607.

[18] Immagine Albero Decisionale, https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

[19] Immagine Random Forest, https://www.tibco.com/it/reference-center/what-is-a-random-forest

[20] Tharwat, Alaa. "Parameter investigation of support vector machine classifier with kernel functions." Knowledge and Information Systems 61 (2019): 1269-1302.

[21] Woźniak, Michał, Manuel Grana, and Emilio Corchado. "A survey of multiple classifier systems as hybrid systems." Information Fusion 16 (2014): 3-17.

[22] D. Wolpert, The supervised learning no-free-lunch theorems, in: Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications, 2001, pp. 25–42.

[23] Duin, Robert PW. "The combining classifier: to train or not to train?." Object recognition supported by user interaction for service robots. Vol. 2. IEEE, 2002.

[24] L.I. Kuncheva, Clustering-and-selection model for classifier combination, in: Fourth International Conference on Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies, KES 2000, Brighton, UK, 30 August - 1 September 2000, Proceedings, 2 Volumes, 2000, pp. 185–188.

[25] Cruz, Rafael MO, Robert Sabourin, and George DC Cavalcanti. "Dynamic classifier selection: Recent advances and perspectives." Information Fusion 41 (2018): 195-216.

[26] K. Woods, W.P. Kegelmeyer Jr., K. Bowyer, Combination of multiple classifiers using local accuracy estimates, IEEE Trans. Pattern Anal. Mach. Intell. 19 (4) (1997) 405–410.

[27] Ko, Albert HR, Robert Sabourin, and Alceu Souza Britto Jr. "From dynamic classifier selection to dynamic ensemble selection." Pattern recognition 41.5 (2008): 1718-1731.

[28] Cruz, Rafael MO, et al. "META-DES: A dynamic ensemble selection framework using meta-learning." Pattern recognition 48.5 (2015): 1925-1935.

[29] J. G. Moreno-Torres, T. Raeder, R. Ala´ız-Rodr´ıguez, N. V. Chawla, and F. Herrera. A unifying view on dataset shift in classification. Pattern Recognition, 2012

[30] W. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, ACM, New York, NY, USA, 2001, pp. 377–382.

[31] Zyblewski, P., Sabourin, R., & Woźniak, M. (2021). Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. Information Fusion, 66, 138-154.

[32] Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." 2008 eighth ieee international conference on data mining. IEEE, 2008.

[33] Guerra-Manzanares, Alejandro, Marcin Luckner, and Hayretdin Bahsi. "Android Malware Concept Drift using System Calls: Detection, Characterization and Challenges." Expert Systems with Applications (2022): 117200

[34] Guerra-Manzanares, Alejandro, Marcin Luckner, and Hayretdin Bahsi. "Concept Drift and Cross-Device Behavior: Challenges and Implications for Effective Android Malware Detection." Computers & Security (2022): 102757.

[35] Drebin dataset, https://www.sec.tu-bs.de/~danarp/drebin/

[36] Canadian Institute for Cybersecurity, https://www.unb.ca/cic/

[37] CIC datasets, https://www.unb.ca/cic/datasets/index.html

[38] AndroZoo, https://androzoo.uni.lu/

[39] Guerra-Manzanares, Alejandro, Hayretdin Bahsi, and Sven Nõmm. "KronoDroid: time-based hybrid-featured dataset for effective android malware detection and characterization." Computers & Security 110 (2021): 102399.

[41] De Paola A., Favaloro S., Gaglio S., Lo Re G., Morana M., Malware detection through low-level features and stacked denoising autoencoders, (2018) CEUR Workshop Proceedings, 2058

[42] De Paola A., Gaglio S., Lo Re G., Morana M. A hybrid system for malware detection on big data, (2018) INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, pp. 45 – 50, DOI:10.1109/INFCOMW.2018.8406963

[43] Agate V., Curaba M., Ferraro P., Lo Re G., Morana M., Secure e-voting in smart communities, (2020) CEUR Workshop Proceedings, 2597, pp. 1 – 11

[44] Concone F., De Paola A., Lo Re G., Morana M., Twitter analysis for real-Time malware discovery, (2017) 2017 AEIT International Annual Conference: Infrastructures for Energy and ICT: Opportunities for Fostering Innovation, AEIT 2017, 2017-January, pp. 1 – 6, DOI: 10.23919/AEIT.2017.824055

[45] F. Concone, G. Lo Re, M. Morana, S. K. Das. SpADe: Multi-Stage Spam Account Detection for Online Social Networks. In IEEE Transactions on Dependable and Secure Computing (TDSC), 2022, doi: 10.1109/TDSC.2022.3198830