

UNIVERSITÀ POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA



*Corso di Laurea Magistrale in
Ingegneria Informatica e dell'Automazione*

*Analisi dataset CSE-CIC-IDS 2018 mediante tecniche di
unsupervised learning*

Professore:
LUCA SPALAZZI

Sviluppatori:
CAPORUSSO CHIARA AMALIA
CEKA DAVID
PROIETTI MELISSA
SCALELLA SIMONE
SCHIAVONI RICCARDO

ANNO ACCADEMICO 2022-2023

Il codice è disponibile al seguente indirizzo: <https://github.com/Simone-Scalella/AdvancedCyberSecurity>

Nella cartella **image_metriche_topfeature.zip** sono presenti le metriche, le immagini e le feature individuate nel lavoro descritto in questa relazione.

Nella cartella **image** sono presenti tutte le immagini relative al lavoro di clustering, organizzate per file analizzato. Sempre nella medesima cartella, sono presenti ulteriori due cartelle contenenti gli histogrammi riportanti i valori di ciascuna feature analizzata e gli scatterplot ottenuti senza riduzione e bilanciamento del dataset su feature significative. È inoltre presente un'immagine riportante lo scatterplot tra le due feature significative utilizzate per il clustering, su dataset bilanciato. Nella cartella **clustering totale** sono presenti le immagini relative al clustering effettuato sul file totale.

Nella cartella **metriche** sono presenti le metriche ottenute in fase di clustering, organizzate per file analizzato. Le metriche relative al clustering sull'intero dataset sono presenti dentro la medesima cartella.

Nella cartella **top_feature** sono presenti le feature più significative ottenute per ciascun file, organizzate per il tipo di algoritmo di feature selection utilizzato. Sono inoltre presenti le combinazioni ottenute per ciascuna coppia di feature tra quelle individuate come più importanti per tali algoritmi. Nel file **top5.csv** sono invece presenti le 5 feature più importanti.

Indice

Indice	ii
1 Introduzione	1
1.1 Obiettivi del progetto	1
1.2 Dataset	3
2 Strumenti e Metodi	11
2.1 Linguaggi e librerie	11
2.2 Strumenti	11
2.2.1 Configurazione macchina remota	11
3 Sviluppo del progetto	13
3.1 Preprocessing dei dati e selezione delle feature	13
3.1.1 Operazioni di preprocessing	13
3.1.2 Operazioni di feature selection	14
3.2 Clustering	17
3.2.1 Scelta degli Algoritmi di Clustering	18
3.2.2 Applicazione degli Algoritmi di Clustering	22
3.3 Principal Component Analysis (PCA)	28
4 Risultati	29
4.1 Friday-02-03-2018_TrafficForML_CICFlowMeter	30
4.2 Friday-16-02-2018_TrafficForML_CICFlowMeter	35
4.3 Friday-23-02-2018_TrafficForML_CICFlowMeter	40
4.4 Thursday-20-02-2018_TrafficForML_CICFlowMeter	45
4.5 Thursday-01-03-2018_TrafficForML_CICFlowMeter	50
4.6 Thursday-15-02-2018_TrafficForML_CICFlowMeter	55
4.7 Thursday-22-02-2018_TrafficForML_CICFlowMeter	60
4.8 Wednesday-14-02-2018_TrafficForML_CICFlowMeter	65
4.9 Wednesday-21-02-2018_TrafficForML_CICFlowMeter	70
4.10 Wednesday-28-02-2018_TrafficForML_CICFlowMeter	75
4.11 Clustering sull'intero dataset	80
5 Conclusioni e Sviluppi futuri	83

5.1 Sviluppi futuri	83
5.2 Conclusioni	84
Bibliografia	86
Elenco delle figure	87

Capitolo 1

Introduzione

1.1 Obiettivi del progetto

Il seguente progetto ha come scopo l'applicazione di algoritmi di **unsupervised learning** su un dataset contenente traffico dati, sia benigno che maligno, al fine di suddividerli in maniera efficiente. Si osserverà e si valuterà il comportamento di questi algoritmi al fine di stabilire quali tra questi ci restituisce il risultato migliore.

Il CSE-CIC-IDS2018 è un progetto di collaborazione tra il Communications Security Establishment (**CSE**) e il Canadian Institute for Cybersecurity (**CIC**).

Il rilevamento delle anomalie è stato al centro dell'attenzione di molti ricercatori per il suo potenziale nel rilevare nuove tipologie di attacchi informatici.

Tuttavia, la sua adozione in applicazioni reali è stata ostacolata dalla complessità del sistema, il quale richiede una quantità sostanziale di test, valutazioni e messe a punto prima dell'implementazione.

Per il test e la valutazione la metodologia più idealistica risulta essere l'esecuzione di questi sistemi su dataset già etichettati e con una serie completa ed estesa di intrusioni e comportamenti anomali.

Tuttavia, la disponibilità di tali dataset è estremamente rara, perché da un lato molti di questi sono interni e non possono essere condivisi per questioni di privacy mentre dall'altro sono pesantemente anonimizzati e non riflettono le tendenze attuali oppure mancano di alcune caratteristiche statistiche. Per questi motivi non esiste ancora un dataset perfetto ed i ricercatori devono ricorrere a set di dati spesso non ottimali.

Poiché i comportamenti e i modelli di rete cambiano e le intrusioni si evolvono, è diventato necessario abbandonare i dataset statici e unici per passare a dataset generati in modo dinamico, che non solo riflettano la composizione del traffico e le intrusioni del momento, ma siano anche modificabili, estensibili e riproducibili.

Per superare queste carenze è stato ideato un approccio sistematico per generare set di dati al fine di analizzare, testare e valutare i sistemi di rilevamento delle intrusioni,

con particolare attenzione ai rilevatori di anomalie network-based.

Con questo progetto è stato, perciò, sviluppato un approccio sistematico per generare set di dati di riferimento, diversificati e completi, per il rilevamento delle intrusioni e basati sulla creazione di profili utente che contengono rappresentazioni astratte di eventi e comportamenti osservati sulla rete.

I profili sono, successivamente, stati combinati per generare una serie di set di dati diversi, ciascuno con un insieme unico di caratteristiche, che coprono una parte del dominio di valutazione.

Al termine di tale processo è stato ottenuto il **CSE-CIC-IDS2018 dataset**, il quale verrà approfondito nel capitolo successivo.

1.2 Dataset

Il dataset è disponibile al seguente link: <https://www.unb.ca/cic/datasets/ids-2018.html>.

Esso include sette diversi scenari di attacco:

- Brute Force: può essere sfruttato per recuperare qualunque tipo di password. Questo attacco viene eseguito utilizzando software o script automatizzati che tentano ripetutamente di indovinare le credenziali di accesso, provando diverse combinazioni di password fino a trovare quella corretta. Il tempo di riuscita dell'impresa dipende dalla velocità di calcolo del computer, nonché dalla complessità e la lunghezza della password (quanti e quali caratteri sono stati utilizzati),
- Heartbleed: è un'importante vulnerabilità di sicurezza che è stata scoperta nel protocollo di sicurezza **OpenSSL** nel 2014. OpenSSL è una libreria crittografica open source ampiamente utilizzata per implementare il protocollo SSL/TLS che fornisce la crittografia dei dati durante la comunicazione su Internet. Consente ad un attaccante di accedere alla memoria di un server protetto da OpenSSL e di estrarre informazioni sensibili come chiavi private, nomi utente, password e altri dati critici. L'attacco sfrutta un difetto nella gestione dell'estensione Heartbeat di OpenSSL, che consente agli utenti di inviare un pacchetto di dati "heartbeat" per verificare la connessione. Tuttavia, a causa di un errore di programmazione, il server OpenSSL non valida correttamente la dimensione dei dati inviati e restituisce in modo errato un segmento di memoria contenente dati sensibili,
- Botnet: è un tipo di attacco informatico in cui un gruppo di dispositivi infettati, noti come **bot**, viene controllato da un attaccante remoto per eseguire operazioni dannose. Questi bot possono essere computer, server, dispositivi IoT o qualsiasi altra macchina connessa a Internet. Un botnet viene solitamente creato sfruttando la vulnerabilità di dispositivi non protetti o compromettendo i sistemi tramite malware come trojan, worm o virus. Una volta infettato, ogni dispositivo diventa un "bot" sotto il controllo dell'attaccante. L'attaccante utilizza poi i bot per eseguire attività malevole come attacchi DDoS (Distributed Denial of Service), invio di spam, furto di informazioni personali, diffusione di malware o altre azioni dannose,
- DoS: è un tipo di attacco informatico mirato a rendere un servizio o una risorsa di rete inaccessibili agli utenti legittimi, sovraccaricandoli o disabilitandoli temporaneamente. L'obiettivo principale di un attacco DoS è quello di interrompere il normale funzionamento di un sistema, rendendolo indisponibile per gli utenti,
- DDoS: è un tipo di attacco informatico in cui un'ampia rete di computer compromessi, noti come **botnet**, viene utilizzata per sovraccaricare un sistema o una risorsa di rete, rendendoli inaccessibili agli utenti legittimi. A differenza di un attacco DoS tradizionale, in cui l'attacco proviene da un singolo punto, un attacco DDoS coinvolge molteplici punti di attacco distribuiti in diverse posizioni geografiche. Gli attaccanti di un attacco DDoS sfruttano la potenza di calcolo

combinata di una rete di computer compromessi per inviare un'enorme quantità di traffico o richieste al bersaglio contemporaneamente, saturandone le risorse e rendendolo inutilizzabile. Questi computer compromessi, noti come bot, sono spesso infettati da malware o da altri tipi di software dannosi.

- Web Attack: sono diverse tecniche utilizzate dagli hacker per sfruttare le vulnerabilità presenti nelle applicazioni Web e nei siti Web al fine di ottenere l'accesso non autorizzato, compromettere i dati o causare danni al sistema. Alcuni esempi comuni di attacchi Web sono le iniezioni di codice (gli attaccanti inseriscono deliberatamente codice dannoso, come **SQL injection** o **JavaScript injection**, all'interno di campi di input o parametri delle pagine Web), Cross-Site Scripting (**XSS**: gli attaccanti inseriscono script dannosi all'interno delle pagine Web visualizzate dagli utenti), Cross-Site Request Forgery (**CSRF**: gli attaccanti sfruttano la fiducia tra un sito Web e un utente legittimo ed attraverso un link o un'immagine ingannevole, quest'ultimo viene indotto a compiere azioni non intenzionali su un sito Web senza la propria consapevolezza o autorizzazione), attacco di forzatura delle password (gli attaccanti utilizzano programmi automatizzati per tentare di indovinare o forzare le password degli utenti), attacco di defacement (si verificano quando gli hacker modellano o alterano il contenuto di un sito Web per scopi malevoli o per mostrare il loro messaggio) e attacco di enumerazione (si concentrano sulla scoperta di informazioni sensibili sugli utenti o sulle risorse del sistema, ad esempio tentando di indovinare nomi utente o elencando file o directory protetti),
- infiltrazione della rete dall'interno: nota anche come **attacco insider**, si verifica quando una persona interna all'organizzazione, come un dipendente o un amministratore di sistema, utilizza il proprio accesso privilegiato per commettere azioni dannose o compromettere la sicurezza della rete. Alcuni esempi di possibili attività di infiltrazione della rete dall'interno sono il furto di dati sensibili (un insider può rubare informazioni sensibili per scopi personali o per venderle a terzi), accesso non autorizzato (un insider può utilizzare le proprie credenziali privilegiate per accedere a risorse o aree della rete a cui non ha diritto di accesso), introduzione di malware (un insider può introdurre volontariamente malware o virus all'interno della rete aziendale, compromettendo i sistemi e facilitando l'accesso non autorizzato a terzi) e modifica o distruzione dei dati (un insider può alterare o eliminare dati critici, causando gravi danni all'azienda o ai suoi clienti).

L'infrastruttura di attacco comprende 50 macchine e l'organizzazione vittima ha 5 dipartimenti, comprendente 420 macchine e 30 server.

Il dataset è stato organizzato per giorno: per ogni giorno (10 totali), sono registrati i dati grezzi, incluso il traffico di rete (Pcaps) e i log degli eventi (Windows e Ubuntu Event Logs) per macchina.

Nel processo di estrazione delle caratteristiche dai dati grezzi, è stato usato **CICFlowMeter-V3**.

CICFlowMeter è un generatore di flussi di traffico di rete, scritto in Java, che gene-

ra flussi bidirezionali (Biflow), in cui il primo pacchetto determina le direzioni avanti (dalla sorgente alla destinazione) e all'indietro (dalla destinazione alla sorgente).

Da qui vengono estratte le 83 caratteristiche statistiche come durata, numero di pacchetti, numero di byte, lunghezza dei pacchetti, ecc, le quali vengono anche calcolate separatamente in direzione avanti e indietro.

L'output dell'applicazione è in formato file CSV con sei colonne, etichettate per ciascun flusso, e con oltre 80 funzioni di traffico di rete.

Le funzioni estratte sono elencate nella seguente tabella:

FUNZIONE	SIGNIFICATO
fl_dur (Flow Duration)	Durata del flusso
tot_fw_pk (Tot Fwd Pkts)	Numero totale pacchetti in direzione forward
tot_bw_pk (Tot Bwd Pkts)	Numero totale pacchetti in direzione backward
tot_l_fw_pkt (TotLen Fwd Pkts)	Dimensione totale del pacchetto in direzione forward
tot_l_bw_pkt (TotLen Bwd Pkts)	Dimensione totale del pacchetto in direzione backward
fw_pkt_l_max (Fwd Pkt Len Max)	Dimensione massima del pacchetto in direzione forward
fw_pkt_l_min (Fwd Pkt Len Min)	Dimensione minima del pacchetto in direzione forward
fw_pkt_l_avg (Fwd Pkt Len Mean)	Dimensione media del pacchetto in direzione forward
fw_pkt_l_std (Fwd Pkt Len Std)	Deviazione standard della dimensione del pacchetto in direzione forward
Bw_pkt_l_max (Bwd Pkt Len Max)	Dimensione massima del pacchetto in direzione backward
Bw_pkt_l_min (Bwd Pkt Len Min)	Dimensione minima del pacchetto in direzione backward

FUNZIONE	SIGNIFICATO
Bw_pkt_l_avg (Bwd Pkt Len Mean)	Dimensione media del pacchetto in direzione backward
Bw_pkt_l_std (Bwd Pkt Len Std)	Deviazione standard della dimensione del pacchetto in direzione backward
fl_byt_s (Flow Byts/s)	Numero di byte trasferiti per secondo (flow byte rate)
fl_pkt_s (Flow Pkts/s)	Numero di pacchetti trasferiti per secondo (flow packet rate)
fl_iat_avg (Flow IAT Mean)	Tempo medio tra due flussi
fl_iat_std (Flow IAT Std)	Deviazione standard di tempo tra due flussi
fl_iat_max (Flow IAT Max)	Tempo massimo tra due flussi
fl_iat_min (Flow IAT Min)	Tempo minimo tra due flussi
fw_iat_tot (Fwd IAT Tot)	Tempo totale tra due pacchetti inviati in direzione forward
fw_iat_avg (Fwd IAT Mean)	Tempo medio tra due pacchetti inviati in direzione forward
fw_iat_std (Fwd IAT Std)	Deviazione standard di tempo tra due pacchetti inviati in direzione forward
fw_iat_max (Fwd IAT Max)	Tempo massimo tra due pacchetti inviati in direzione forward
fw_iat_min (Fwd IAT Min)	Tempo minimo tra due pacchetti inviati in direzione forward
bw_iat_tot (Bwd IAT Tot)	Tempo totale tra due pacchetti inviati in direzione backward
bw_iat_avg (Bwd IAT Mean)	Tempo medio tra due pacchetti inviati in direzione backward
bw_iat_std (Bwd IAT Std)	Deviazione standard di tempo tra due pacchetti inviati in direzione backward
bw_iat_max (Bwd IAT Max)	Tempo massimo tra due pacchetti inviati in direzione backward
bw_iat_min (Bwd IAT Min)	Tempo minimo tra due pacchetti inviati in direzione backward

FUNZIONE	SIGNIFICATO
fw_psh_flag (Fwd PSH Flags)	Numero di volte che PSH (push) flag è stato cambiato nei pacchetti che viaggiano in direzione forward
bw_psh_flag (Bwd PSH Flags)	Numero di volte che PSH (push) flag è stato cambiato nei pacchetti che viaggiano in direzione backward
fw_urg_flag (Fwd URG Flags)	Numero di volte che URG (urgent) flag è stato cambiato nei pacchetti che viaggiano in direzione forward
bw_urg_flag (Bwd URG Flags)	Numero di volte che URG (urgent) flag è stato cambiato nei pacchetti che viaggiano in direzione backward
fw_hdr_len (Fwd Header Len)	Byte totali utilizzati per l'intestazione in direzione forward
bw_hdr_len (Bwd Header Len)	Byte totali utilizzati per l'intestazione in direzione backward
fw_pkt_s (Fwd Pkts/s)	Numero di pacchetti forward per secondo
bw_pkt_s (Bwd Pkts/s)	Numero di pacchetti backward per secondo
pkt_len_min (Pkt Len Min)	Lunghezza minima di un flusso
pkt_len_max (Pkt Len Max)	Lunghezza massima di un flusso
pkt_len_avg (Pkt Len Mean)	Lunghezza media di un flusso
pkt_len_std (Pkt Len Std)	Deviazione standard della lunghezza di un flusso
pkt_len_va (Pkt Len Var)	Tempo di arrivo minimo tra un pacchetto e l'altro [Minimum inter-arrival time of packet (il tempo di inter-arrivo è il tempo che intercorre tra un arrivo nel sistema e il successivo)]

FUNZIONE	SIGNIFICATO
fin_cnt (FIN Flag Cnt)	Numero di pacchetti con flag = FIN [FIN (finished flag) viene utilizzato per indicare l'ultimo pacchetto inviato dal mittente, indicante che non ci sono più dati dopo di quello inviati dal mittente]
syn_cnt (SYN Flag Cnt)	Numero di pacchetti con flag = SYN [SYN (synchronization flag) viene utilizzato per stabilire un handshake a tre vie tra due host. Solo il primo pacchetto sia del mittente che del destinatario dovrebbe avere questo flag impostato]
rst_cnt (RST Flag Cnt)	Numero di pacchetti con flag = RST [RST (reset flag) viene inviato dal destinatario al mittente quando un pacchetto viene inviato ad un particolare host che non lo aspettava]
pst_cnt (PSH Flag Cnt)	Numero di pacchetti con flag = PUSH [PUSH indica al destinatario di elaborare questi pacchetti man mano che vengono ricevuti invece di memorizzarli nel buffer]
ack_cnt (ACK Flag Cnt)	Numero di pacchetti con flag = ACK [ACK (acknowledgment flag) viene utilizzata per confermare la corretta ricezione di un pacchetto]
urg_cnt (URG Flag Cnt)	Numero di pacchetti con flag = URG [URG (urgent flag) viene utilizzato per notificare il destinatario di processare i pacchetti urgenti prima degli altri pacchetti]
cwe_cnt (CWE Flag Count)	Numero di pacchetti con flag = CWE [CWE (Congestion Window Reduced flag) viene utilizzato dal mittente per indicare che ha ricevuto un pacchetto con il flag ECE impostato]

FUNZIONE	SIGNIFICATO
ece_cnt (ECE Flag Cnt)	Numero di pacchetti con flag = ECE [ECE viene utilizzato per indicare se il peer TCP è compatibile con ECN; consente la notifica end-to-end della congestione della rete senza eliminare i pacchetti]
down_up_ratio (Down/Up Ratio)	Rapporto tra download e upload
pkt_size_avg (Pkt Size Avg)	Dimensione media del pacchetto
fw_seg_avg (Fwd Seg Size Avg)	Dimensione media osservata in direzione forward
bw_seg_avg (Bwd Seg Size Avg)	Dimensione media osservata in direzione backward
fw_byt_blk_avg (Fwd Byts/b Avg)	Numero medio del tasso di massa di bytes (bytes bulk rate) in direzione forward
fw_blk_rate_avg (Fwd Blk Rate Avg)	Numero medio del tasso di massa in direzione forward
bw_byt_blk_avg (Bwd Byts/b Avg)	Numero medio del tasso di massa di bytes (bytes bulk rate) in direzione backward
bw_pkt_blk_avg (Bwd Pkts/b Avg)	Numero medio del tasso di massa di pacchetti (packets bulk rate) in direzione backward
bw_blk_rate_avg (Bwd Blk Rate Avg)	Numero medio del tasso di massa in direzione backward
subfl_fw_pk (Subflow Fwd Pkts)	Numero medio di pacchetti in un sottoflusso in direzione forward
subfl_fw_byt (Subflow Fwd Byts)	Numero medio di bytes in un sottoflusso in direzione forward
subfl_bw_pk (Subflow Bwd Pkts)	Numero medio di pacchetti in un sottoflusso in direzione backward
subfl_bw_byt (Subflow Bwd Byts)	Numero medio di bytes in un sottoflusso in direzione backward
fw_win_byt (Init Fwd Win Byts)	Numero di bytes inviati nella finestra iniziale in direzione forward

FUNZIONE	SIGNIFICATO
bw_win_byt (Init Bwd Win Byts)	Numero di bytes inviati nella finestra iniziale in direzione backward
fw_act_pkt (Fwd Act Data Pkts)	Numero di pacchetti con almeno 1 byte di payload di dati TCP in direzione forward
fw_seg_min (Fwd Seg Size Min)	Dimesione minima di segmento osservato in direzione forward
atv_avg (Active Mean)	Tempo medio di attività di un flusso prima di diventare inattivo
atv_std (Active Std)	Deviazione standard del tempo di attività di un flusso prima di diventare inattivo
atv_max (Active Max)	Tempo massimo di attività di un flusso prima di diventare inattivo
atv_min (Active Min)	Tempo minimo di attività di un flusso prima di diventare inattivo
idl_avg (Idle Mean)	Tempo medio di inattività di un flusso prima di diventare attivo
idl_std (Idle Std)	Deviazione standard del tempo di inattività di un flusso prima di diventare attivo
idl_max (Idle Max)	Tempo massimo di inattività di un flusso prima di diventare attivo
idl_min (Idle Min)	Tempo minimo di inattività di un flusso prima di diventare attivo
Label	indica se il flusso è benigno o se è presente un qualsiasi tipo di attacco

Capitolo 2

Strumenti e Metodi

2.1 Linguaggi e librerie

Tutto il codice presente nel progetto è stato scritto in linguaggio Python[1], sfruttando svariate librerie legate al mondo della rappresentazione dei dati, del calcolo scientifico e della clusterizzazione. Tra le più importanti possiamo individuare:

- **Pandas:** strumento per la manipolazione e l'analisi dei dati veloce e semplice da utilizzare.
- **Scikit-Learn:** libreria open source di apprendimento automatico che contiene algoritmi di classificazione, regressione, clustering (raggruppamento), macchine a vettori di supporto, regressione logistica, classificatore bayesiano, k-mean e DBSCAN, ed è progettato per operare con le librerie *NumPy* e *SciPy*.

2.2 Strumenti

Per lo sviluppo di questo progetto, è necessario installare i seguenti tool:

- versione Python 3.11, utilizzata per lo sviluppo del progetto.
- librerie Python utilizzate: Pandas, Scikit-Learn.
- Jupyter Notebook[2], per l'elaborazione interattiva in tutti i linguaggi di programmazione, utilizzato per l'elaborazione di parti di codice al fine di analizzarne i risultati intermedi.

2.2.1 Configurazione macchina remota

Data la grande quantità di dati a disposizione su cui sono stati applicati algoritmi di apprendimento non supervisionato è risultato necessario l'utilizzo di una macchina su server remoto, con buone capacità prestazionali e che ci permetesse di eseguire, in tempi ragionevoli, gli algoritmi implementati.

```
MemoryError: Unable to allocate 77.8 GiB for an array with shape (10445255380,) and data type float64
```

Figura 2.1: MemoryError su macchina con 16 Gb di RAM

Per stabilire il collegamento con tale macchina è risultato necessario l'utilizzo del software **PuTTY**, ovvero un client SSH Telnet e rlogin combinato con un emulatore di terminale per la gestione in remoto di sistemi informatici.

Mediante PuTTY è stata stabilita una connessione remota, attraverso la creazione di un tunnel SSH. Stabilita la connessione, per interfacciarsi con la macchina remota, è necessario installare il software RealVNC.

Capitolo 3

Sviluppo del progetto

Lo sviluppo del progetto è stato suddiviso in quattro fasi principali:

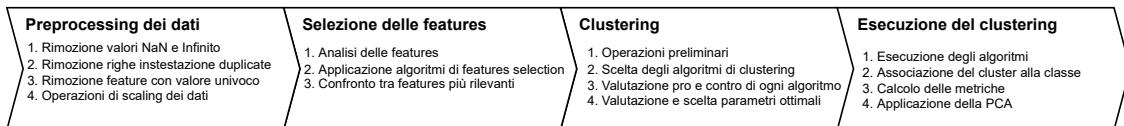


Figura 3.1: Flowchart

Di seguito approfondiamo la descrizione delle singole fasi.

3.1 Preprocessing dei dati e selezione delle feature

3.1.1 Operazioni di preprocessing

Per poter applicare i vari algoritmi di clustering è stato necessario pulire il dataset di partenza.

Dopo un'attenta esplorazione iniziale del dataset è stato osservato che questo contiene al suo interno valori **NaN** e **infinito**, oltre a campi che contengono i nomi delle colonne. Per questo motivo, la prima operazione effettuata è stata la loro eliminazione.

Inoltre, visto l'elevato numero di feature, basandoci sulla loro descrizione, si è proceduto all'eliminazione di quelle che assumono valori univoci (ad esempio indirizzi IP del mittente e del destinatario) e quelle che non hanno alcun contenuto informativo d'interesse al fine di discriminare traffico benigno e maligno (come ad esempio il *timestamp*).

Infine, è stato osservato che le feature presenti nel dataset assumono valori su scale completamente diverse.

Questa condizione genera anomalie e risultati sbagliati nel momento in cui si applicano

gli algoritmi di clustering e feature selection. Per questo motivo, è stata eseguita un'operazione di normalizzazione per modificare i valori delle feature, in modo tale che assumano un valore compreso tra zero e uno.

La formula applicata è la seguente:

$$\frac{X - \min(x)}{\max(x) - \min(x)}$$

3.1.2 Operazioni di feature selection

Terminata l'operazione di pulizia del dataset è stato necessario ridurre il numero di feature.

Il processo di feature selection è iniziato con un'analisi dei valori delle feature. Al fine di ottenere una panoramica completa, abbiamo deciso di stampare un istogramma per ciascuna feature, in modo da visualizzare il conteggio dei diversi valori che la feature assume.

Le feature che mantengono costantemente lo stesso valore, sia per il traffico benigno che per quello maligno, non risultano interessanti in quanto mancano di potere discriminante.

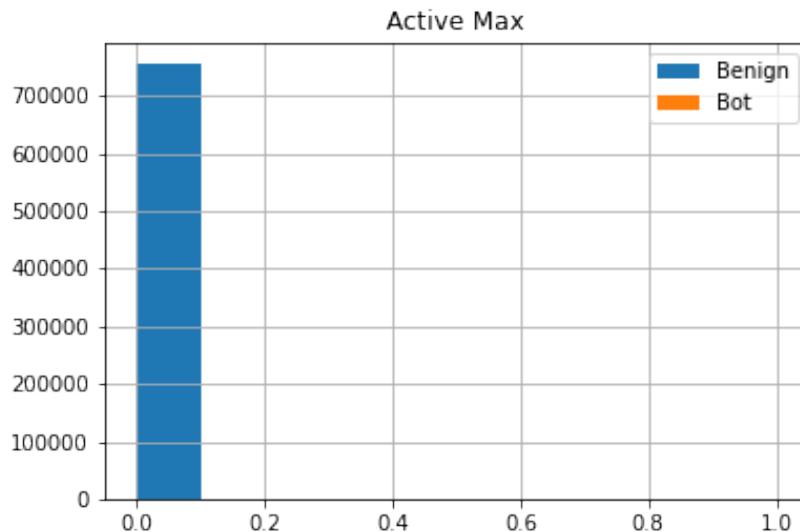


Figura 3.2: Feature che assume sempre lo stesso valore (0)

Tuttavia, analizzando attentamente gli histogrammi e i valori assunti dalle feature, abbiamo iniziato a individuarne alcune particolarmente discriminanti.

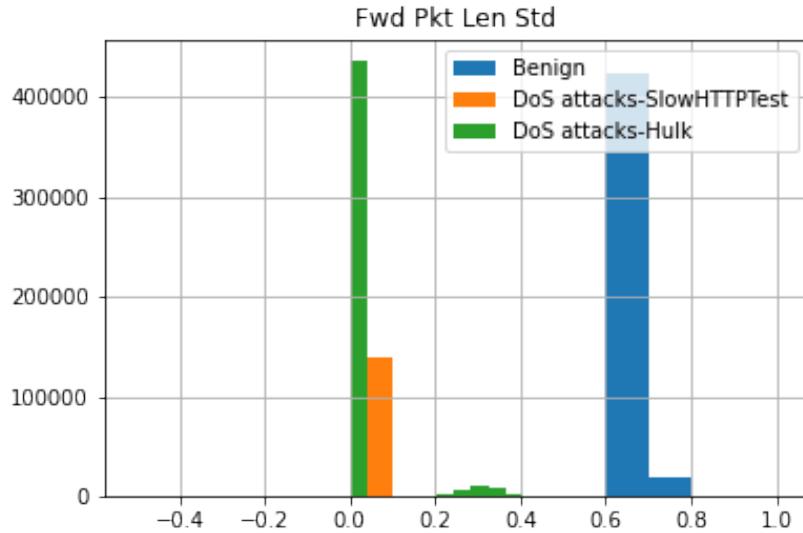


Figura 3.3: Feature discriminante

Successivamente, si è proseguito con l'applicazione di diversi algoritmi di feature selection per ottenere una maggiore sicurezza sulla selezione delle feature più importanti.

Il primo algoritmo utilizzato è **ANOVA**, implementato grazie all'utilizzo della libreria *scikit-learn*. Questo algoritmo ci consente di condurre un'analisi della varianza, selezionando le migliori feature sulla base di test statistici univariati.

Per ciascun test, viene calcolato il valore F, che ci permette di stabilire se il test è statisticamente significativo o meno.

In seguito, abbiamo usato un selettore basato su alberi per la feature selection.

Il modello **Random Forest** può essere utilizzato per calcolare l'importanza delle feature utilizzando il criterio *impurity-based*.

Questo criterio consente di selezionare le feature che contribuiscono a una divisione più accurata dei dati di input. Di conseguenza, le prime feature utilizzate dall'albero per suddividere i dati sono anche considerate le più importanti.

Abbiamo inoltre impiegato un altro algoritmo per la selezione delle feature, basato su un modello lineare.

Nello specifico, abbiamo addestrato un modello di **Support Vector Machine** (SVM). Dopo l'addestramento, abbiamo selezionato le feature con il punteggio più elevato, ovvero quelle che risultano più discriminanti per questo particolare modello di machine learning.

Infine, abbiamo utilizzato l'algoritmo **Recursive Feature Eliminator** (RFE). Questo algoritmo si avvale di uno stimatore esterno, ovvero un modello, che attribuisce pesi alle feature con l'obiettivo di eliminarle in modo ricorsivo.

L'algoritmo RFE seleziona le caratteristiche considerando insiemi sempre più piccoli in maniera ricorsiva. Inizialmente, il modello viene addestrato sull'insieme completo di feature e viene ottenuto uno score per ciascuna caratteristica utilizzando un attributo specifico del modello. Successivamente, le caratteristiche meno rilevanti vengono eliminate dall'attuale set di feature. Questa procedura viene ripetuta ricorsivamente sul set ridotto fino a raggiungere il numero desiderato di feature da selezionare.

Per implementare il RFE, abbiamo utilizzato un *decision tree classifier* come modello.

Al termine di ogni algoritmo sono state registrate le 10 feature più importanti all'interno di un file csv.

Confronto tra le feature

A questo punto del progetto, per ciascun file del dataset, abbiamo ottenuto **10** feature, ognuna con il suo punteggio. Dato l'intero dataset e il numero di file che lo compongono, abbiamo deciso di considerare solo le migliori **5** feature tra quelle ottenute in precedenza.

Per garantire una comparabilità dei punteggi assegnati alle varie feature, abbiamo normalizzato i valori in modo che rientrassero nell'intervallo tra zero e uno. Per fare ciò, abbiamo diviso tutti i punteggi per il valore più alto presente nel file CSV corrispondente.

Per alcuni algoritmi useremo tutte e cinque le migliori feature mentre per altri ne utilizzeremo solo le prime due.

Per identificare le migliori cinque feature abbiamo eseguito un conteggio del numero di volte in cui ogni feature compare negli altri file. Successivamente, abbiamo ordinato le feature in base alla loro frequenza e selezionato le prime dieci in classifica.

Per valutare ulteriormente queste feature, abbiamo sommato i punteggi che ogni feature ha ottenuto nei diversi file. Infine, abbiamo moltiplicato il punteggio totale di ciascuna feature per la sua frequenza.

Dopo aver ordinato i risultati ottenuti, abbiamo scelto le prime cinque feature come le migliori.

Al fine di effettuare un controllo visivo sulla qualità delle feature selezionate, abbiamo generato tutte le possibili combinazioni di due elementi senza ripetizione. Queste combinazioni sono state utilizzate per creare degli scatter plot utilizzando il nostro dataset.

Di seguito è riportato uno degli scatter plot ottenuti come esempio.

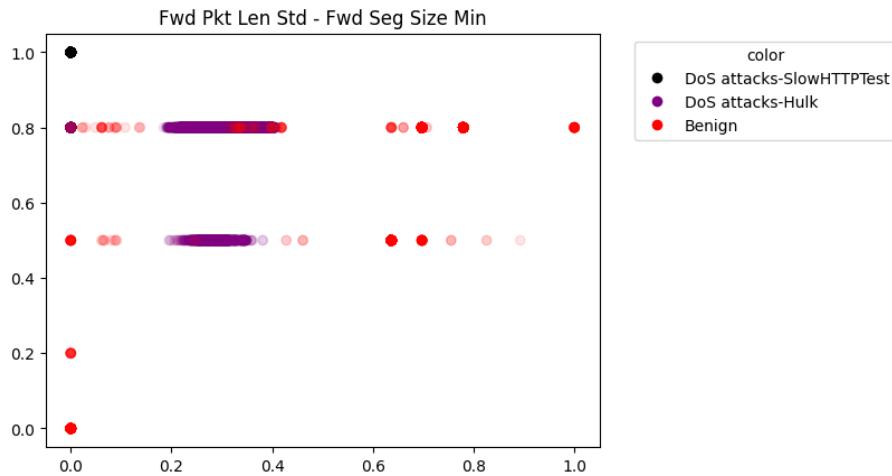


Figura 3.4: Esempio di scatterplot su feature significative

Infine, le feature selezionate sono state salvate in un nuovo file CSV.

3.2 Clustering

Per ragioni di tempistiche si è deciso di prendere, tra le feature individuate come più importanti, le prime due più significative. Tale scelta è stata adottata in quanto, per ogni file analizzato, il tempo medio di esecuzione è intorno ai 45 minuti. L'esecuzione di tutte le combinazioni delle feature tra le 5 più importanti avrebbe quindi non solo richiesto una grande mole di tempo ma avrebbe comunque portato alla generazione di grandi quantità di immagini, la maggior parte di queste non rilevanti al fine del progetto.



Figura 3.5: Tempo di esecuzione di ciascun algoritmo di clustering sul singolo file

Operazioni Preliminari

Innanzitutto, si seleziona una coppia di feature tra quelle più significative e si crea un nuovo dataset contenente solo le due feature e il campo label. Successivamente, il dataset viene suddiviso in due sottogruppi in base al valore del campo label.

Inoltre, affinchè gli algoritmi di clustering producano dei risultati affidabili e interpretabili bisogna effettuare delle ulteriori operazioni preliminari, nello specifico ridurre le dimensioni del dataset e bilanciare le classi.

L'esecuzione del clustering su un un enorme quantitativo di dati può richiedere molte risorse computazionali e tempo di calcolo.

La presenza di una classe contenente un numero significativamente maggiore di elementi rispetto ad un'altra, invece, induce gli algoritmi di clustering a sovrappesare un cluster rispetto ad un altro. Il bilanciamento del dataset permette di ottenere risultati rappresentativi e, in particolare nel nostro caso, generalizzabili.

```
MemoryError: Unable to allocate 4.83 TiB for an array with shape (663991560771,) and data type float64
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Figura 3.6: MemoryError sull'esecuzione del clustering senza riduzione del dataset

3.2.1 Scelta degli Algoritmi di Clustering

I metodi utilizzati per dividere in cluster i nostri dataset sono i seguenti:

- **K-Means:**

Algoritmo che forma i cluster in base alla distanza euclidea dei punti dei dati da n centroidi (dove n è il numero di cluster desiderati).

Nella fase di inizializzazione i centroidi vengono scelti casualmente, ogni iterazione si divide in due fasi:

- assegnazione: i punti vengono assegnati al cluster identificato dal centroide più vicino,
- scelta dei centroidi: si ricalcolano in base ai punti del proprio cluster.

L'algoritmo viene iterato finchè la posizione dei centroidi non si stabilizza.

Pro:

- semplicità: è un algoritmo relativamente semplice da comprendere e implementare,
- efficienza: è computazionalmente efficiente e può essere applicato a grandi set di dati,
- scalabilità: è in grado di gestire un numero elevato di punti dati e cluster,
- interpretazione dei risultati: i risultati possono essere facilmente interpretati visto che ogni cluster viene rappresentato da un centroide, il quale rappresenta il punto medio dei punti dati nel cluster,
- adatto per dati numerici: funziona bene con dati numerici e continui poiché si basa sulla distanza euclidea tra i punti dati.

Contro:

- scelta arbitraria del numero di cluster: è necessario specificare a priori il numero di cluster desiderato. La scelta di un numero errato può portare a una cattiva suddivisione dei dati o a una sovrapposizione tra cluster,
- sensibile alla scelta iniziale dei centroidi: dipende dalla scelta iniziale dei centroidi. Diverse inizializzazioni possono portare a risultati diversi,
- non adatto per cluster di forma irregolare o di dimensioni diverse: il K-means assume che i cluster abbiano forme sferiche e dimensioni simili, quindi potrebbe non funzionare bene con cluster complessi o di dimensioni molto diverse,
- sensibile ai dati anomali: la presenza di dati anomali può influenzare significativamente i risultati dell'algoritmo visto che questi possono spostare i centroidi e influire la suddivisione dei cluster,
- non garantisce una soluzione ottimale globale: l'algoritmo può convergere a minimi locali, che tuttavia potrebbero non rappresentare la soluzione di clustering ottimale per il problema.

- **DBSCAN:**

Algoritmo che utilizza la densità dei punti per definire i cluster. Utilizza due parametri: raggio ϵ , definisce la distanza massima per considerare due punti vicini, ed N , numero minimo di punti che devono essere all'interno del raggio di un punto per considerarlo core point.

Nella fase di inizializzazione i punti vengono suddivisi in core point, border point (si trovano entro il raggio di un core point), noise point (isolati).

Ogni iterazione si divide in due fasi:

- creazione dei cluster: ad ogni core point viene assegnato un cluster,
- espansione dei cluster: se due core point condividono almeno un border point allora vengono uniti in un unico cluster.

L'algoritmo viene iterato finché ci sono cluster da unire.

Pro:

- non richiede la specificazione del numero di cluster: a differenza del K-means, DBSCAN non richiede di specificare a priori il numero di cluster desiderato ma trova autonomamente i cluster presenti nei dati sulla base della densità dei punti,
- gestione di cluster di forma e dimensioni diverse: è in grado di identificare cluster di forma irregolare e di dimensioni diverse ed è particolarmente efficace nell'individuare cluster di densità variabile,

- capacità di identificare punti di rumore: può rilevare e contrassegnare i punti di rumore che non appartengono a nessun cluster specifico. Questo è utile per la gestione dei dati anomali,
- robusto rispetto ai dati iniziali e alla presenza di outlier: è meno sensibile alla scelta iniziale dei punti dati o dei centroidi e può funzionare bene anche in presenza di dati anomali o punti isolati,
- scalabilità: è generalmente efficiente e può essere applicato a grandi set di dati.

Contro:

- sensibilità alla scelta dei parametri: richiede di specificare due parametri, quali il raggio di vicinanza (ϵ) e il numero minimo di punti in un vicinato ($MinPts$). La scelta errata di questi parametri può influenzare significativamente i risultati del clustering,
- difficoltà con la densità uniforme: potrebbe non funzionare bene con dati in cui la densità è uniforme in tutto lo spazio poiché potrebbe identificare erroneamente tutto il dataset come rumore o come un unico grande cluster,
- problemi di scala: non funziona bene con dati di scala diversa poiché la scelta del raggio di vicinanza può essere problematica in presenza di dimensioni diverse dei dati.

• **Clustering gerarchico agglomerativo:**

Questo algoritmo utilizza una misura di distanza/somiglianza per creare nuovi cluster.

Inizialmente ogni punto viene considerato come un singolo cluster. Ad ogni iterazione, i cluster simili si fondono con altri cluster fino a formare un cluster o K cluster. Gli step effettuati dall'algoritmo possono essere riassunti come segue:

- step 1: calcola la matrice di prossimità utilizzando una particolare metrica di distanza
- step 2: ogni punto dati viene assegnato a un cluster
- step 3: unire i cluster in base a una metrica per la somiglianza tra i cluster
- step 4: aggiornare la matrice delle distanze
- step 5: ripetere gli step 3 e 4 finché non rimane un solo cluster

Pro:

- struttura gerarchica: produce una struttura gerarchica di cluster, nota anche come dendrogramma, che mostra le relazioni di similarità tra i cluster e questo consente una migliore comprensione della struttura dei dati,

- flessibilità nella scelta del numero di cluster: consente di esplorare diverse partizioni dei dati consentendo di scegliere il numero di cluster in base alle proprie esigenze o di estrarre cluster di varie dimensioni dalla gerarchia,
- interpretazione dei risultati: la struttura gerarchica fornisce un’interpretazione intuitiva dei risultati. I cluster vicini sulla gerarchia sono più simili tra loro rispetto a quelli distanti, consentendo di identificare facilmente i gruppi simili o le strutture di sottoclassi,
- non richiede specifiche iniziali: non richiede la specificazione del numero di cluster a priori o l’inizializzazione dei centroidi, infatti l’algoritmo costruisce automaticamente la gerarchia di cluster basandosi sulla similarità dei dati,
- scalabilità: può essere applicato a set di dati di grandi dimensioni ed è relativamente scalabile.

Contro:

- costo computazionale: può essere computazionalmente costoso, specialmente con un gran numero di punti dati. Il tempo di esecuzione aumenta in modo esponenziale con la dimensione dei dati,
- sensibilità alla metrica di similarità/distanza: la scelta della metrica di similarità o distanza può influire significativamente sui risultati, infatti la scelta di una metrica inappropriata può portare a cluster errati o a una gerarchia di cluster non rappresentativa,
- difficoltà nella gestione di grandi dataset: può incontrare difficoltà nella gestione di grandi dataset a causa dei problemi di scalabilità e del costo computazionale,
- non adatto per dati con cluster di forme e dimensioni complesse: questo algoritmo assume che i cluster abbiano forme e dimensioni simili, quindi potrebbe non funzionare bene con dati contenenti cluster di forme irregolari o di dimensioni molto diverse,
- irreversibilità delle decisioni: una volta presa una decisione sulla combinazione o sulla divisione dei cluster durante la costruzione della gerarchia, questa decisione è irreversibile, infatti potrebbe non essere possibile correggere o aggiornare le decisioni in seguito.

• **Self-Organizing Map (SOM):**

Utilizza una rete neurale per assegnare i dati ad una griglia di neuroni caratterizzate da pesi (inizialmente casuali), ognuna delle quali rappresenta un cluster. Ogni iterazione viene scelto casualmente un punto dei dati e si procede come segue:

- selezione della cella: viene selezionato il neurone con i pesi più simili a quelli del dato corrente, corrisponde all’assegnare un punto ad un cluster.

- aggiornamento dei pesi: i pesi dei neuroni vengono aggiornati in base all'esito dell'assegnazione, questo permette alla griglia di adattarsi ai dati in ingresso

Pro:

- visualizzazione dei dati: offrono una rappresentazione visiva dei dati di input consentendo di esplorare e comprendere meglio la struttura intrinseca dei dati,
- rilevamento di pattern e cluster: sono in grado di rilevare pattern e cluster nei dati senza richiedere una specifica conoscenza a priori sul numero di cluster desiderato,
- conservazione delle relazioni topologiche: preservano le relazioni topologiche tra i dati di input nella loro mappatura bidimensionale; ciò significa che i dati simili saranno mappati in aree vicine sulla griglia SOM,
- scalabilità: sono relativamente scalabili e possono gestire grandi set di dati
- riduzione della dimensionalità: possono essere utilizzate per ridurre la dimensionalità dei dati, mappando gli spazi ad alta dimensione in uno spazio bidimensionale o tridimensionale.

Contro:

- interpretazione dei risultati: può essere soggettiva e richiede un'analisi attenta per comprendere la distribuzione dei dati sulla griglia SOM,
- sensibilità ai parametri: richiedono la specificazione di parametri come la dimensione della griglia, il tasso di apprendimento e il raggio di vicinanza iniziale. La scelta errata di questi parametri può influenzare significativamente i risultati del clustering,
- apprendimento lento: a differenza di altri algoritmi di clustering, le SOM possono richiedere tempo per convergere verso una mappatura stabile,
- sensibili alla dimensionalità: le SOM possono incontrare difficoltà quando si lavora con dati ad alta dimensionalità. In tali casi può essere necessario ridurre la dimensionalità dei dati prima di applicare l'algoritmo,
- sono richiesti dati completi: le SOM richiedono dati completi per il processo di addestramento perché la presenza di valori mancanti può influire sulla qualità dei risultati.

3.2.2 Applicazione degli Algoritmi di Clustering

Per realizzare i metodi appena descritti sono stati utilizzati i moduli forniti dalla libreria *sklearn*. Per ogni algoritmo si è proceduto come segue:

- valutazione e scelta dei parametri ottimali da dare all'algoritmo.

- tramite le funzioni fit e predict (o 'fit_predict') il modello di clustering viene addestrato sui dati per poter poi assegnare i cluster.
- si associa il cluster i-esimo con la classe i-esima.
- vengono calcolati l'accuracy, la precision e recall per valutare le prestazioni dell'algoritmo corrente.

Di seguito approfondiremo le varie operazioni svolte.

Valutazione e scelta dei parametri ottimali da dare all'algoritmo

Gli algoritmi di clustering richiedono la definizione di alcuni iper-parametri per funzionare correttamente. Tuttavia, in alcuni casi, i valori di alcuni di questi iper-parametri possono essere già noti a priori.

Nel caso dell'algoritmo K-means, ad esempio, l'unico iper-parametro da definire è il numero di cluster che si desidera trovare. Poiché il dataset è etichettato questa informazione la abbiamo già disponibile. Nel nostro caso, ci sono uno o due attacchi per file e il traffico benevolo, il che implica la presenza di due o tre cluster in un file.

Un'osservazione simile si può fare per gli algoritmi SOM e per il clustering gerarchico agglomerativo.

Anche per questi algoritmi, infatti, possiamo definire il numero di cluster a priori, poiché conosciamo il numero di attacchi per file e il traffico benevolo.

Per quanto riguarda l'algoritmo DBSCAN, invece, abbiamo utilizzato l'algoritmo **elbow method** per determinare i valori degli iper-parametri *eps* e *min_samples*.

L'elbow method coinvolge l'analisi della curva delle distanze medie per vari valori di *eps*. In particolare, si traccia la curva e si cerca il punto in cui si verifica un "gomito" o una brusca diminuzione delle distanze medie. Questo punto può essere considerato come un buon valore per *eps*.

Per quanto riguarda *min_samples*, invece, si può scegliere un valore che rappresenti il numero minimo di campioni richiesti per formare un cluster.

L'elbow method è stato implementato utilizzando la libreria *sklearn* di *Python*, importando la classe **NearestNeighbors** dalla libreria *sklearn.neighbors*. Questa classe viene utilizzata per calcolare le distanze tra i punti nel dataset.

Più nel dettaglio si è scelto di trovare i 5 punti più vicini a ciascun punto nel dataset per poi andare a calcolare internamente le distanze tra i punti e gli indici dei punti più vicini per ciascun punto nel dataset.

In seguito, le distanze vengono ordinate per ottenerne una rappresentazione ordinata. Infine, viene tracciato un grafico delle distanze attraverso la cui osservazione si può andare a scegliere il valore migliore di *eps*.

Attivazione degli algoritmi di clustering

Una volta preparato il dataset e impostati i valori degli iper-parametri, si può procedere con l'attivazione degli algoritmi di clustering utilizzando i metodi *fit* e *predict*, oppure il metodo *fit_predict*. Questi metodi consentono di calcolare i centri dei cluster e assegnare ogni elemento al cluster corrispondente.

Il metodo **fit** viene utilizzato per addestrare l'algoritmo di clustering utilizzando il dataset fornito. Questo metodo calcola i centri dei cluster basandosi sui dati di input e li memorizza per l'utilizzo successivo.

Ad esempio, con l'algoritmo K-means, fit calcola i centroidi dei cluster.

Una volta che l'algoritmo è stato addestrato, è possibile utilizzare il metodo **predict** per assegnare ciascun elemento del dataset al cluster corrispondente.

Questo metodo utilizza i centri dei cluster precedentemente calcolati per effettuare le assegnazioni.

In alternativa, il metodo **fit_predict** può essere utilizzato per eseguire entrambe le operazioni contemporaneamente.

Esso addestra l'algoritmo di clustering utilizzando il dataset fornito e restituisce l'assegnazione dei cluster per ogni elemento.

Associazione del cluster *i*-esimo con la classe *i*-esima

Prima di valutare i cluster ottenuti, è necessario effettuare l'allineamento tra i cluster e le rispettive classi di riferimento.

Gli algoritmi di clustering assegnano i cluster in modo casuale, il che significa che non è possibile imporre a priori che un determinato cluster corrisponda a una specifica classe e questo può causare un errore nella fase di valutazione dei risultati.

Di seguito è riportato un esempio in cui è stato generato un dataset di test.

È possibile osservare che la clusterizzazione è stata eseguita in modo perfetto, ma a causa dell'assegnazione casuale dei cluster alle classi, valutando i risultati, otteniamo un'accuratezza pari a 0.33 periodico.

Ciò accade perché solo uno dei tre cluster è stato associato correttamente alla classe corrispondente.

Per affrontare questa problematica, si è optato per un'ulteriore soluzione dopo l'operazione di clustering: rinominare i cluster in base alla classe di riferimento corrispondente.

Per individuare la classe di riferimento è stato eseguito un conteggio del numero di elementi presenti in ciascun cluster, suddivisi per classe. Di conseguenza, la classe con il maggior numero di elementi in quel cluster è stata designata come la classe corrispondente per quel cluster.

Tuttavia, questa soluzione ha portato ad affrontare il problema dello sbilanciamento

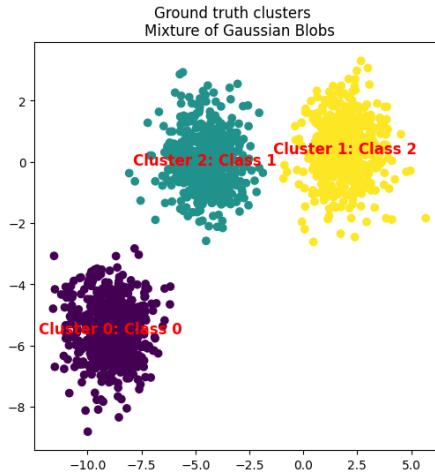


Figura 3.7: Assegnazione casuale cluster

delle classi. Sulla base della maggioranza, la classe più numerosa (nel caso specifico, quella benigna) risultava essere associata a tutti i cluster o a due di essi.

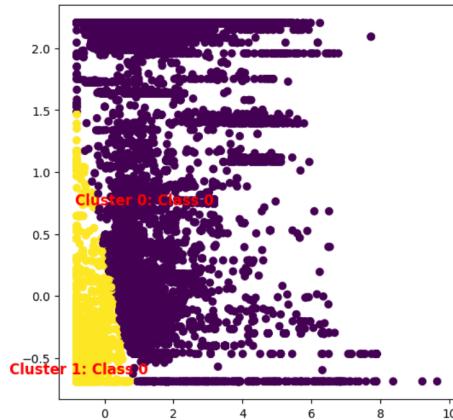


Figura 3.8: Assegnazione cluster senza bilanciamento

Per questo motivo è stato necessario effettuare un'operazione di bilanciamento delle classi.

La riduzione delle dimensioni del dataset ha consentito anche l'utilizzo di algoritmi di clustering molto computazionalmente onerosi, come DBSCAN e clustering gerarchico agglomerativo, che altrimenti avrebbero causato errori nella macchina virtuale a causa delle limitate risorse disponibili.

Per eseguire l'allineamento dei cluster con le rispettive classi si è proceduto acquisendo gli indici delle righe corrispondenti a un determinato cluster. Questo è stato fatto per tutti i cluster.

Successivamente, ai valori di questi indici è stato assegnato il valore del cluster corrispondente alla classe. In questo modo siamo riusciti a effettuare con successo l'ope-

zione di allineamento dei cluster con le rispettive classi.
Di seguito riportiamo un esempio del risultato ottenuto.

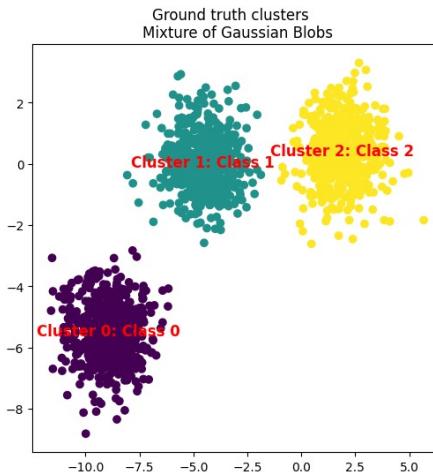


Figura 3.9: Allineamento cluster - classe

Calcolo delle metriche per valutare le prestazioni dell'algoritmo di clustering

Per valutare le prestazioni degli algoritmi di clustering, sono state utilizzate metriche supervisionate che forniscono una valutazione obiettiva dei risultati ottenuti. Le metriche prese in considerazione sono **Precision**, **Recall** e **Accuracy**.

La **Precision** misura l'omogeneità del cluster. La precisione del cluster i rispetto alla classe j è data da:

$$\text{Precision} = \frac{N_{ij}}{N_i}$$

dove N_{ij} corrisponde al numero di elementi di classe j nel cluster i e N_i corrisponde al numero di elementi del cluster i .

Se il cluster i è composto da soli elementi della classe j allora questo rapporto vale 1, quindi il cluster è omogeneo. Inversamente, se il cluster i è composto da elementi di più classi allora il valore della Precision si abbassa.

La **Recall** misura la capacità di rappresentare una classe e viene calcolata come segue:

$$\text{Recall} = \frac{N_{ij}}{N_j}$$

dove dove N_{ij} corrisponde al numero di elementi di classe j nel cluster i e N_j corrisponde al numero di elementi della classe j .

L' **Accuracy** indica la percentuale di istanze correttamente classificate rispetto al totale delle istanze nel dataset.

Formalmente, l'accuratezza ha la seguente definizione:

$$\text{Accuracy} = \frac{\text{Number_of_correct_predictions}}{\text{Total_number_of_predictions}}$$

Tuttavia per il clustering il modo per calcolare l'accuracy è quello di trovare trovare la migliore corrispondenza tra le etichette della classe e le etichette del cluster, quindi è definita da:

$$\text{accuracy}(y, \hat{y}) = \max_{\text{perm} \in P} \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{1}(\text{perm}(\hat{y}_i) = y_i)$$

dove P è l'insieme di tutte le permutazioni in $[1;K]$ e K è il numero di cluster.

Le suddette metriche sono state selezionate poiché il dataset iniziale era già fornito con le etichette.

3.3 Principal Component Analysis (PCA)

Un’ulteriore tecnica utilizzata è la **Principal Component Analysis** (PCA). La PCA è una tecnica statistica che consente di trasformare un insieme di variabili correlate in un nuovo insieme di variabili non correlate, chiamate componenti principali.

Le componenti principali sono combinazioni lineari delle variabili originali che vengono ordinate in modo tale che la prima componente principale indichi la direzione lungo la quale i dati presentano la maggiore variazione, seguita dalla seconda componente principale e così via.

Viene utilizzata per via dei suoi innumerevoli vantaggi: consente di ridurre la dimensionalità dei dati mantenendo al contempo la maggior parte delle informazioni rilevanti, semplificando, in questo modo, l’analisi e riducendo i tempi di calcolo.

Inoltre, permette di proiettare i dati su un numero inferiore di dimensioni (solitamente due o tre) per visualizzarli in uno spazio più semplice da interpretare. Questa visualizzazione può aiutare a identificare pattern, cluster o relazioni tra i dati, facilitando la comprensione e l’interpretazione. Infine, può essere utilizzata per comprimere i dati riducendo la loro dimensione senza perdere informazioni significative e questo può essere utile quando si lavora con grandi quantità di dati.

Le componenti principali si trovano come segue:

- si calcola la matrice di covarianza che misura la relazione lineare tra le variabili,
- si calcolano gli autovettori e gli autovalori della matrice di covarianza dei dati del dataset originale. Gli autovettori rappresentano la direzione dei nuovi assi nello spazio dei dati, mentre gli autovalori indicano l’importanza relativa di ciascuna componente principale. Inoltre, gli autovettori vengono normalizzati in modo che abbiano una lunghezza unitaria
- si selezionano gli n autovettori (2 nel nostro caso) con gli autovalori più alti. Questi forniscono la quantità di varianza trasportata in ogni componente principale.
- si riformulano i dati lungo gli assi dei componenti principali: questo può essere fatto moltiplicando la matrice trasposta dei dati del dataset originale per la matrice trasposta contenente gli n autovettori selezionati. In questo modo si è ridotta la dimensionalità dei dati, mantenendo una quantità significativa di informazione. Infine, i dati vengono proiettati sulle componenti principali selezionate per ottenere le nuove variabili trasformate. Questa proiezione consente di rappresentarli in un nuovo spazio a dimensionalità inferiore.

Capitolo 4

Risultati

In questo capitolo andremo a descrivere i risultati ottenuti applicando diversi algoritmi di clustering.

Dato il lungo tempo di esecuzione richiesto dagli algoritmi di clustering abbiamo deciso di utilizzare solo le prime **due** feature più significative per ogni file, così come per il dataset complessivo, ottenuto dall'unione di tutti gli altri dataset.

Abbiamo applicato l'algoritmo **PCA** al dataset completo, composto da tutte e cinque le feature più importanti. Questo ha generato un nuovo dataset con due nuove feature.

Per ogni file, presentiamo, inoltre, gli **scatter plot** che mostrano sia il dataset originale che il dataset suddiviso in vari cluster dall'algoritmo di clustering. Riportiamo anche le due feature più importanti utilizzate per eseguire gli algoritmi.

Infine, mostriamo i risultati ottenuti applicando l'algoritmo PCA sul dataset del file.

Abbiamo incluso una tabella per ciascun algoritmo, nella quale sono riportati i valori delle metriche. Inoltre, sono inclusi anche gli scatterplot e i valori ottenuti applicando gli algoritmi di clustering a due soli cluster, "benigno" e "maligno". Se il dataset di partenza ha solo due cluster, la tabella conterrà un solo una sola riga e vi sarà un solo scatterplot per ciascun algoritmo di clustering.

4.1 Friday-02-03-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti due tipologie di cluster, **Benign** e l'attacco **Bot**. Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature. Prima di eseguire il clustering, il dataset relativo a tale file è stato ridotto dell'80% oltre che effettuare il bilanciamento del dataset.

Le due feature utilizzate sono **Init Bwd Win Byts** e **Fwd Pkts_s**.

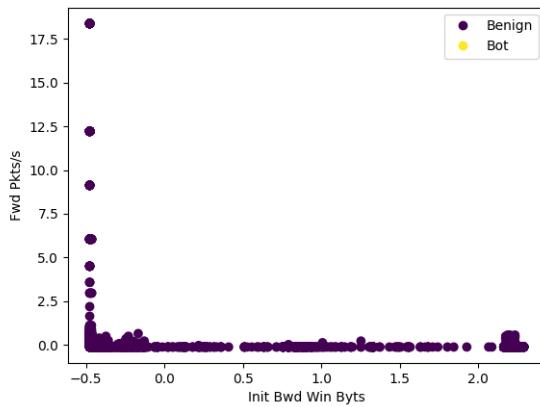


Figura 4.1: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

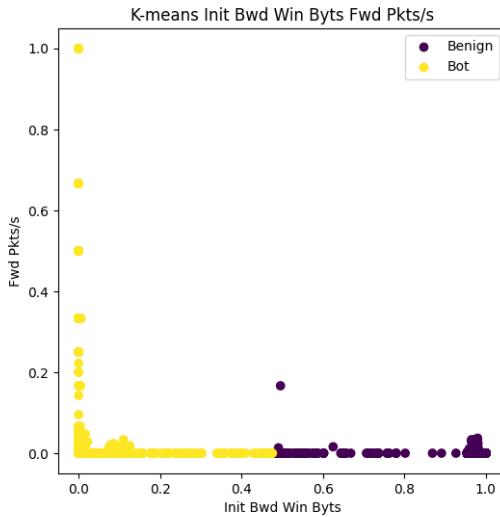


Figura 4.2: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.62	0.62	0.78

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

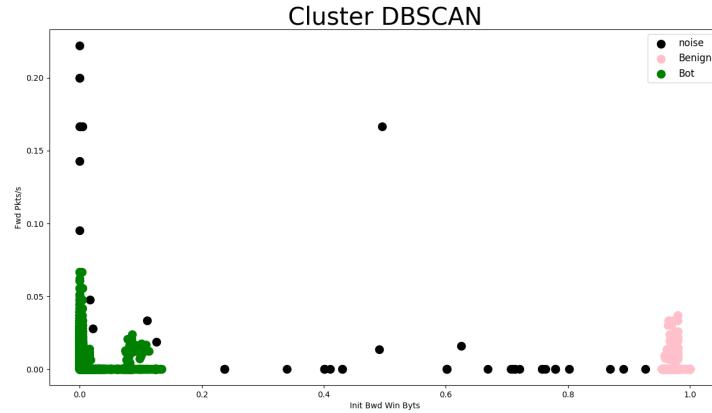


Figura 4.3: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
2	0.61	0.06	0.08

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

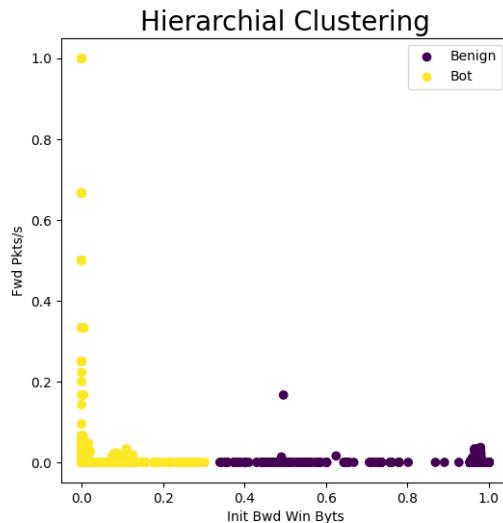


Figura 4.4: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
2	0.62	0.90	0.62

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

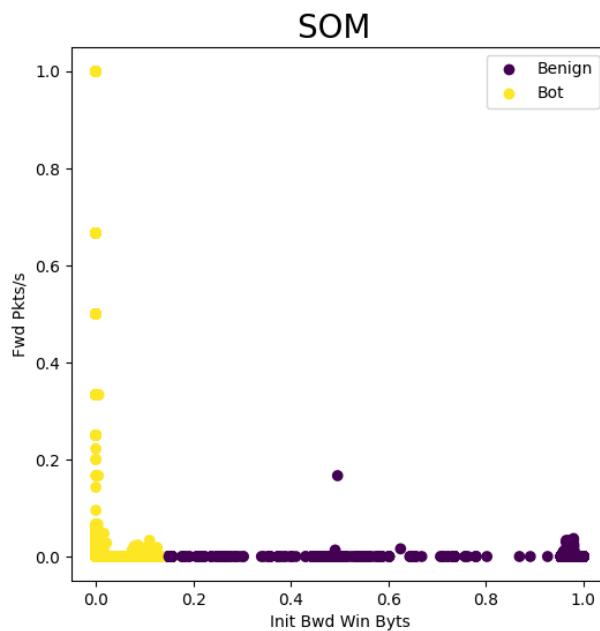


Figura 4.5: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
2	0.62	0.62	0.78

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

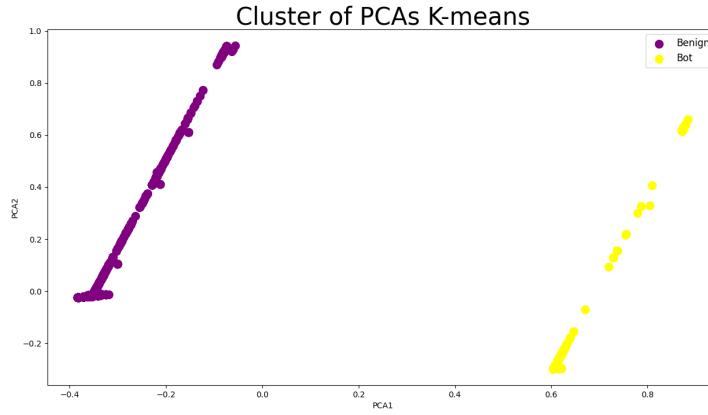


Figura 4.6: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.66	0.66	0.68

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

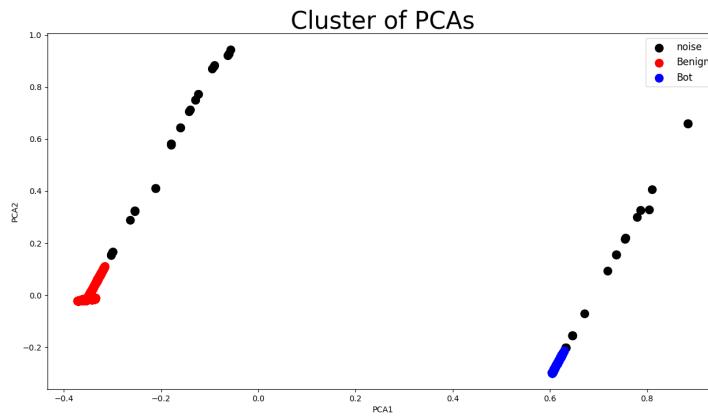


Figura 4.7: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
2	0.61	0.05	0.06

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

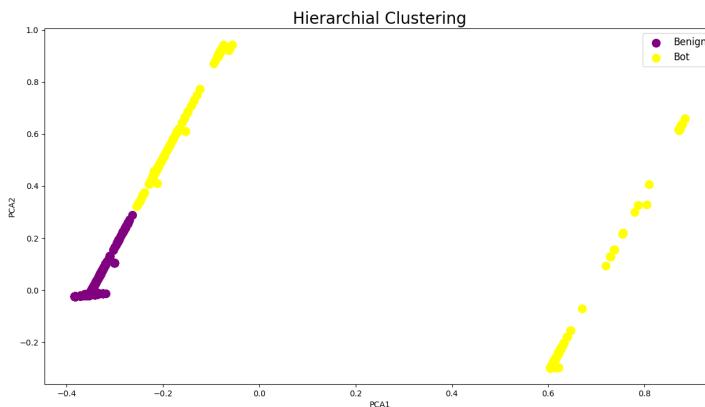


Figura 4.8: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
2	0.62	0.62	0.63

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

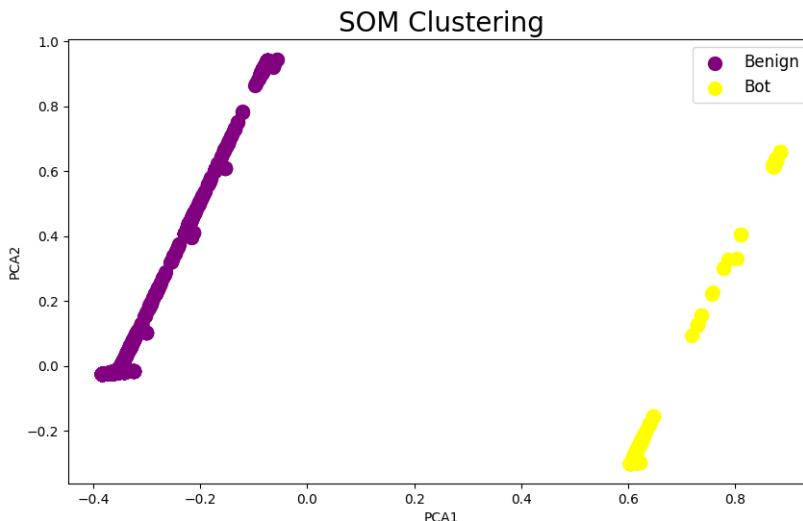


Figura 4.9: Risultati ottenuti con clustering SOM

Num. cluster	Accuracy	Precision	Recall
2	0.66	0.66	0.68

4.2 Friday-16-02-2018_TrafficForML_CICflowMeter

All'interno di questo file sono presenti tre tipologie di cluster, **Benign**, l'attacco **DDoS Attacks - SlowHTTPTest** e l'attacco **DDoS Attacks - Hulk**. Oltre al clustering effettuato considerando il flusso benigno ed i singoli flussi relativi ad attacchi, si è optato anche per un clustering binario (con distinzione tra flusso benigno e maligno), al fine di analizzare eventuali differenze tra le due analisi. Prima di eseguire il clustering, il dataset relativo a tale file è stato ridotto del 70% oltre che effettuare il bilanciamento del dataset.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Fwd Pkt Len Std** e **Fwd Seg Size Min**.

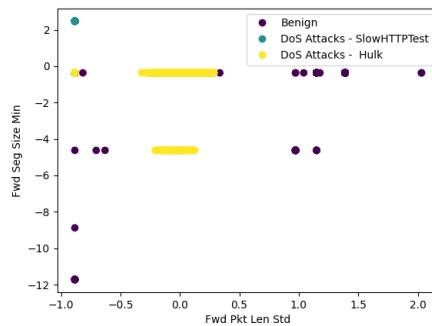


Figura 4.10: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

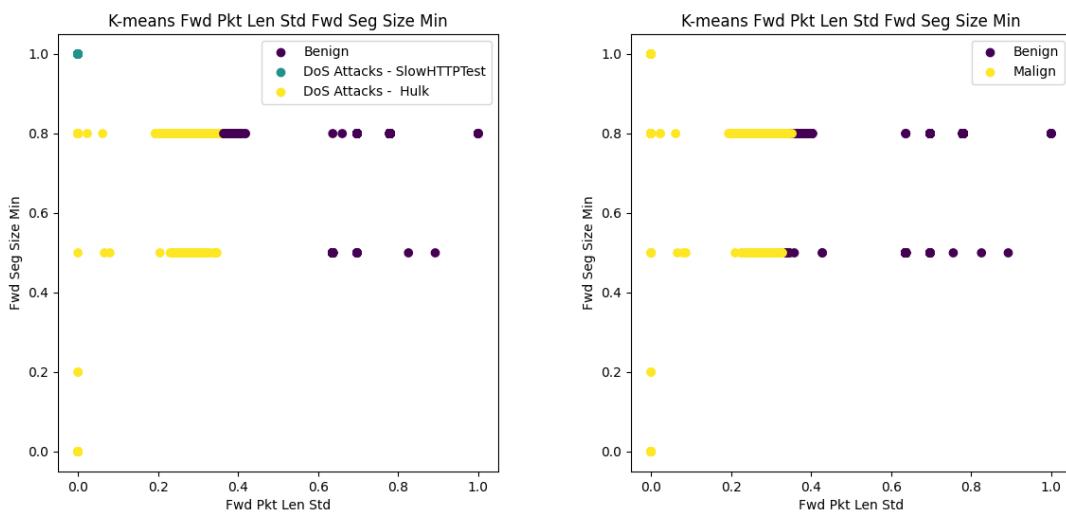


Figura 4.11: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

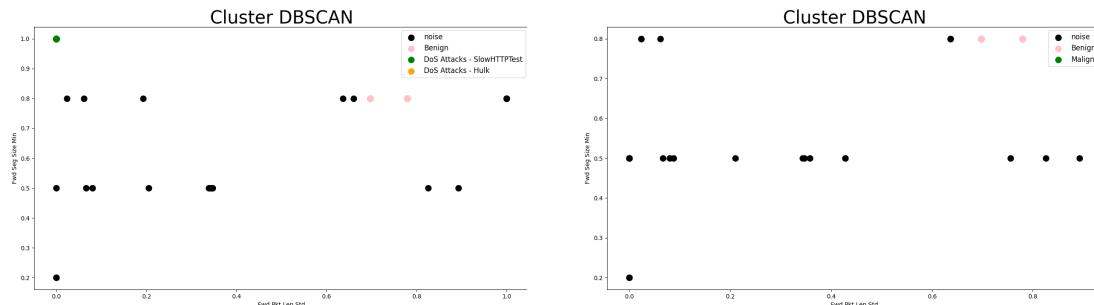


Figura 4.12: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
3	0.66	0.22	0.22
2	0.49	0.09	0.09

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

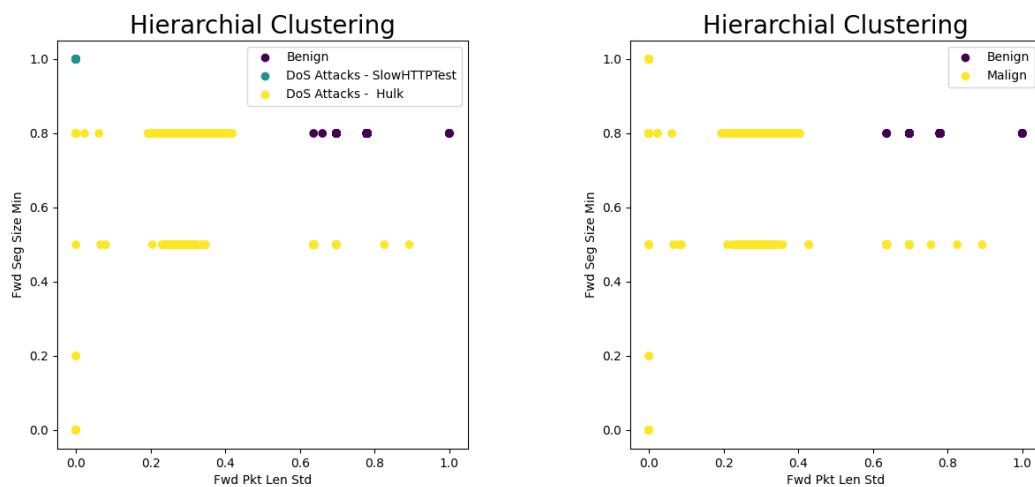


Figura 4.13: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

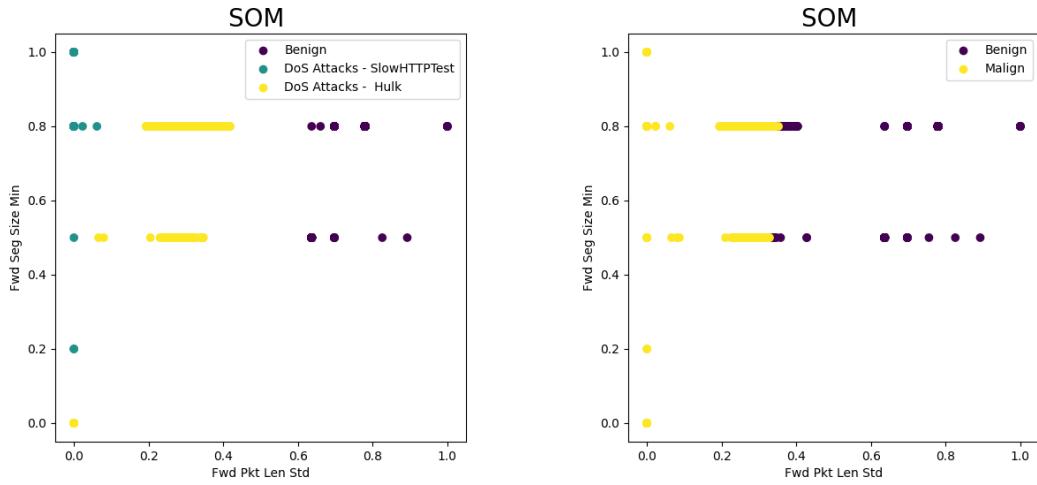


Figura 4.14: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
3	0.68	0.68	0.83
2	0.99	0.99	0.99

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

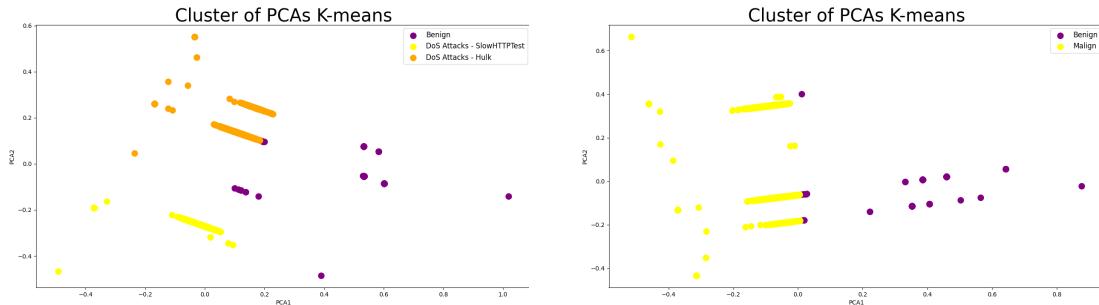


Figura 4.15: Risultati ottenuti con Kmeans

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

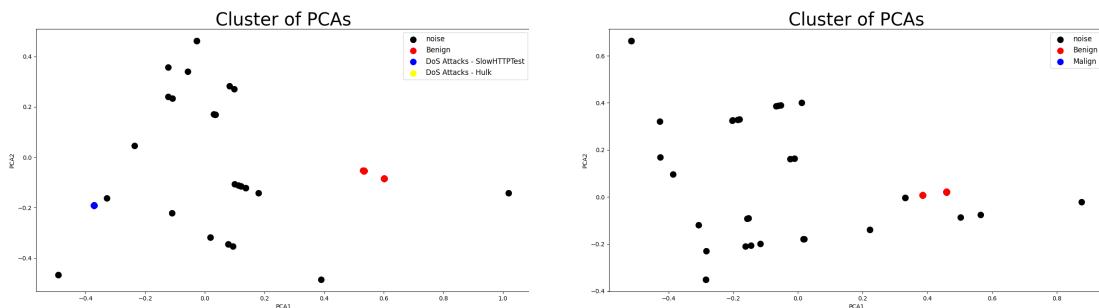


Figura 4.16: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
3	0.66	0.2	0.2
2	0.49	0.08	0.08

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

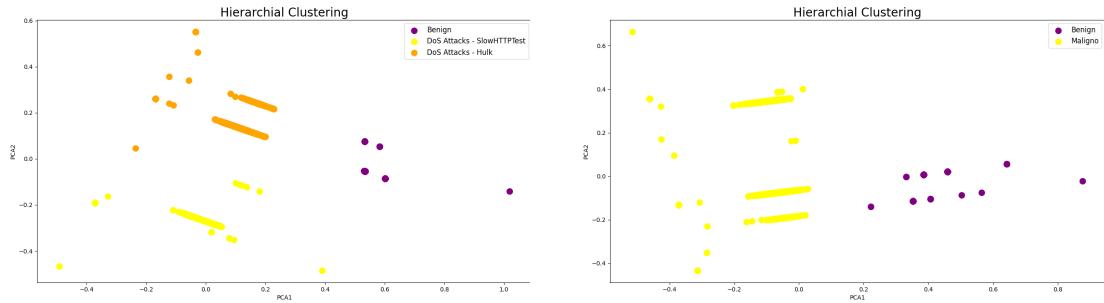


Figura 4.17: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.99	0.99	0.99

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

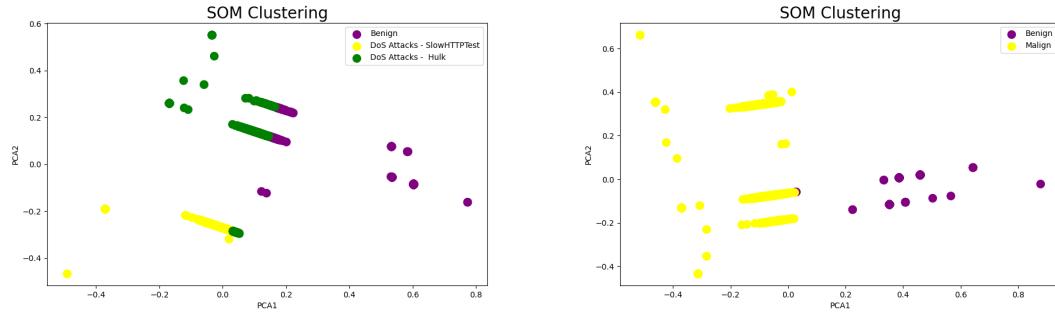


Figura 4.18: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.99	0.99	0.99

4.3 Friday-23-02-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti quattro tipologie di cluster, **Benign**, l'attacco **Brute Force - Web**, l'attacco **Brute Force - XSS** e l'attacco **SQL Injection**. Oltre al clustering effettuato considerando il flusso benigno ed i singoli flussi relativi ad attacchi, si è optato anche per un clustering binario (con distinzione tra flusso benigno e maligno), al fine di analizzare eventuali differenze tra le due analisi.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Fwd Win Byts** e **Bwd Pkt Len Max**.

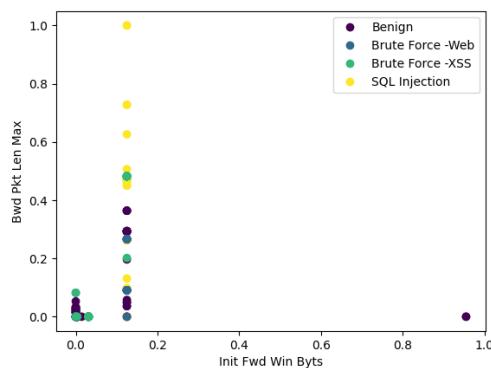


Figura 4.19: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

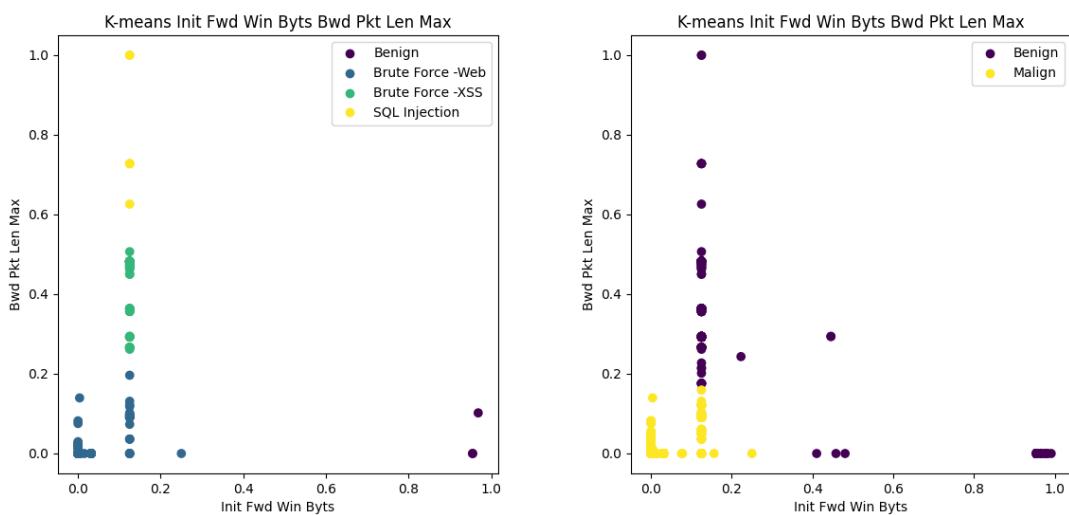


Figura 4.20: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
4	0.39	0.39	0.36
2	0.55	0.55	0.76

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

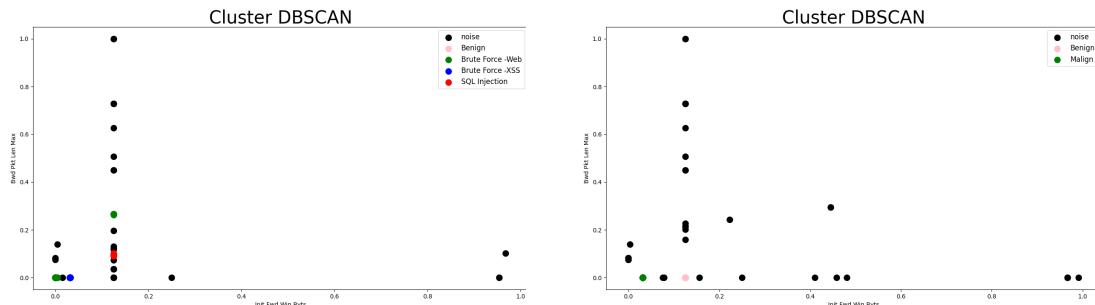


Figura 4.21: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
4	0.32	0.14	0.11
2	0.30	0.04	0.08

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

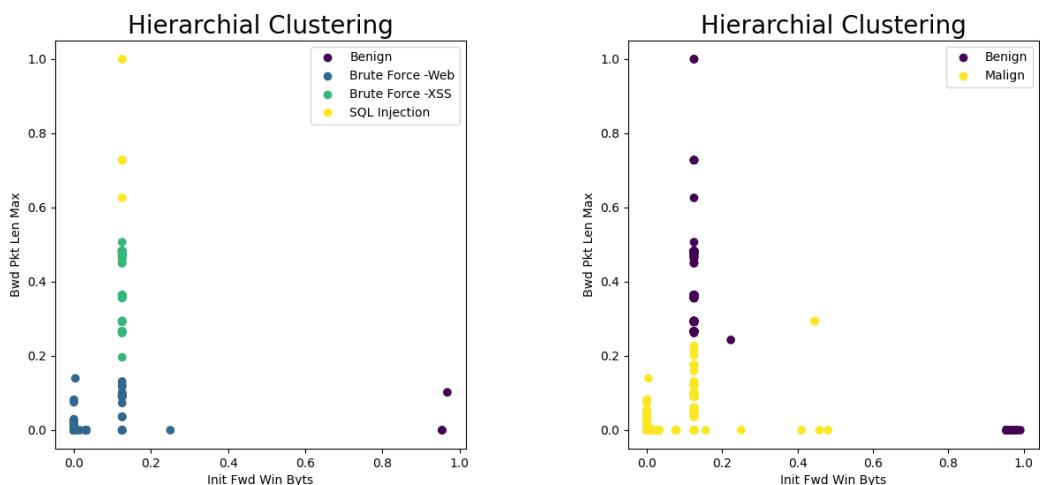


Figura 4.22: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
4	0.39	0.39	0.46
2	0.55	0.55	0.76

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

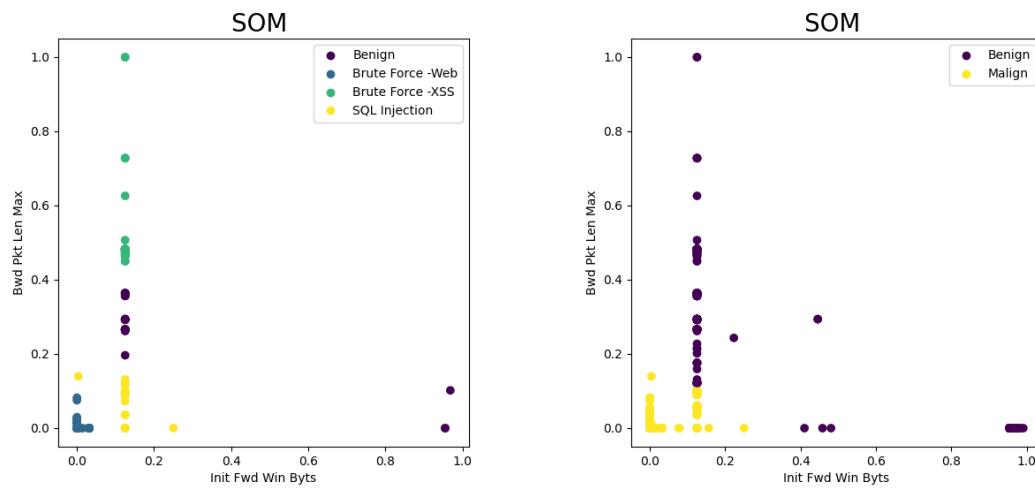


Figura 4.23: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
4	0.37	0.37	0.31
2	0.53	0.53	0.53

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

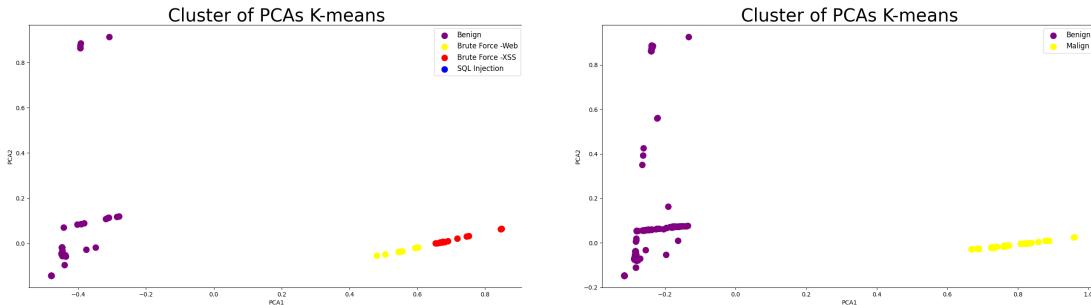


Figura 4.24: Risultati ottenuti con Kmeans

Num. cluster	Accuracy	Precision	Recall
4	0.44	0.44	0.37
2	0.63	0.63	0.66

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

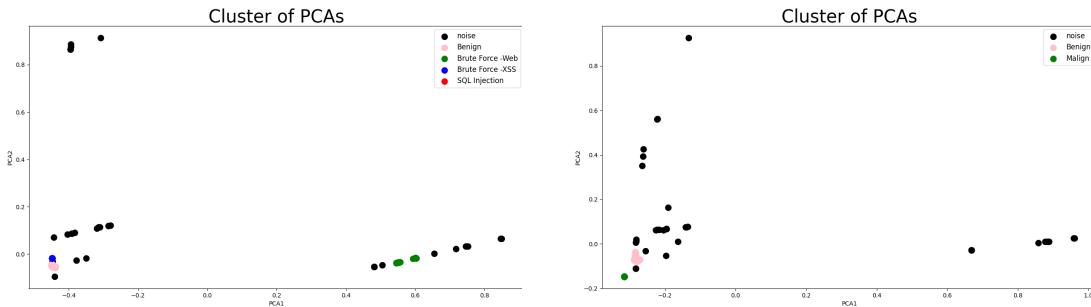


Figura 4.25: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
4	0.37	0.25	0.30
2	0.38	0.06	0.12

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

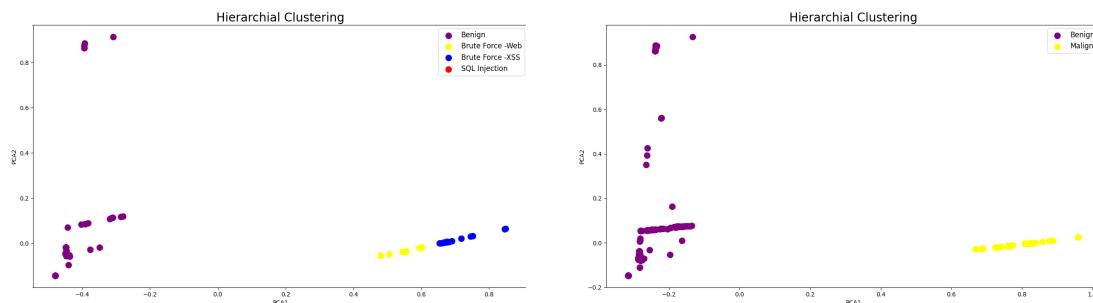


Figura 4.26: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
4	0.44	0.44	0.37
2	0.63	0.63	0.66

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

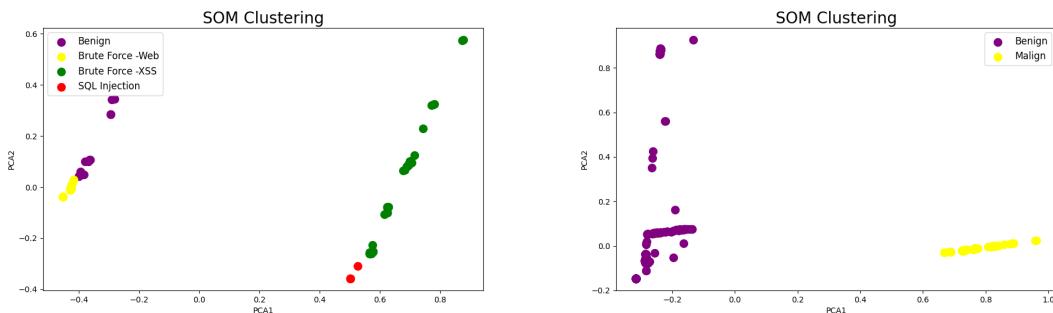


Figura 4.27: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
4	0.44	0.44	0.37
2	0.63	0.63	0.66

4.4. THUESDAY-20-02-2018_TRAFFICFORML_CICFLOWMETER45

4.4 Tuesday-20-02-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti due tipologie di cluster, **Benign** e l'attacco **DDoS Attack Loic HTTP**. Prima di eseguire il clustering, il dataset relativo a tale file è stato ridotto del 90% oltre che effettuare il bilanciamento del dataset.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Flow IAT Min** e **RST Flag Cnt**.

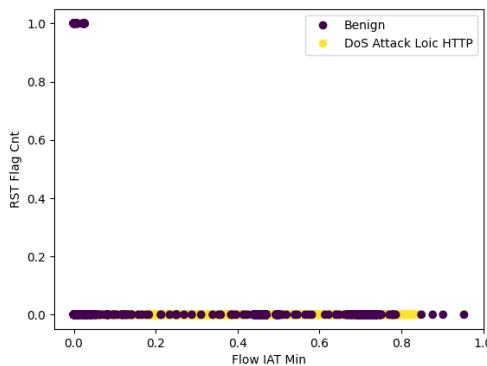


Figura 4.28: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

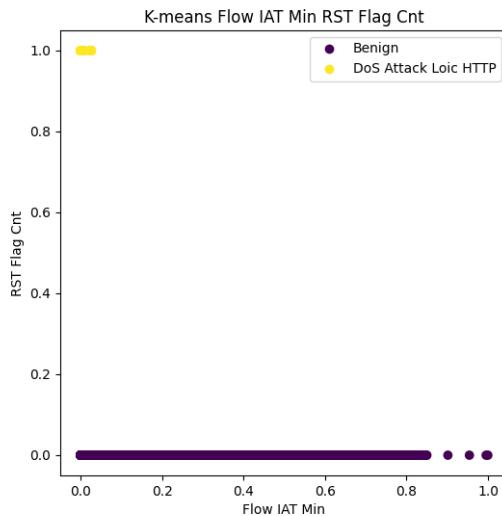


Figura 4.29: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.65	0.65	0.67

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

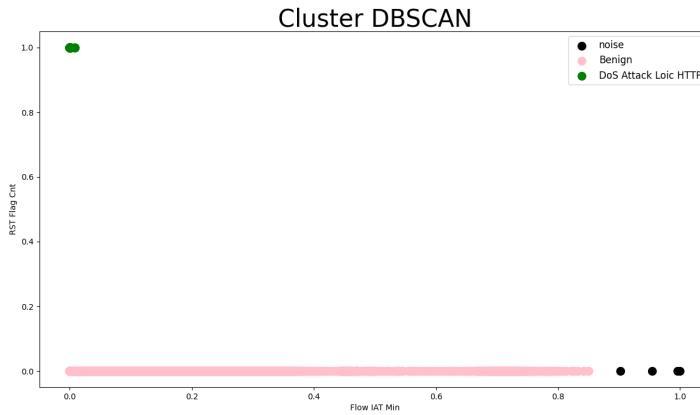


Figura 4.30: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
2	0.65	0.33	0.33

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

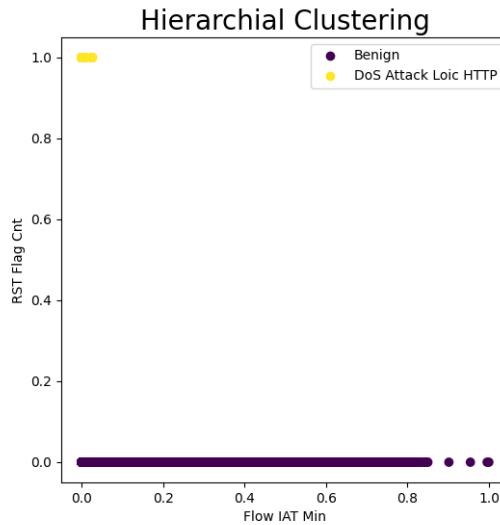


Figura 4.31: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
2	0.65	0.65	0.67

4.4. THUESDAY-20-02-2018_TRAFFICFORML_CICFLOWMETER

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

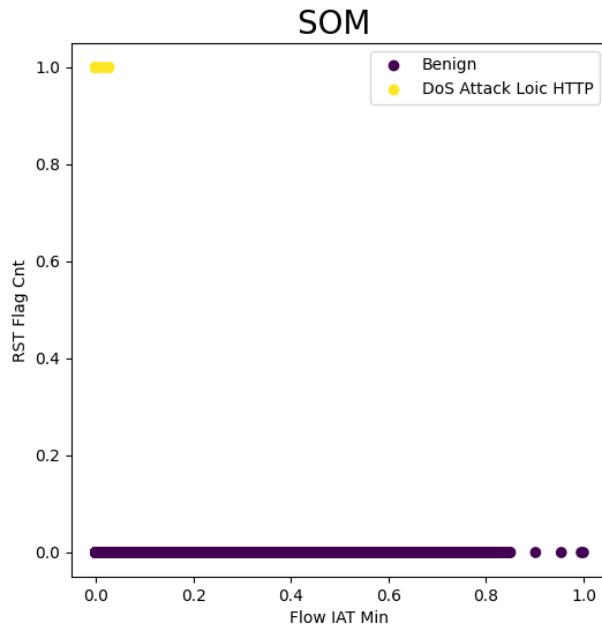


Figura 4.32: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
2	0.65	0.65	0.67

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

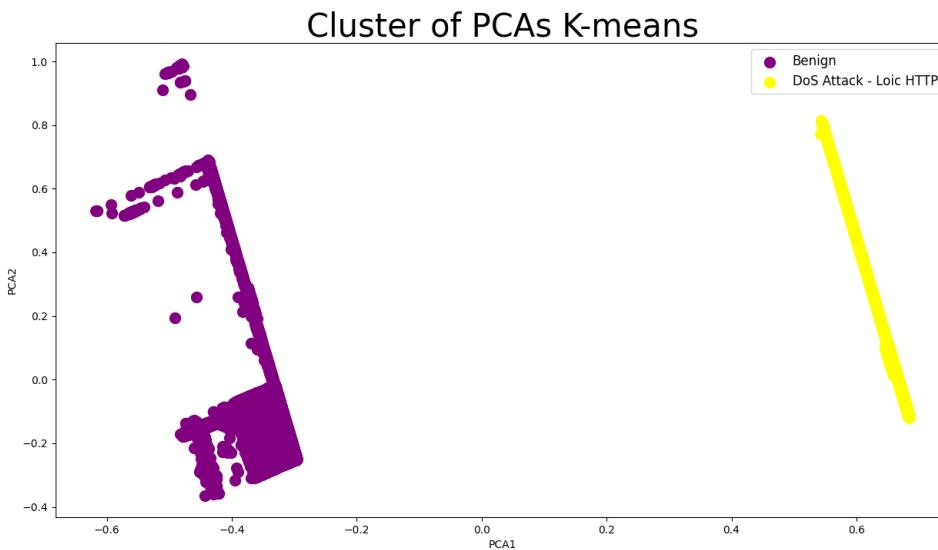


Figura 4.33: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.65	0.65	0.67

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

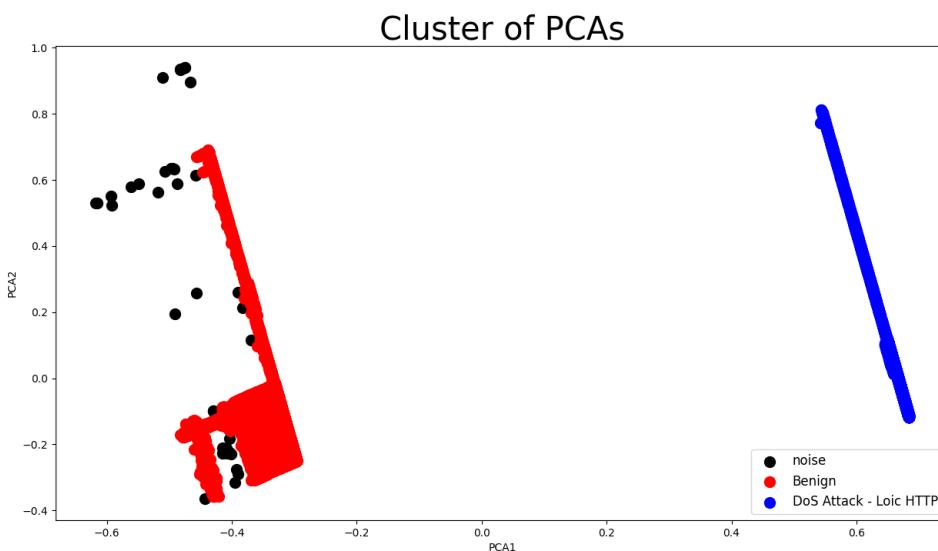


Figura 4.34: Risultati ottenuti con DBscan

4.4. THUESDAY-20-02-2018_TRAFFICFORML_CICFLOWMETER19

Num. cluster	Accuracy	Precision	Recall
2	0.62	0.14	0.14

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

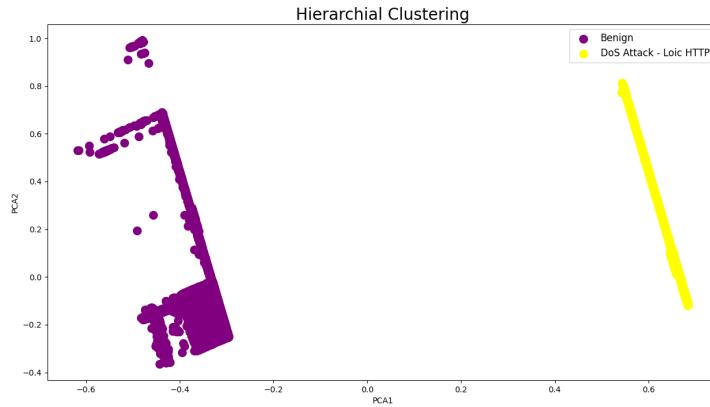


Figura 4.35: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
2	0.65	0.65	0.67

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

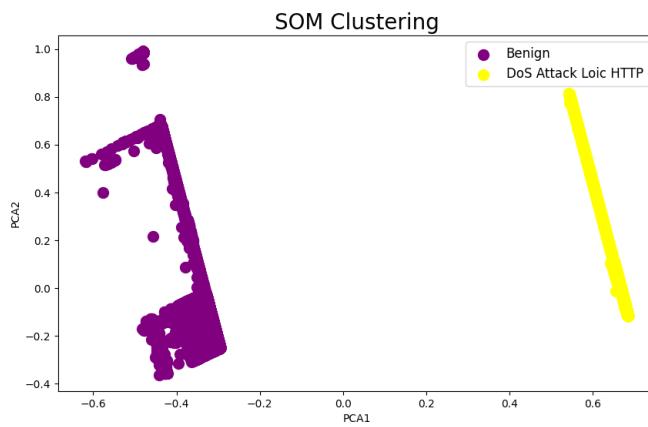


Figura 4.36: Risultati ottenuti con clustering SOM

Num. cluster	Accuracy	Precision	Recall
2	0.65	0.65	0.67

4.5 Thursday-01-03-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti due tipologie di cluster, **Benign** e l'attacco **Infiltration**. Prima di eseguire il clustering, il dataset relativo a tale file è stato ridotto del 40% oltre che effettuare il bilanciamento del dataset.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Fwd Seg Size Min** e **Flow Byts/s**.

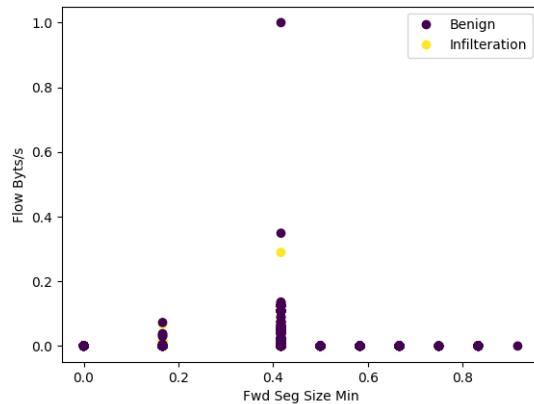


Figura 4.37: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

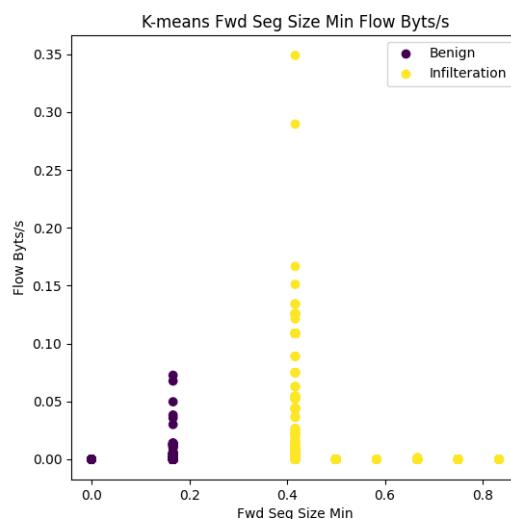


Figura 4.38: Risultati ottenuti con k-means

4.5. THURSDAY-01-03-2018_TRAFFICFORML_CICFLOWMETE~~RS~~1

Num. cluster	Accuracy	Precision	Recall
2	0.54	0.54	0.54

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

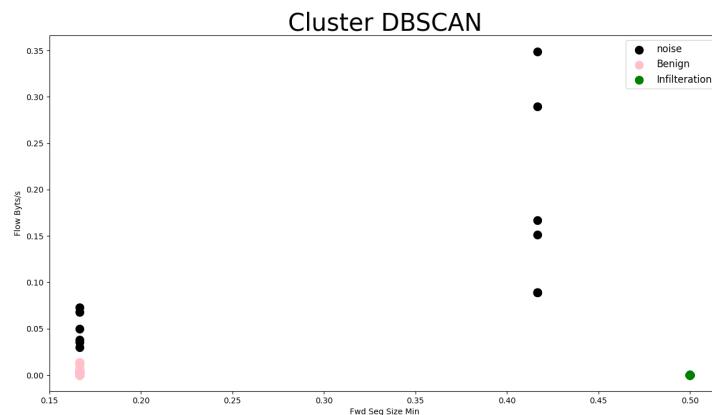


Figura 4.39: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
2	0.30	0.5	0.13

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

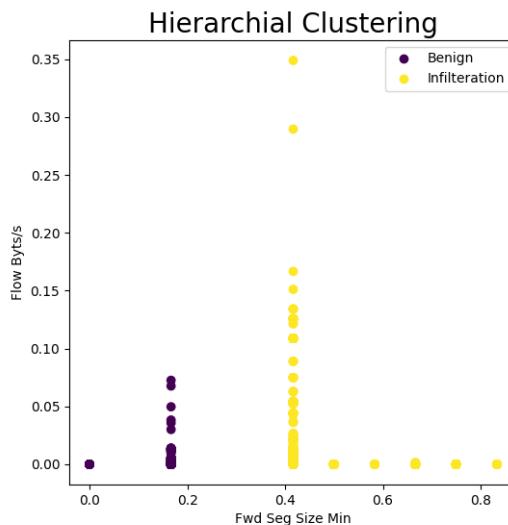


Figura 4.40: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
2	0.54	0.54	0.54

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

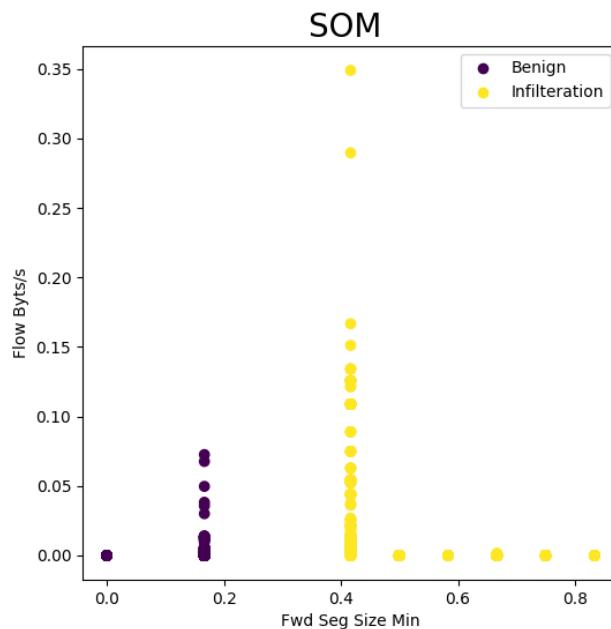


Figura 4.41: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
2	0.54	0.54	0.54

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

4.5. THURSDAY-01-03-2018_TRAFFICFORML_CICFLOWMETE~~ES~~3

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

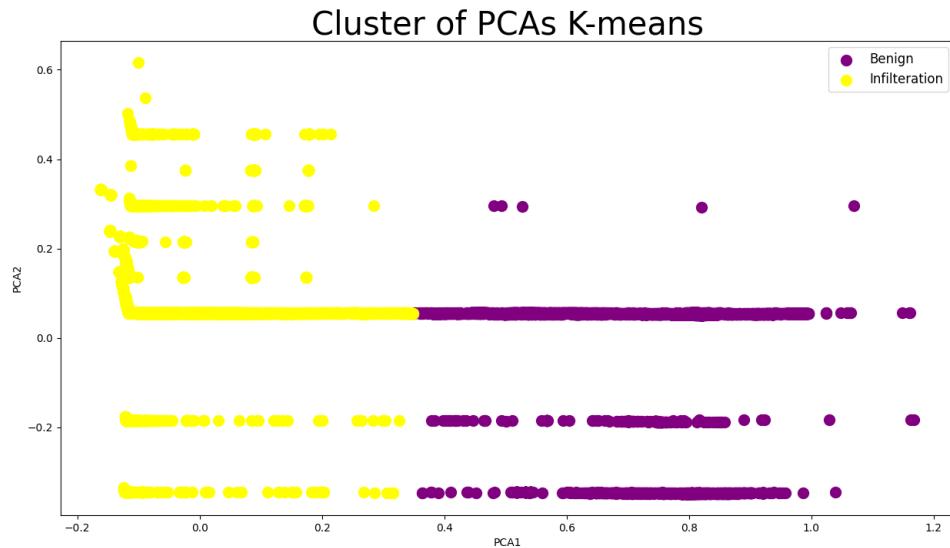


Figura 4.42: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.52	0.52	0.56

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

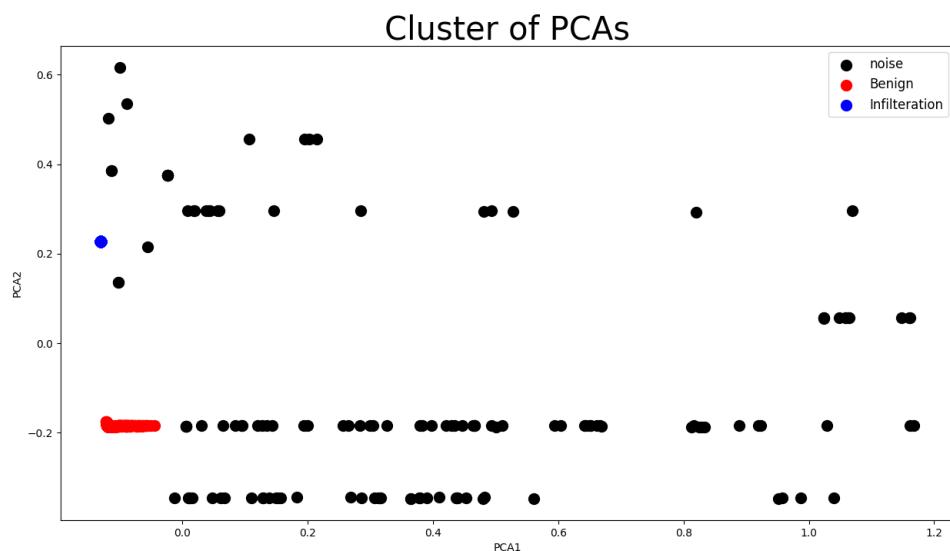


Figura 4.43: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
2	0.20	0.01	0.03

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:



Figura 4.44: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
2	0.52	0.52	0.55

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

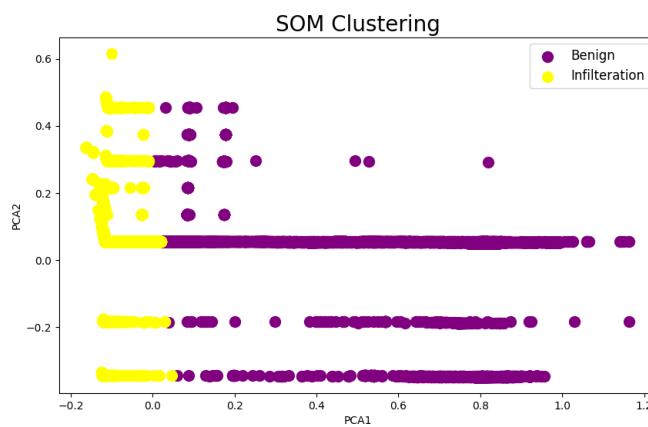


Figura 4.45: Risultati ottenuti con clustering SOM

Num. cluster	Accuracy	Precision	Recall
2	0.52	0.52	0.54

4.6. THURSDAY-15-02-2018_TRAFFICFORML_CICFLOWMETE~~RS~~5

4.6 Thursday-15-02-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti tre tipologie di cluster, **Benign**, l'attacco **DoS Attack - GoldenEye** e l'attacco **DoS Attack - Slowloris**. Oltre al clustering effettuato considerando il flusso benigno ed i singoli flussi relativi ad attacchi, si è optato anche per un clustering binario (con distinzione tra flusso benigno e maligno), al fine di analizzare eventuali differenze tra le due analisi.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Fwd Seg Size Min** e **Bwd IAT Mean**.

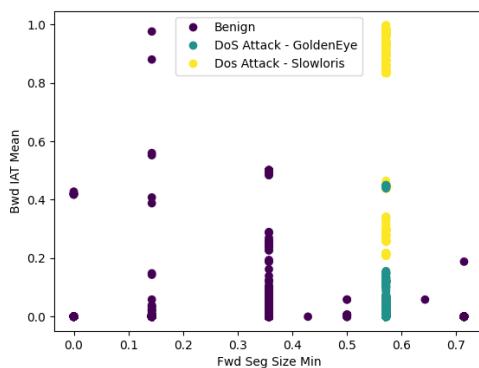


Figura 4.46: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

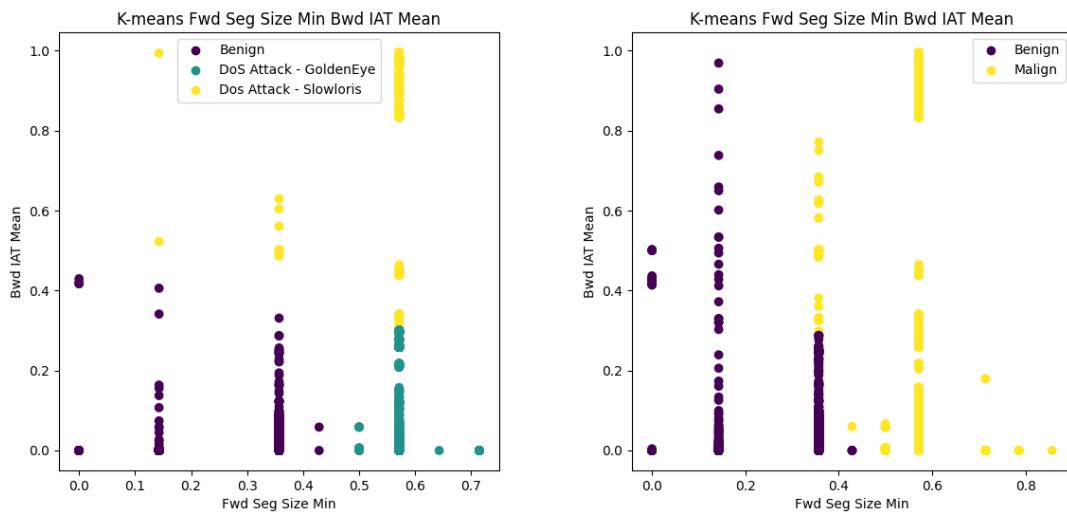


Figura 4.47: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
3	0.87	0.87	0.90
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

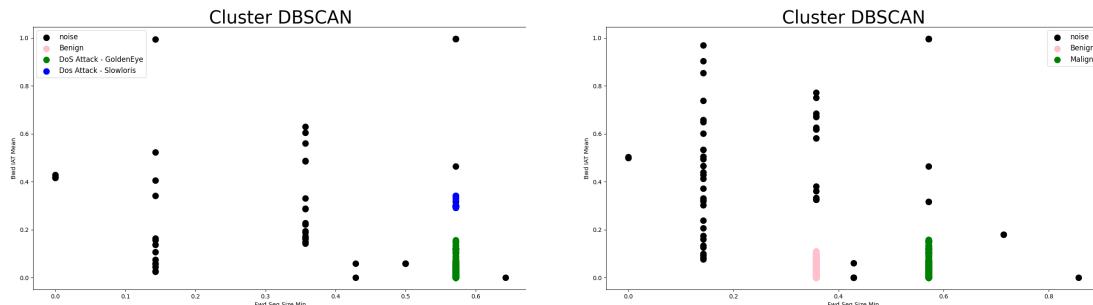


Figura 4.48: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
3	0.33	0.05	0.09
2	0.71	0.06	0.09

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

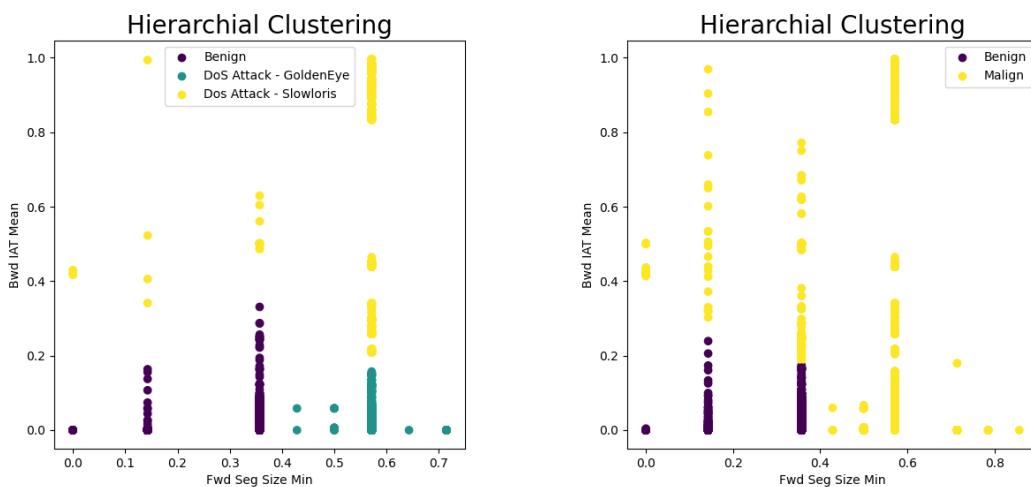


Figura 4.49: Risultati ottenuti con clustering gerarchico agglomerativo

4.6. THURSDAY-15-02-2018_TRAFFICFORML_CICFLOWMETE~~87~~

Num. cluster	Accuracy	Precision	Recall
3	0.88	0.88	0.90
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

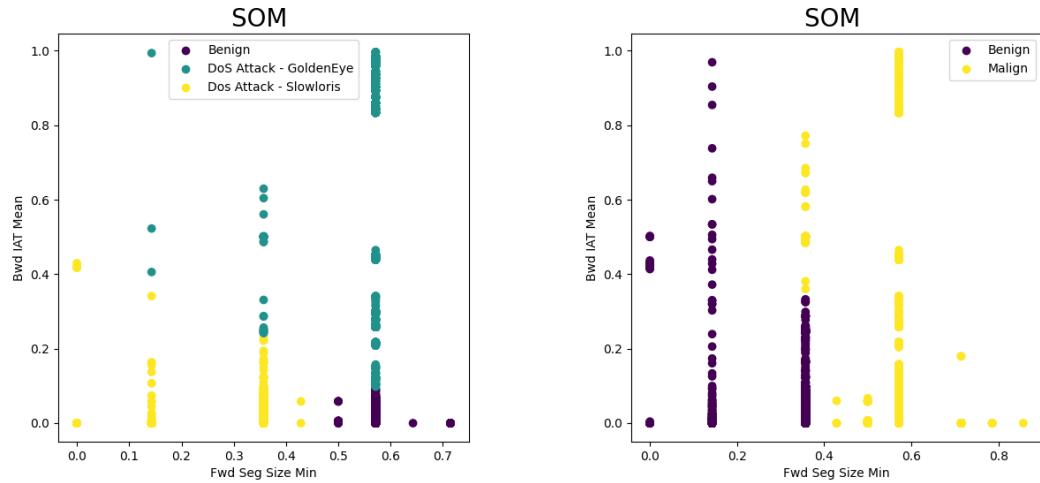


Figura 4.50: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
3	0.62	0.46	0.47
2	0.99	0.99	0.99

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

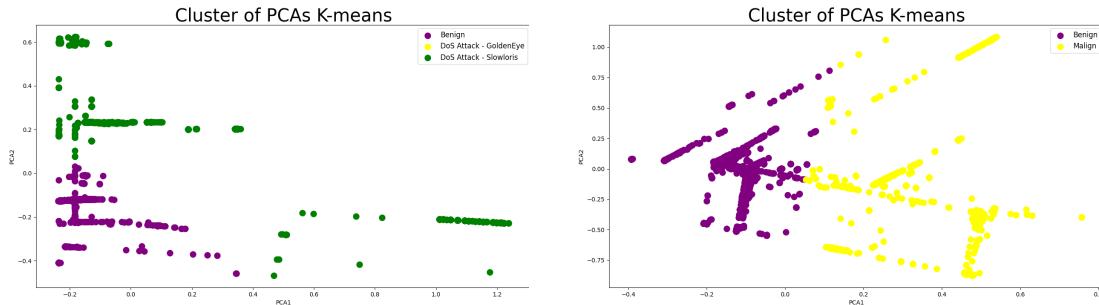


Figura 4.51: Risultati ottenuti con Kmeans

Num. cluster	Accuracy	Precision	Recall
3	0.61	0.61	0.41
2	0.82	0.82	0.85

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

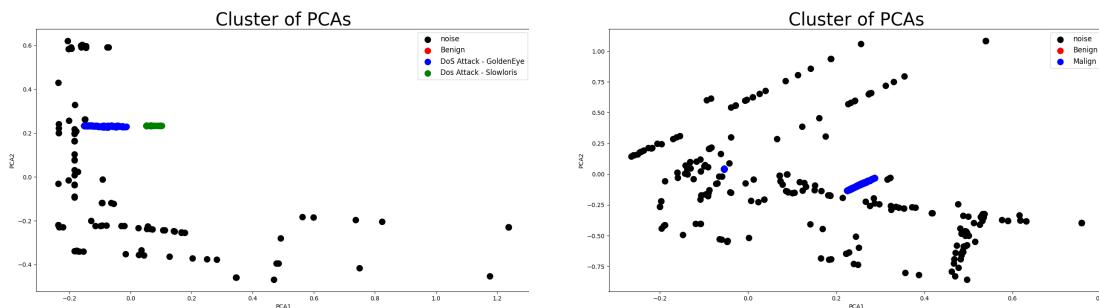


Figura 4.52: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
3	0.22	0.02	0.05
2	0.50	0.02	0.03

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

4.6. THURSDAY-15-02-2018_TRAFFICFORML_CICFLOWMETE~~E9~~

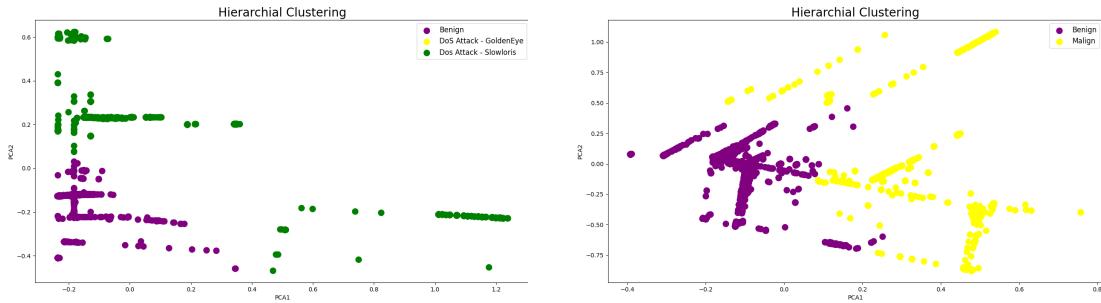


Figura 4.53: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
3	0.61	0.61	0.41
2	0.78	0.78	0.79

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

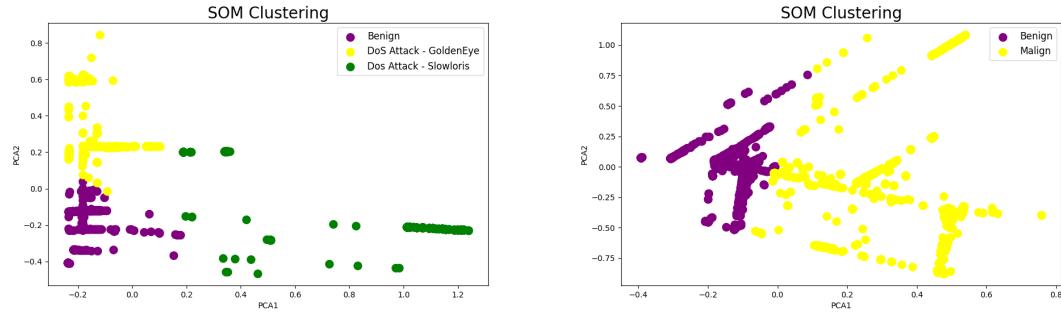


Figura 4.54: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
3	0.75	0.75	0.79
2	0.82	0.82	0.85

4.7 Thursday-22-02-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti quattro tipologie di cluster, **Benign**, l'attacco **Brute Force - Web**, l'attacco **Brute Force - XSS** e l'attacco **SQL Injection**. Oltre al clustering effettuato considerando il flusso benigno ed i singoli flussi relativi ad attacchi, si è optato anche per un clustering binario (con distinzione tra flusso benigno e maligno), al fine di analizzare eventuali differenze tra le due analisi.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Fwd Pkt Len Max** e **Init Fwd Win Byts**.

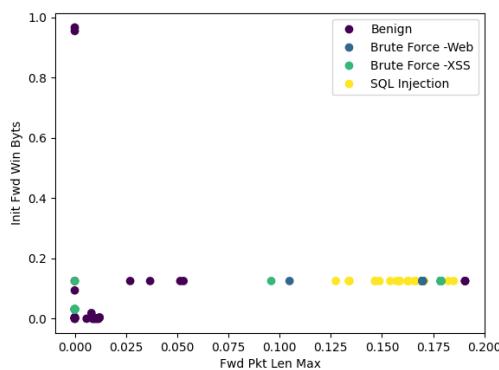


Figura 4.55: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

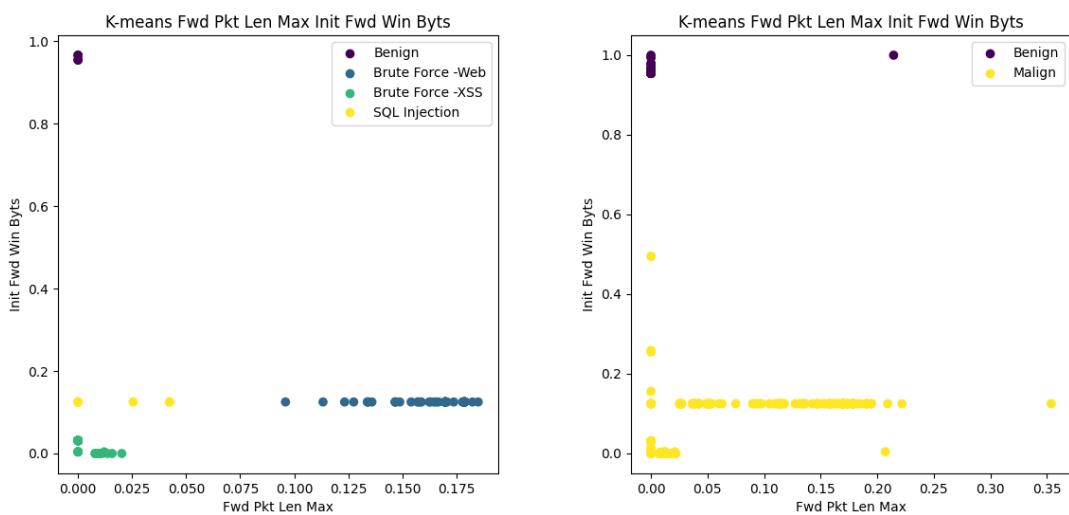


Figura 4.56: Risultati ottenuti con k-means

4.7. THURSDAY-22-02-2018_TRAFFICFORML_CICFLOWMETE~~61~~

Num. cluster	Accuracy	Precision	Recall
4	0.33	0.33	0.44
2	0.54	0.54	0.76

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

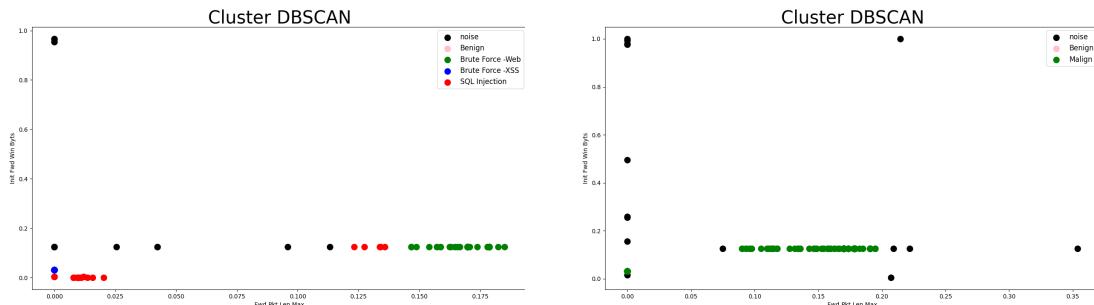


Figura 4.57: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
4	0.32	0.25	0.19
2	0.49	0.12	0.10

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

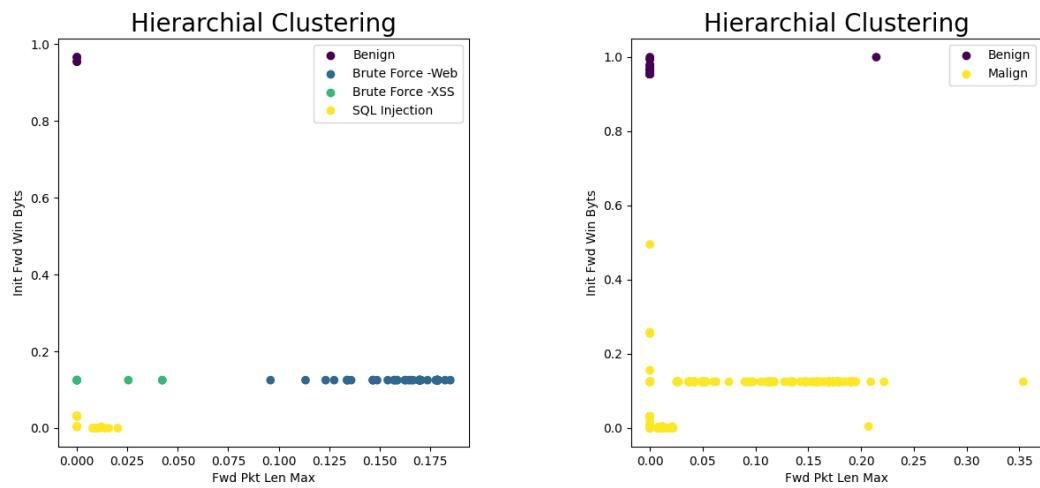


Figura 4.58: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
4	0.28	0.28	0.42
2	0.54	0.54	0.76

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

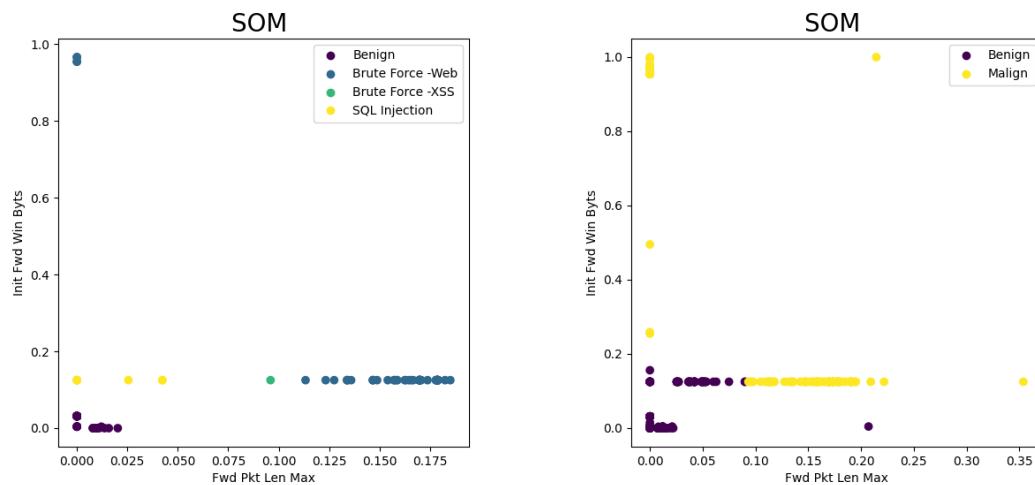


Figura 4.59: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
4	0.29	0.29	0.43
2	0.62	0.62	0.63

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

4.7. THURSDAY-22-02-2018_TRAFFICFORML_CICFLOWMETE~~13~~3

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

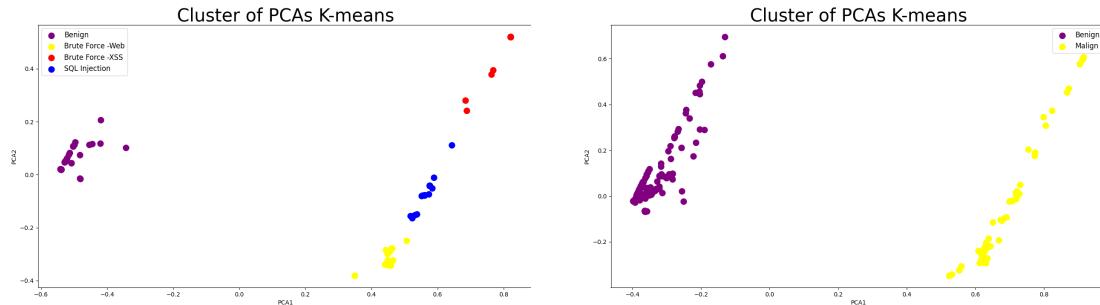


Figura 4.60: Risultati ottenuti con Kmeans

Num. cluster	Accuracy	Precision	Recall
4	0.49	0.49	0.55
2	0.72	0.72	0.75

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

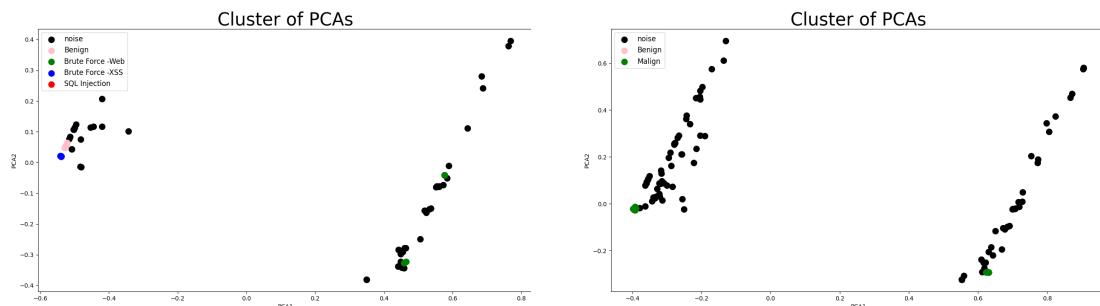


Figura 4.61: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
4	0.34	0.27	0.28
2	0.26	0.04	0.05

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

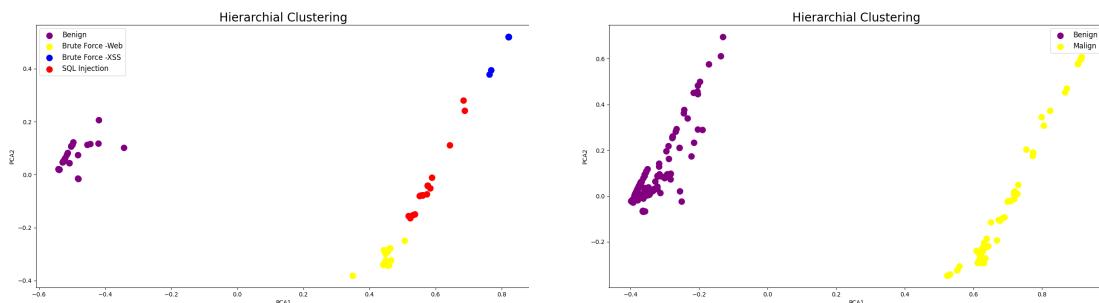


Figura 4.62: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
4	0.5	0.5	0.56
2	0.72	0.72	0.75

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

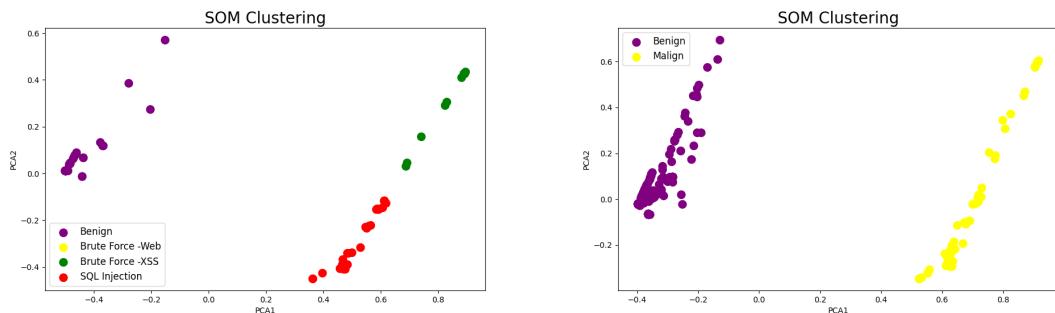


Figura 4.63: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
4	0.48	0.48	0.47
2	0.72	0.72	0.75

4.8.

WEDNESDAY-14-02-2018_TRAFFICFORML_CICFLOWMETER 65

4.8 Wednesday-14-02-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti tre tipologie di cluster, **Benign**, l'attacco **FTP - BruteForce** e l'attacco **SSH - BruteForce**. Oltre al clustering effettuato considerando il flusso benigno ed i singoli flussi relativi ad attacchi, si è optato anche per un clustering binario (con distinzione tra flusso benigno e maligno), al fine di analizzare eventuali differenze tra le due analisi. Prima di eseguire il clustering, il dataset relativo a tale file è stato ridotto dell'80% oltre che effettuare il bilanciamento del dataset.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Fwd Seg Size Min** e **Bwd Pkts/s**.

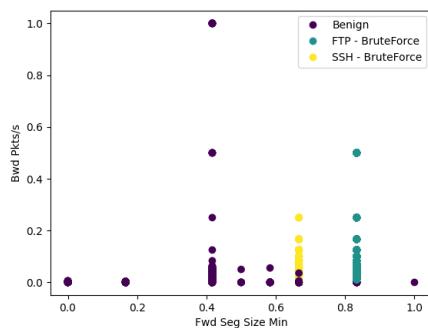


Figura 4.64: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

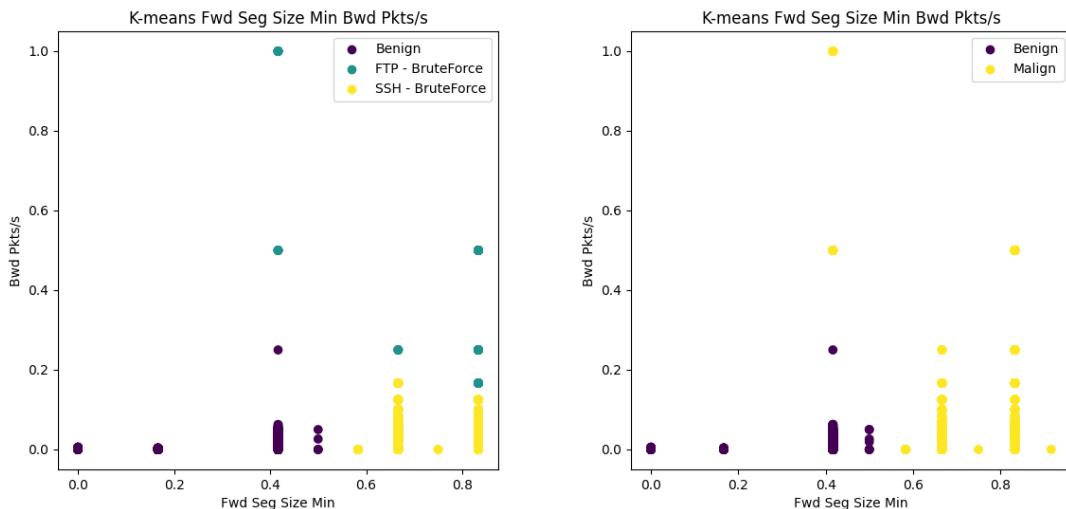


Figura 4.65: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
3	0.95	0.95	0.95
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

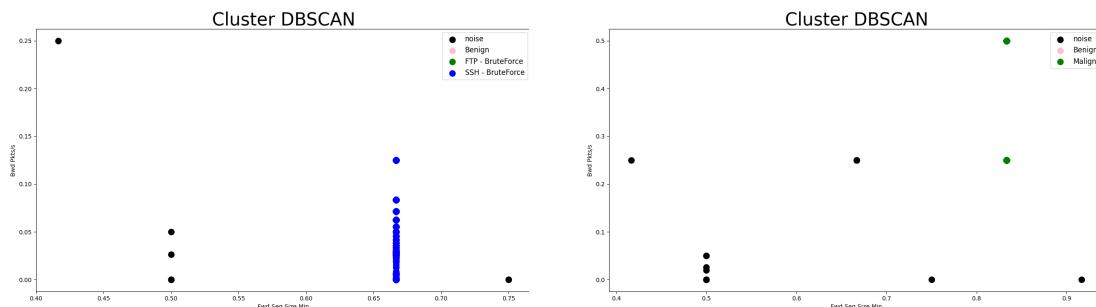


Figura 4.66: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
3	0.31	0.04	0.05
2	0.2	0.02	0.05

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

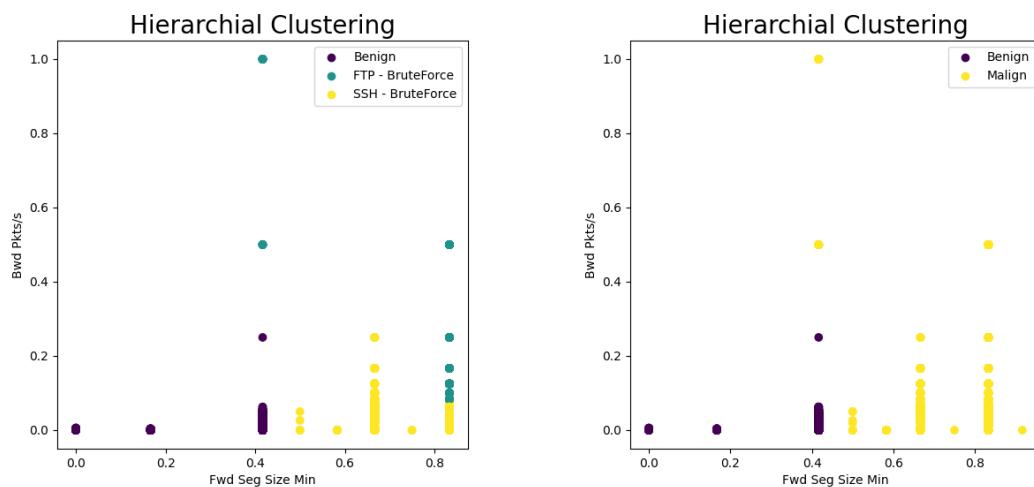


Figura 4.67: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
3	0.96	0.96	0.96
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

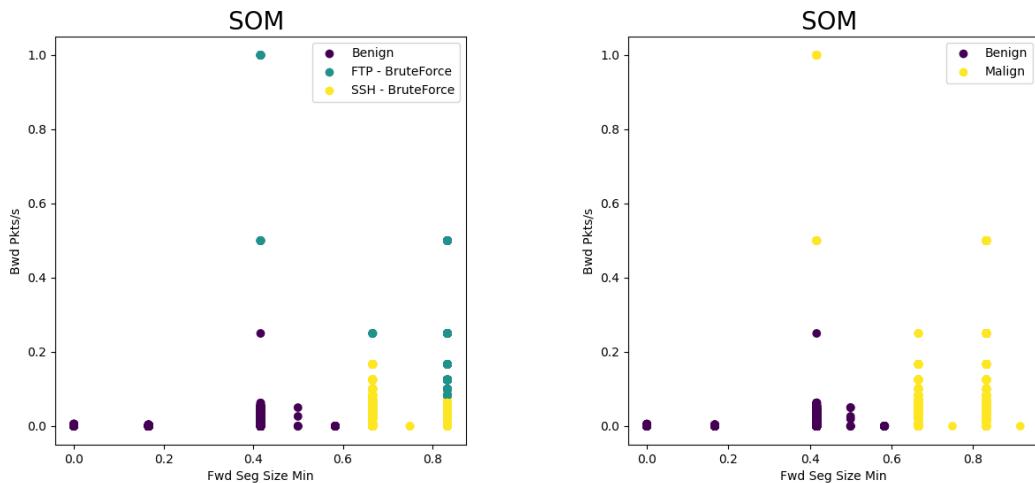


Figura 4.68: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
3	0.96	0.96	0.96
2	0.99	0.99	0.99

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

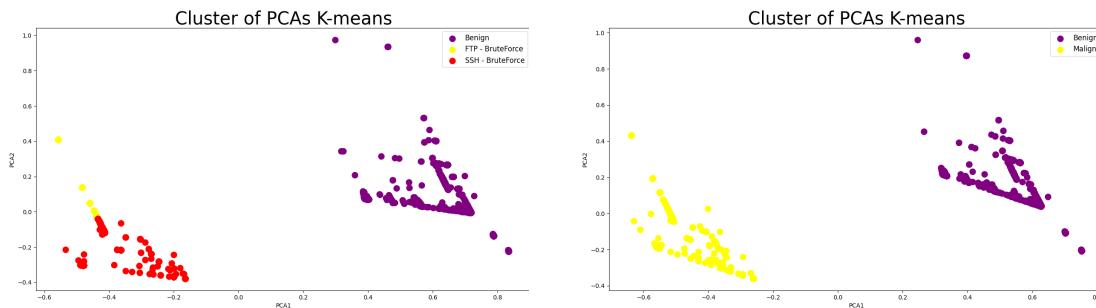


Figura 4.69: Risultati ottenuti con Kmeans

Num. cluster	Accuracy	Precision	Recall
3	0.66	0.66	0.68
2	0.68	0.68	0.68

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

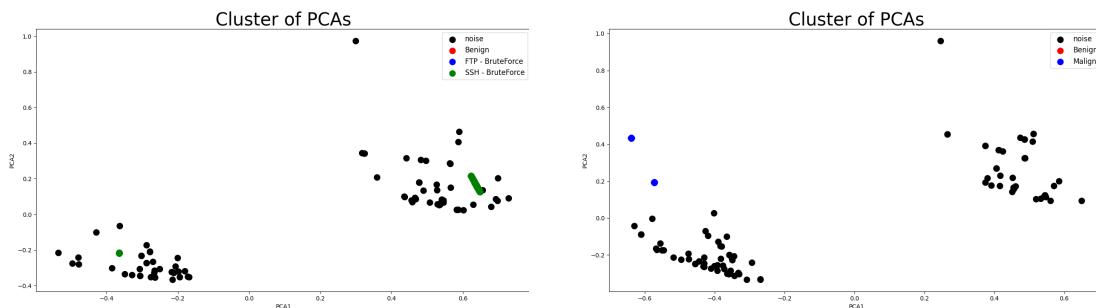


Figura 4.70: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
3	0.31	0.02	0.02
2	0.02	0.01	0.02

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

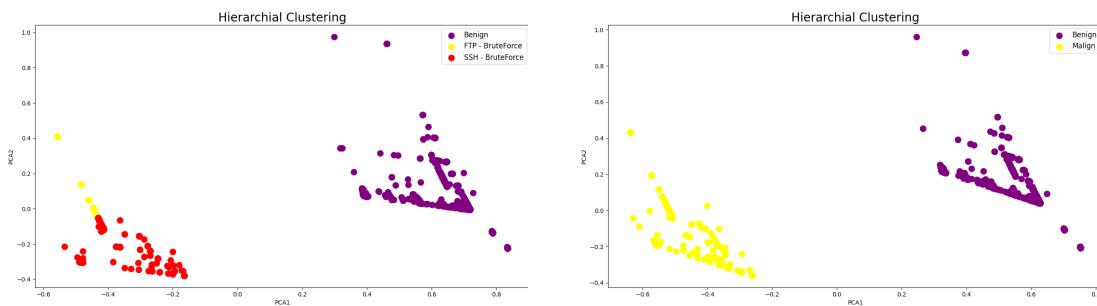


Figura 4.71: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
3	0.66	0.66	0.68
2	0.68	0.68	0.68

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

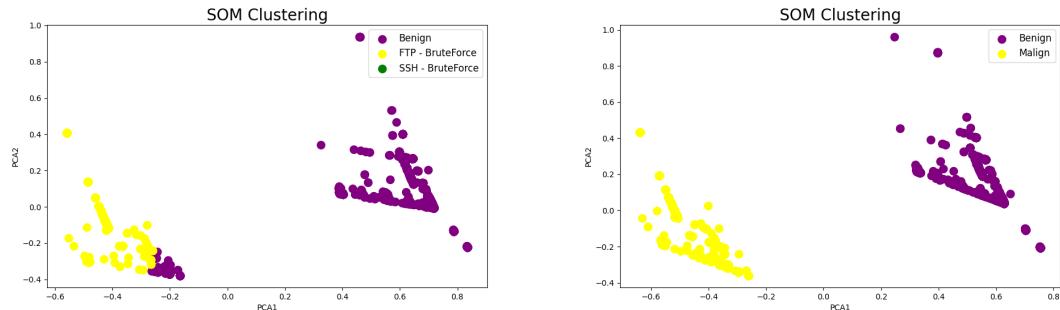


Figura 4.72: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
3	0.66	0.66	0.44
2	0.68	0.68	0.68

4.9 Wednesday-21-02-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti tre tipologie di cluster, **Benign**, l'attacco **DDoS attack - LOIC - UDP** e l'attacco **DDoS attack - HOIC**. Oltre al clustering effettuato considerando il flusso benigno ed i singoli flussi relativi ad attacchi, si è optato anche per un clustering binario (con distinzione tra flusso benigno e maligno), al fine di analizzare eventuali differenze tra le due analisi.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Init Bwd Win Byts** e **Subflow Fwd Pkts**.

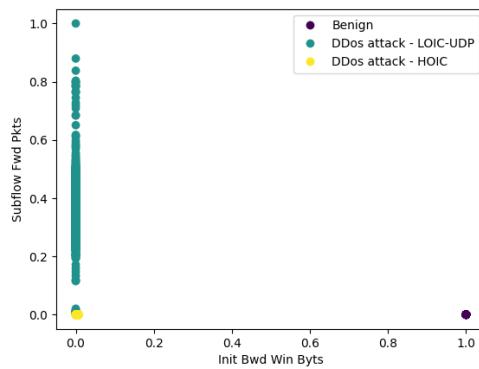


Figura 4.73: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

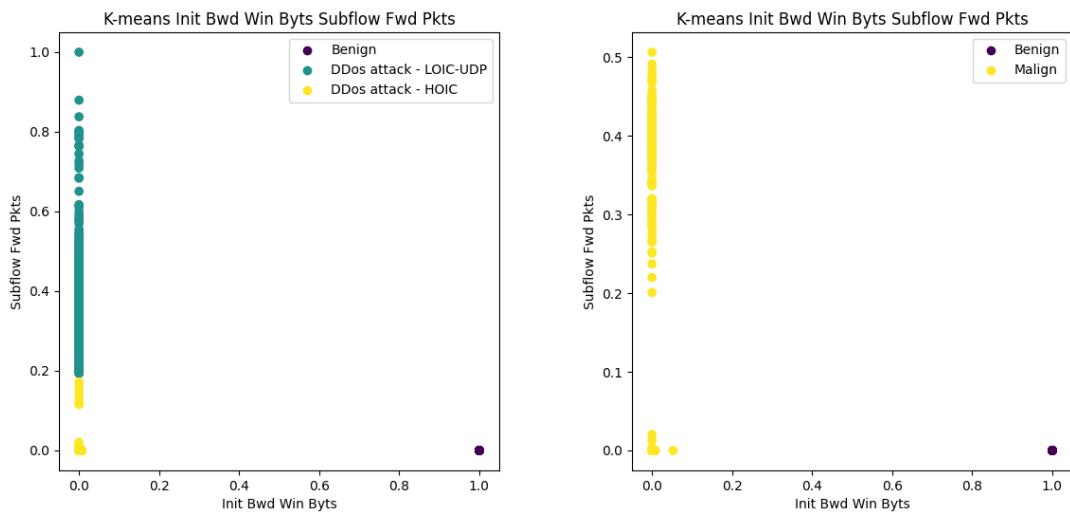


Figura 4.74: Risultati ottenuti con k-means

4.9.

WEDNESDAY-21-02-2018_TRAFFICFORML_CICFLOWMETER 71

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

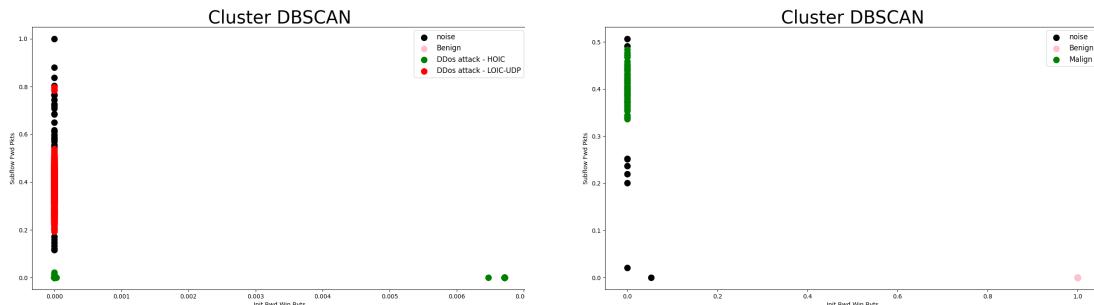


Figura 4.75: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
3	0.65	0.39	0.40
2	0.99	0.33	0.33

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

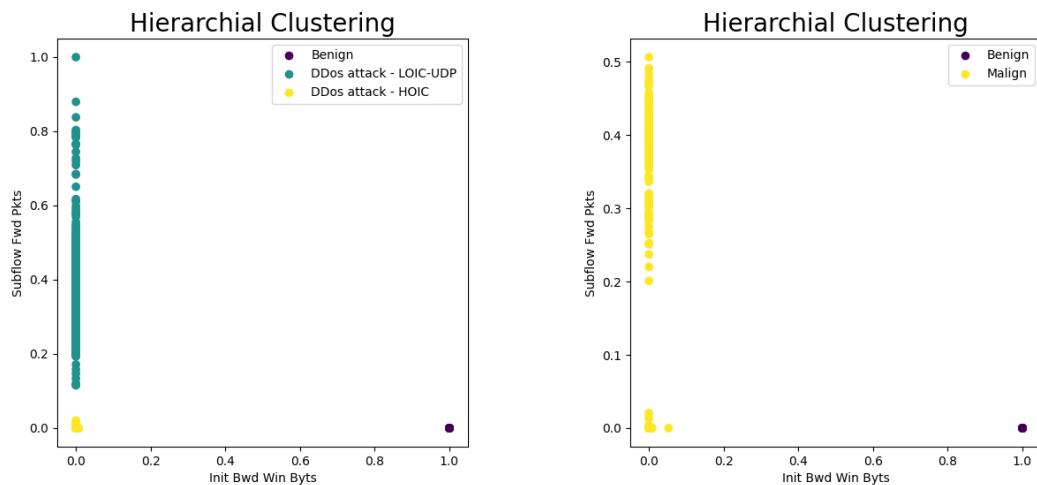


Figura 4.76: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.99	0.99	0.99

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

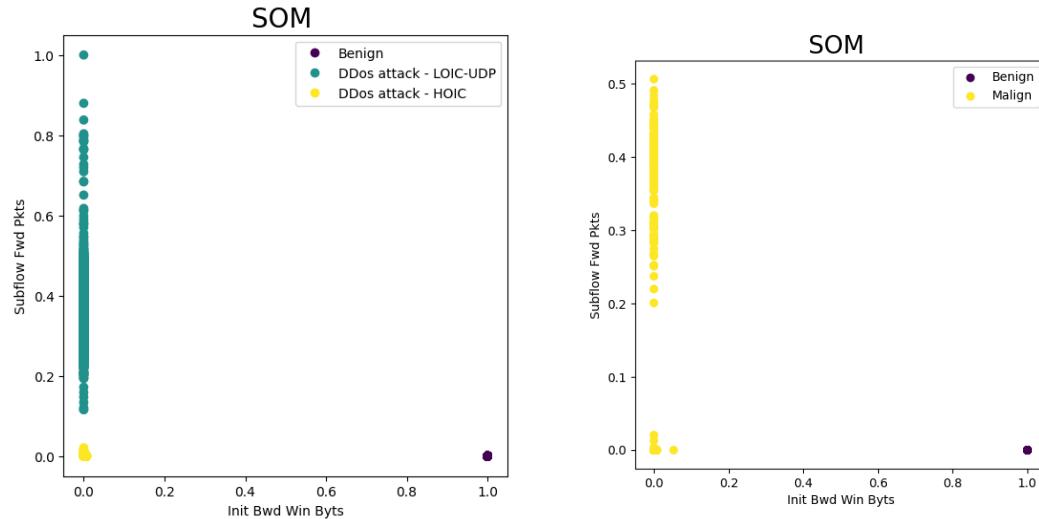


Figura 4.77: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.99	0.99	0.99

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

4.9.

WEDNESDAY-21-02-2018_TRAFFICFORML_CICFLOWMETER 73

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

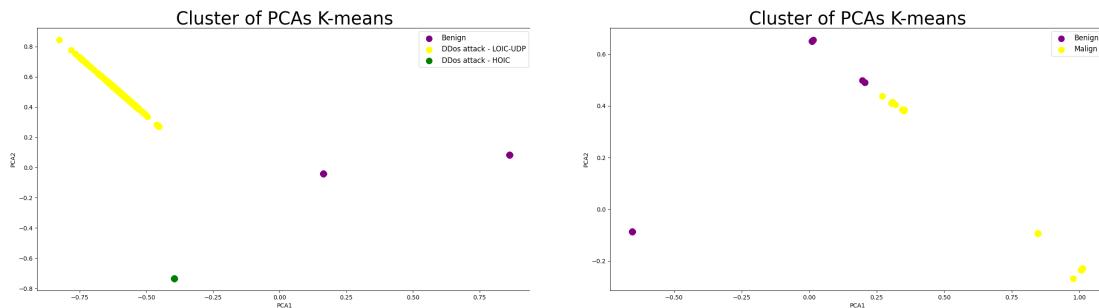


Figura 4.78: Risultati ottenuti con Kmeans

Num. cluster	Accuracy	Precision	Recall
3	0.91	0.91	0.93
2	0.88	0.88	0.9

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

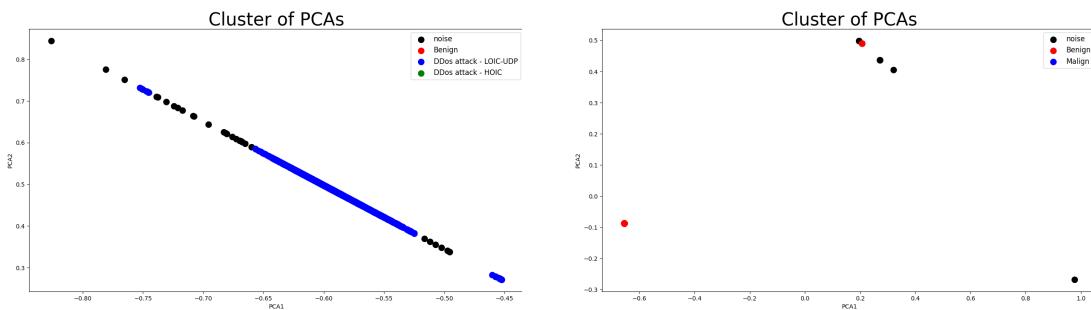


Figura 4.79: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
3	0.32	0.14	0.14
2	0.49	0.12	0.12

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

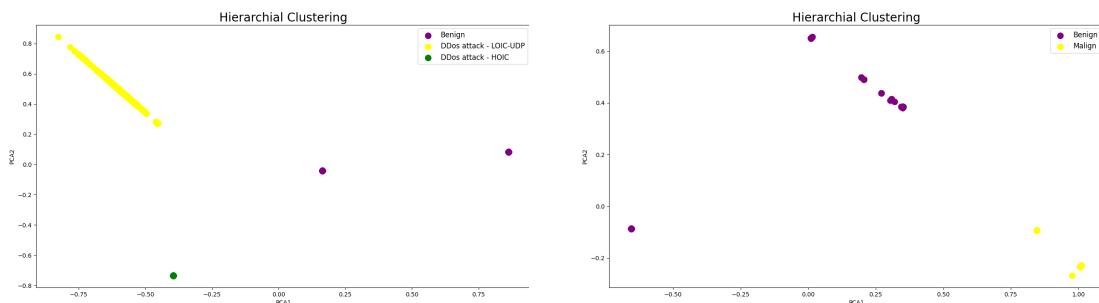


Figura 4.80: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
3	0.91	0.91	0.93
2	0.88	0.88	0.9

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

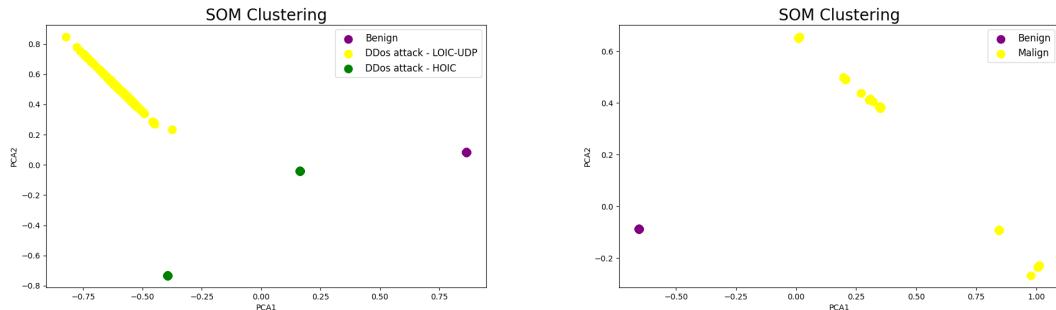


Figura 4.81: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
3	0.99	0.99	0.99
2	0.88	0.88	0.9

4.10.

WEDNESDAY-28-02-2018_TRAFFICFORML_CICFLOWMETER 75

4.10 Wednesday-28-02-2018_TrafficForML_CICFlowMeter

All'interno di questo file sono presenti tre tipologie di cluster, **Benign** e l'attacco **Infiltration**.

Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Flow Pkts.s** e **Flow IAT Mean**.

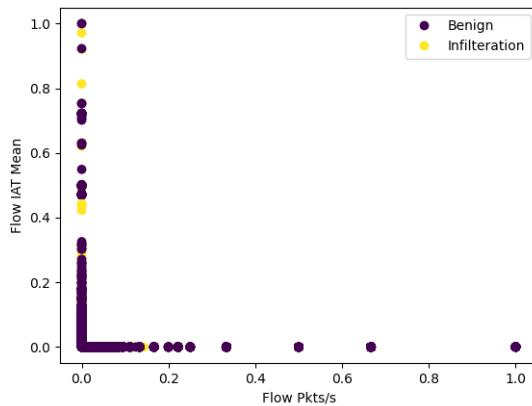


Figura 4.82: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

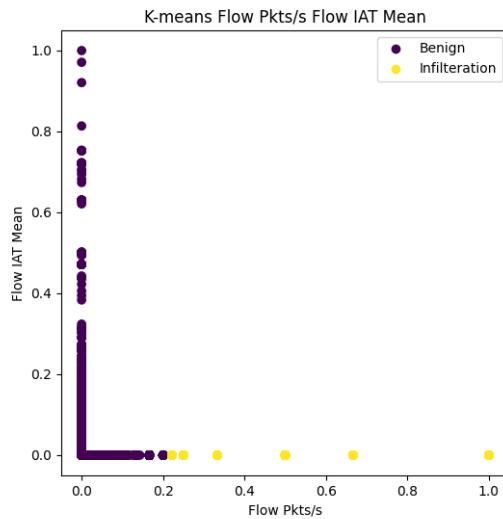


Figura 4.83: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.51	0.51	0.56

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

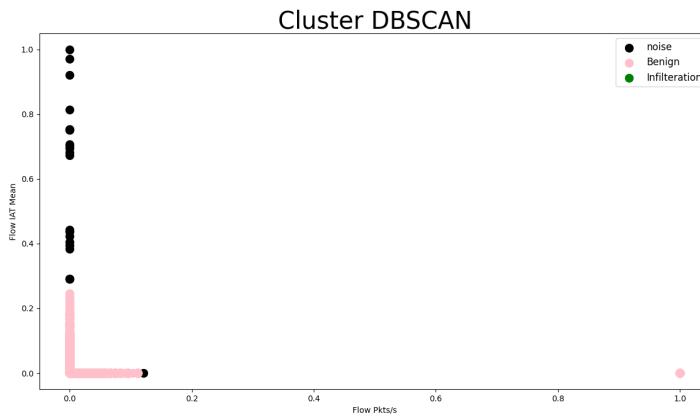


Figura 4.84: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
2	0.47	0.05	0.03

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

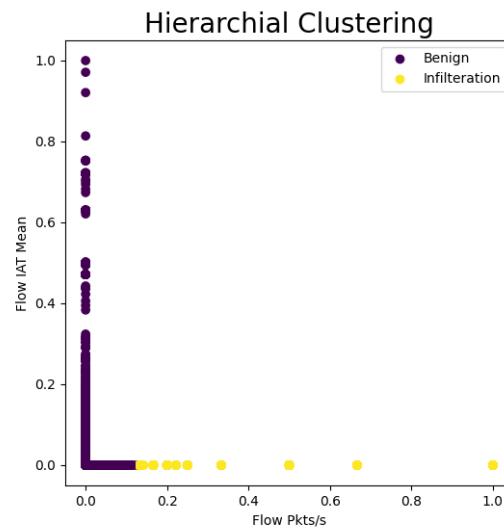


Figura 4.85: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
2	0.51	0.51	0.56

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

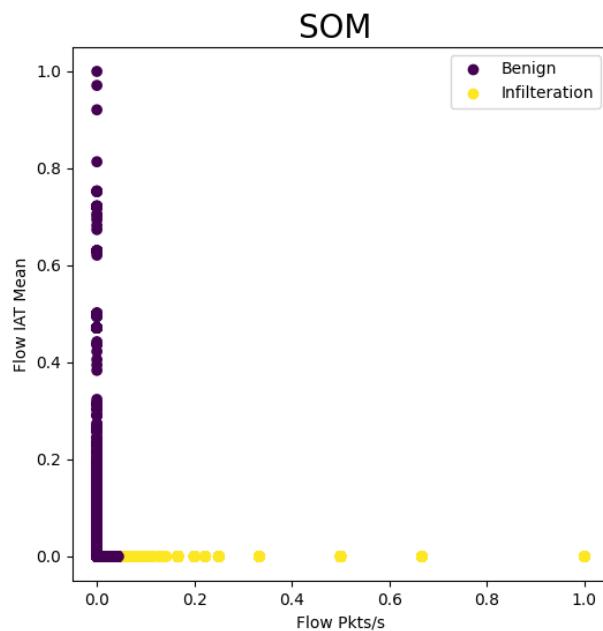


Figura 4.86: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
2	0.51	0.51	0.55

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

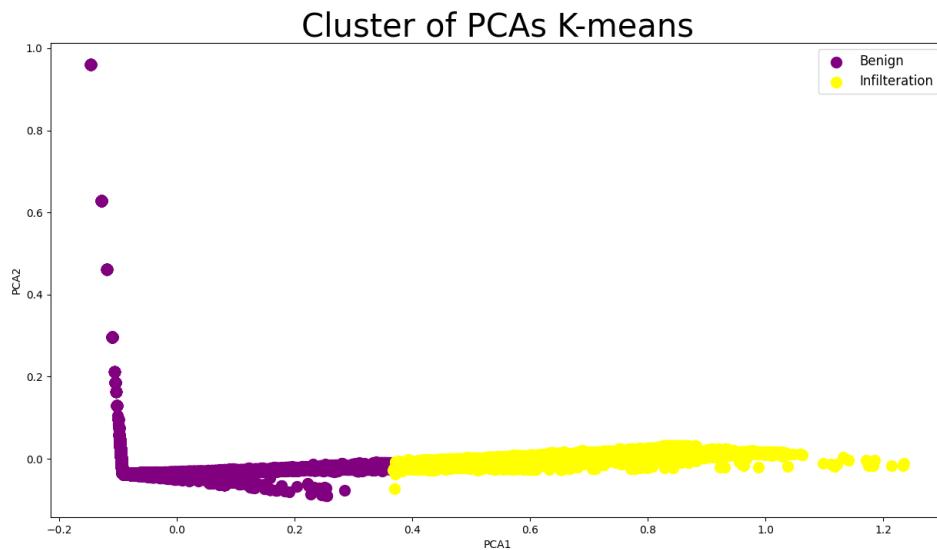


Figura 4.87: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.50	0.50	0.50

E' stato applicato l'algoritmo **DBscan**, ottenendo i seguenti risultati:

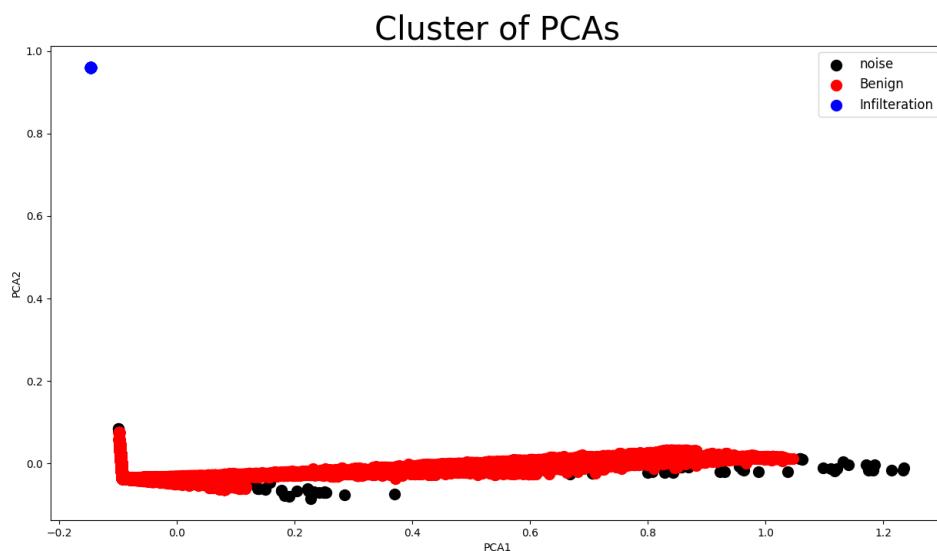


Figura 4.88: Risultati ottenuti con DBscan

Num. cluster	Accuracy	Precision	Recall
2	0.43	0.05	0.03

E' stato applicato l'algoritmo di clustering **gerarchico agglomerativo**, ottenendo i seguenti risultati:

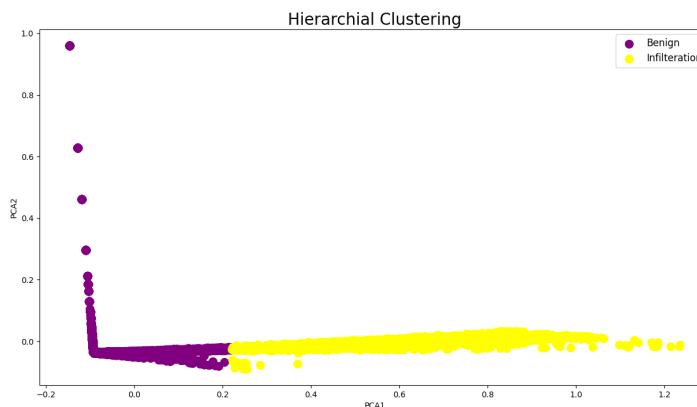


Figura 4.89: Risultati ottenuti con clustering gerarchico agglomerativo

Num. cluster	Accuracy	Precision	Recall
2	0.50	0.50	0.50

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

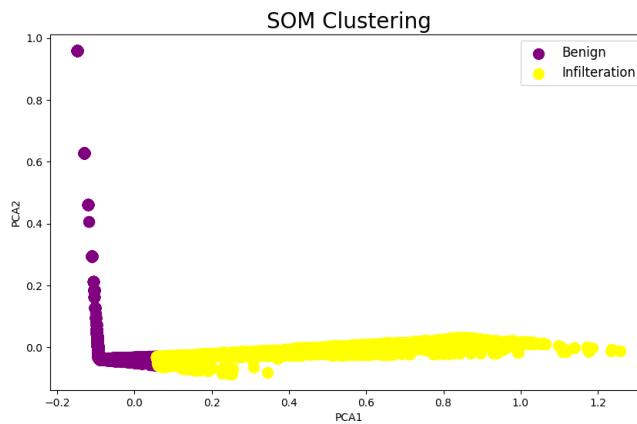


Figura 4.90: Risultati ottenuti con clustering SOM

Num. cluster	Accuracy	Precision	Recall
2	0.50	0.50	0.50

4.11 Clustering sull'intero dataset

All'interno di questo file sono presenti due tipologie di cluster, **Benign** e **Malign**. Riportiamo di seguito lo scatter plot che mostra come sono distribuiti gli elementi nello spazio delle feature.

Le due feature utilizzate sono **Init Fwd Win Byts** e **Init Bwd Win Byts**.

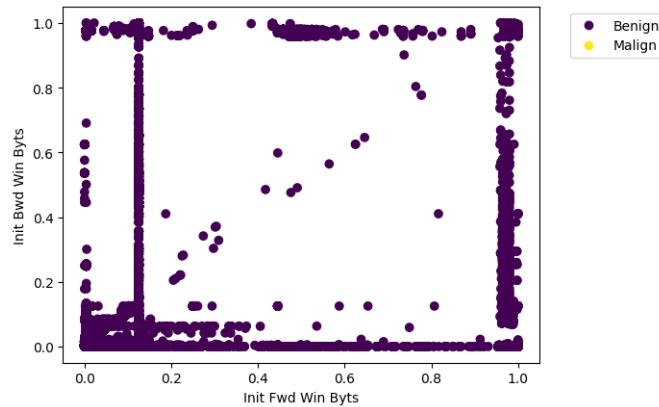


Figura 4.91: Scatter plot del dataset

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

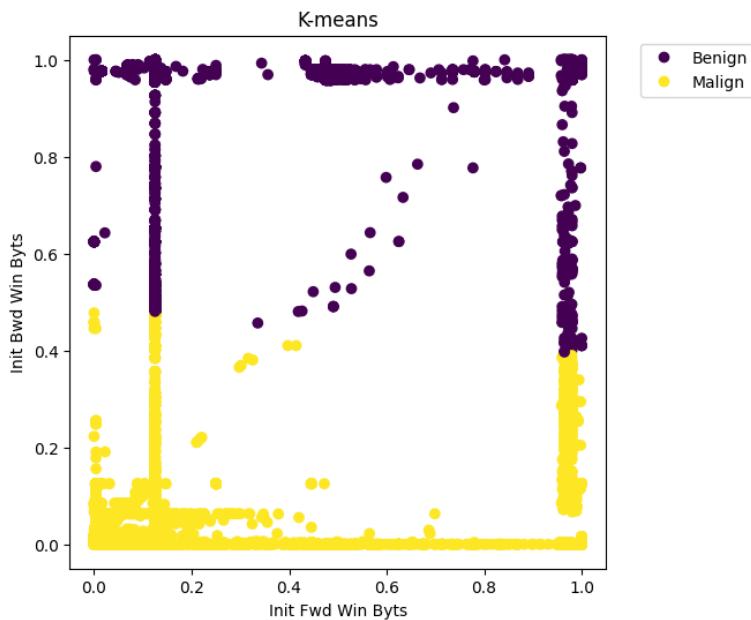


Figura 4.92: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.58	0.58	0.77

E' stato applicato l'algoritmo **SOM**, ottenendo i seguenti risultati:

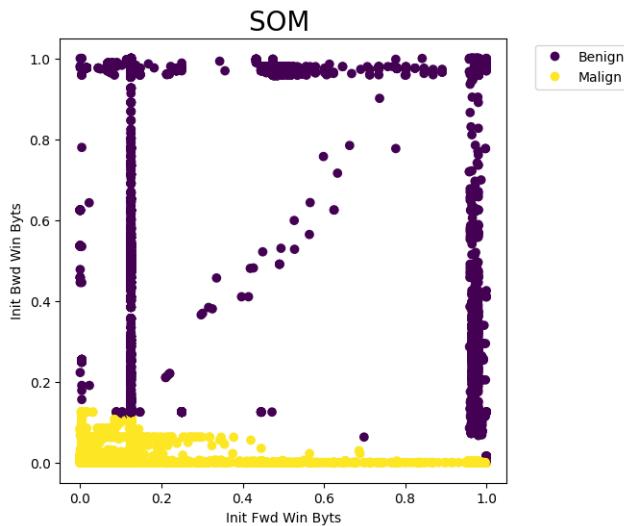


Figura 4.93: Risultati ottenuti con SOM

Num. cluster	Accuracy	Precision	Recall
2	0.61	0.61	0.78

Si è proceduto applicando l'algoritmo **PCA** al dataset, con le cinque feature più importanti. Il nuovo dataset contiene solo due feature e ad esso applichiamo gli algoritmi di clustering.

E' stato applicato l'algoritmo **K-means**, ottenendo i seguenti risultati:

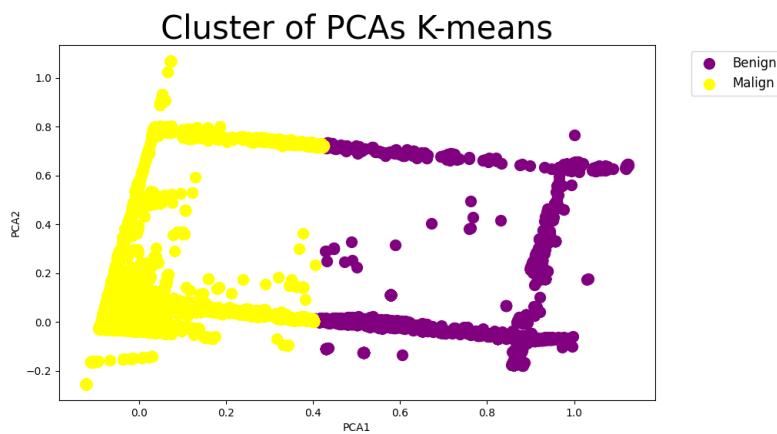


Figura 4.94: Risultati ottenuti con k-means

Num. cluster	Accuracy	Precision	Recall
2	0.58	0.58	0.77

E' stato applicato l'algoritmo di clustering **SOM**, ottenendo i seguenti risultati:

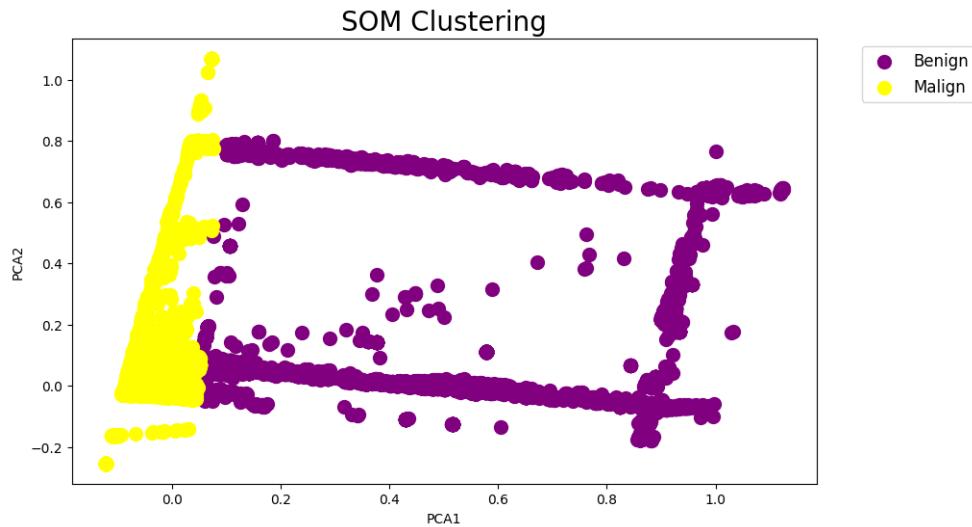


Figura 4.95: Risultati ottenuti con clustering SOM

Num. cluster	Accuracy	Precision	Recall
2	0.61	0.61	0.78

Capitolo 5

Conclusioni e Sviluppi futuri

Il nostro progetto è strutturato in diversi file.

Inizialmente abbiamo analizzato singolarmente ciascun file, eseguendo un preprocessing del dataset per rimuovere eventuali valori infiniti o nulli e successivamente normalizzando i dati.

Ogni file è stato quindi salvato in formato pickle per poter essere utilizzato nella fase di clustering.

Contemporaneamente abbiamo identificato le feature più rilevanti utilizzando diversi algoritmi di selezione e combinando i risultati ottenuti.

Abbiamo, inoltre, creato degli scatterplot per individuare le correlazioni tra le feature selezionate.

Per il clustering abbiamo preso in considerazione le prime due feature individuate come le più importanti.

Su queste feature abbiamo applicato diversi algoritmi per ottenere diversi cluster, distinguendo tra i cluster relativi ai tipi di attacco e il cluster di dati benigni, per ciascun file.

Successivamente abbiamo calcolato le metriche di valutazione dei cluster ottenuti e le abbiamo confrontate con le metriche ottenute applicando la tecnica di riduzione delle dimensioni PCA agli stessi algoritmi.

Abbiamo seguito lo stesso approccio per l'intero dataset, con l'unica differenza che durante la fase di clustering abbiamo distinti i flussi tra dati benigni e maligni.

5.1 Sviluppi futuri

I risultati mostrano come determinate feature siano altamente discriminanti.

Un possibile sviluppo futuro potrebbe essere quello di valutare se tali caratteristiche conservano la loro capacità discriminante anche in un dataset contenente attacchi reali.

Nel caso in cui questa valutazione rivelasse un esito negativo, sarebbe importante comprendere se ci sono state modifiche negli attacchi stessi o se il dataset artificialmente generato presenta qualche tipo di bias.

I modelli ottenuti potrebbero essere sottoposti a test all'interno di un *intrusion detection system*, un *intrusion prevention system*, oppure all'interno di *SIEM*.

Si è osservato come alcune feature risultino essere le più discriminanti per più di un file.

Una ulteriore analisi potrebbe consistere nell'unificare questi dataset e applicare vari algoritmi di feature selection per osservare se le stesse caratteristiche, identificate in precedenza, rimangono ancora tra le più importanti.

Inoltre, potrebbe essere di interesse eseguire tutte le possibili permutazioni dei dataset al fine di identificare le caratteristiche che risultano più discriminanti su sottoinsiemi specifici di attacchi.

5.2 Conclusioni

E' possibile osservare come alcune feature presentino una forte capacità discriminante per l'esecuzione di clustering.

Infatti, si nota che i cluster risultano essere adeguatamente separati e ciò è favorito dal fatto che tali feature assumono un numero limitato di valori, nonostante il dominio sia continuo.

In questi casi si verifica che il traffico benigno si limita ad assumere solo determinati valori delle feature, mentre il traffico maligno assume i restanti valori.

Di conseguenza, osserviamo una chiara separazione nello spazio tra i due tipi di traffico.

Tra le migliori feature individuate, **Init Bwd Win Byts** e **Fwd Seg Size Min** si distinguono per essere presenti il maggior numero di volte, essendo presenti entrambe in tre documenti. Inoltre, **Init Bwd Win Byts** risulta essere una delle prime due feature più importanti del file che contiene tutti gli altri.

Tra gli algoritmi di clustering utilizzati, il DBscan risulta essere quello con le prestazioni peggiori, in quanto non sempre riesce a costruire un numero di cluster corretto, le metriche che vengono calcolate attraverso la sua predizione non sono buone. Identifica troppi elementi come rumore, infine, è molto difficile calcolare gli iperparametri su dataset così grandi.

Gli algoritmi SOM, k-means e gerarchico agglomerativo restituiscono dei risultati molto simili, ma in termini prestazionali, il gerarchico agglomerativo risulta essere il più lento. Tali algoritmi restituiscono risultati ottimi quando le feature permettono di ottenere insiemi di elementi ben separati nello spazio, quindi con poca sovrapposizione. In

alcune situazioni le metriche calcolate sono altissime perché la suddivisione in cluster rispecchia la suddivisione del traffico in benigno e maligno.

In generale, gli algoritmi lavorano meglio quando il traffico è diviso in benigno e maligno, altrimenti, se proviamo a fare clustering, considerando tutti gli attacchi singolarmente, si osserva come le prestazioni peggiorino. Però, in alcuni casi le prestazioni rimangono comunque molto alte, anche se si aumenta il numero dei cluster, questo succede quando gli elementi sono ben separati nello spazio, perché, gli attacchi essendo ben separati, dal traffico benigno, sono più facili da individuare.

Si è osservato come l'algoritmo della PCA, in generale, non ha migliorato le prestazioni in maniera significativa. In alcuni casi si è osservato come abbia peggiorato le prestazioni degli algoritmi, mentre in altri, le ha migliorate.

L'algoritmo della PCA peggiora le prestazioni degli algoritmi quando il nuovo spazio delle feature che viene generato perde la separazione che avevano gli elementi nello spazio originale. Introducendo questa sovrapposizione le prestazioni degli algoritmi peggiorano.

Viceversa, quando la PCA introduce nel nuovo spazio separazione tra gli elementi, si osserva come le prestazioni degli algoritmi migliorano.

Si è osservato come lavorare sul dataset ottenuto tramite l'unione dei singoli file non abbia migliorato le prestazioni degli algoritmi, mentre, lavorare sui singoli file ci ha permesso di ottenere dei risultati migliori.

Bibliografia

- [1] *Python 3.10 documentation.* URL: <https://docs.python.org/3.10/>.
- [2] *Jupyter Notebook.* URL: <https://jupyter.org/>.

Elenco delle figure

2.1	MemoryError su macchina con 16 Gb di RAM	12
3.1	Flowchart	13
3.2	Feature che assume sempre lo stesso valore (0)	14
3.3	Feature discriminante	15
3.4	Esempio di scatterplot su feature significative	17
3.5	Tempo di esecuzione di ciascun algoritmo di clustering sul singolo file .	17
3.6	MemoryError sull'esecuzione del clustering senza riduzione del dataset .	18
3.7	Assegnazione casuale cluster	25
3.8	Assegnazione cluster senza bilanciamento	25
3.9	Allineamento cluster - classe	26
4.1	Scatter plot del dataset	30
4.2	Risultati ottenuti con k-means	30
4.3	Risultati ottenuti con DBscan	31
4.4	Risultati ottenuti con clustering gerarchico agglomerativo	31
4.5	Risultati ottenuti con SOM	32
4.6	Risultati ottenuti con k-means	33
4.7	Risultati ottenuti con DBscan	33
4.8	Risultati ottenuti con clustering gerarchico agglomerativo	34
4.9	Risultati ottenuti con clustering SOM	34
4.10	Scatter plot del dataset	35
4.11	Risultati ottenuti con k-means	35
4.12	Risultati ottenuti con DBscan	36
4.13	Risultati ottenuti con clustering gerarchico agglomerativo	36
4.14	Risultati ottenuti con SOM	37
4.15	Risultati ottenuti con Kmeans	38
4.16	Risultati ottenuti con DBscan	38
4.17	Risultati ottenuti con clustering gerarchico agglomerativo	39
4.18	Risultati ottenuti con SOM	39
4.19	Scatter plot del dataset	40
4.20	Risultati ottenuti con k-means	40
4.21	Risultati ottenuti con DBscan	41
4.22	Risultati ottenuti con clustering gerarchico agglomerativo	41

4.23 Risultati ottenuti con SOM	42
4.24 Risultati ottenuti con Kmeans	43
4.25 Risultati ottenuti con DBscan	43
4.26 Risultati ottenuti con clustering gerarchico agglomerativo	44
4.27 Risultati ottenuti con SOM	44
4.28 Scatter plot del dataset	45
4.29 Risultati ottenuti con k-means	45
4.30 Risultati ottenuti con DBscan	46
4.31 Risultati ottenuti con clustering gerarchico agglomerativo	46
4.32 Risultati ottenuti con SOM	47
4.33 Risultati ottenuti con k-means	48
4.34 Risultati ottenuti con DBscan	48
4.35 Risultati ottenuti con clustering gerarchico agglomerativo	49
4.36 Risultati ottenuti con clustering SOM	49
4.37 Scatter plot del dataset	50
4.38 Risultati ottenuti con k-means	50
4.39 Risultati ottenuti con DBscan	51
4.40 Risultati ottenuti con clustering gerarchico agglomerativo	51
4.41 Risultati ottenuti con SOM	52
4.42 Risultati ottenuti con k-means	53
4.43 Risultati ottenuti con DBscan	53
4.44 Risultati ottenuti con clustering gerarchico agglomerativo	54
4.45 Risultati ottenuti con clustering SOM	54
4.46 Scatter plot del dataset	55
4.47 Risultati ottenuti con k-means	55
4.48 Risultati ottenuti con DBscan	56
4.49 Risultati ottenuti con clustering gerarchico agglomerativo	56
4.50 Risultati ottenuti con SOM	57
4.51 Risultati ottenuti con Kmeans	58
4.52 Risultati ottenuti con DBscan	58
4.53 Risultati ottenuti con clustering gerarchico agglomerativo	59
4.54 Risultati ottenuti con SOM	59
4.55 Scatter plot del dataset	60
4.56 Risultati ottenuti con k-means	60
4.57 Risultati ottenuti con DBscan	61
4.58 Risultati ottenuti con clustering gerarchico agglomerativo	61
4.59 Risultati ottenuti con SOM	62
4.60 Risultati ottenuti con Kmeans	63
4.61 Risultati ottenuti con DBscan	63
4.62 Risultati ottenuti con clustering gerarchico agglomerativo	64
4.63 Risultati ottenuti con SOM	64
4.64 Scatter plot del dataset	65
4.65 Risultati ottenuti con k-means	65
4.66 Risultati ottenuti con DBscan	66

4.67 Risultati ottenuti con clustering gerarchico agglomerativo	66
4.68 Risultati ottenuti con SOM	67
4.69 Risultati ottenuti con Kmeans	68
4.70 Risultati ottenuti con DBscan	68
4.71 Risultati ottenuti con clustering gerarchico agglomerativo	69
4.72 Risultati ottenuti con SOM	69
4.73 Scatter plot del dataset	70
4.74 Risultati ottenuti con k-means	70
4.75 Risultati ottenuti con DBscan	71
4.76 Risultati ottenuti con clustering gerarchico agglomerativo	71
4.77 Risultati ottenuti con SOM	72
4.78 Risultati ottenuti con Kmeans	73
4.79 Risultati ottenuti con DBscan	73
4.80 Risultati ottenuti con clustering gerarchico agglomerativo	74
4.81 Risultati ottenuti con SOM	74
4.82 Scatter plot del dataset	75
4.83 Risultati ottenuti con k-means	75
4.84 Risultati ottenuti con DBscan	76
4.85 Risultati ottenuti con clustering gerarchico agglomerativo	76
4.86 Risultati ottenuti con SOM	77
4.87 Risultati ottenuti con k-means	78
4.88 Risultati ottenuti con DBscan	78
4.89 Risultati ottenuti con clustering gerarchico agglomerativo	79
4.90 Risultati ottenuti con clustering SOM	79
4.91 Scatter plot del dataset	80
4.92 Risultati ottenuti con k-means	80
4.93 Risultati ottenuti con SOM	81
4.94 Risultati ottenuti con k-means	81
4.95 Risultati ottenuti con clustering SOM	82