Elaborato di Analisi e Prestazioni di Internet

Classificazione del traffico con sistema di Multi-classificazione

Prof. Antonio Pescapè

Serena Paesano - Mat. 63/729
Gabriele De Blasi - Mat. 63/686

27 luglio 2018

Indice

Traccia

Definire e implementare tecniche per la classificazione del traffico mediante un sistema di Multi-classificazione, basato su hard/soft combiner.

Il dataset oggetto della classificazione è **ISCX CICIDS17**, il quale contiene traffico rilevato da Intrusion Detection System (IDSs) e Intrusion Prevention Systems (IPSs). Il traffico presente è sia benigno che malevolo (relativo agli attacchi più comuni) ed è memorizzato in forma di tracce di traffico PCAP ed in formato CSV contenente le feature statistiche (per biflusso).

Introduzione

Scopo di questo elaborato è valutare il comportamento di un sistema di Multi-classificazione in termini di performance rispetto ai classificatori allo stato dell'arte.
Rispetto al tipico workflow di classificazione del traffico l'approccio con sistema di Multi-classificazione prevede l'introduzione di un nuovo blocco, il *Combiner*. Pertanto, il workflow è costituito dalle seguenti fasi:

1. estrazione degli oggetti di classificazione dal traffico grezzo;
2. estrazione delle features;
3. pre-processing dei dati;
4. input delle features all'algoritmo di classificazione;
5. input al Combiner degli output dei classificatori;
6. output della classe predetta.

Il lavoro in esame inzia dalla fase 3, essendo già stato definito come oggetto di classificazione il biflusso ed essendo stato fornito il dataset con le feature relative ai biflussi (fase 1 e 2 del workflow).

1. Pre-processing dei dati

Il dataset è costituito da più file e da una prima analisi risulta essere sbilanciato a favore della classe *BENIGN*, come mostrato nella figura sottastante.

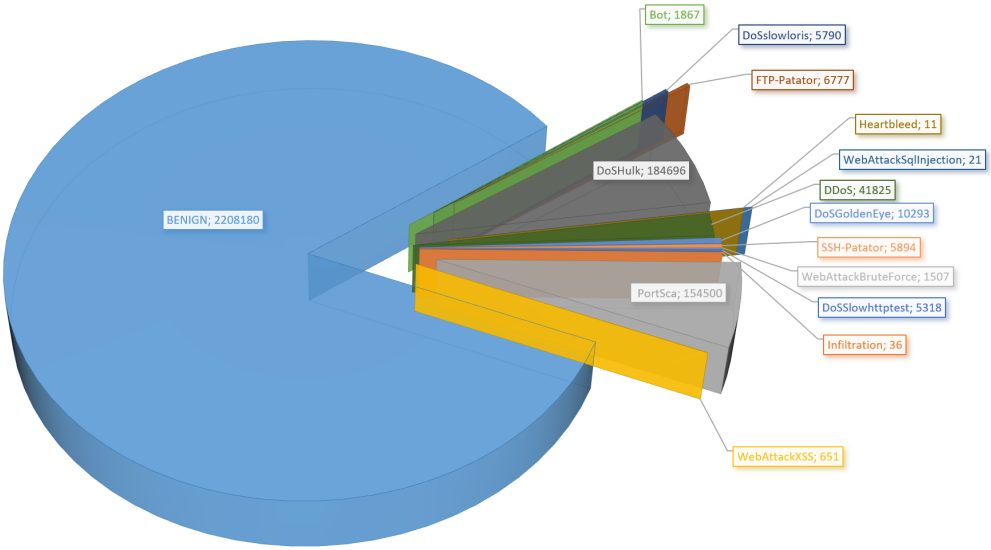


Figura 1: Distribuzione Classi

È stato quindi eseguito un "Down Sampling" della classe *BENIGN* al fine di ridurre la probabilità che si decida sempre per la classe predominante: da ogni file è stato selezionato in maniera random il 10% dei campioni corrispondenti alla classe *BENIGN*, lasciando inalterate le altre classi.

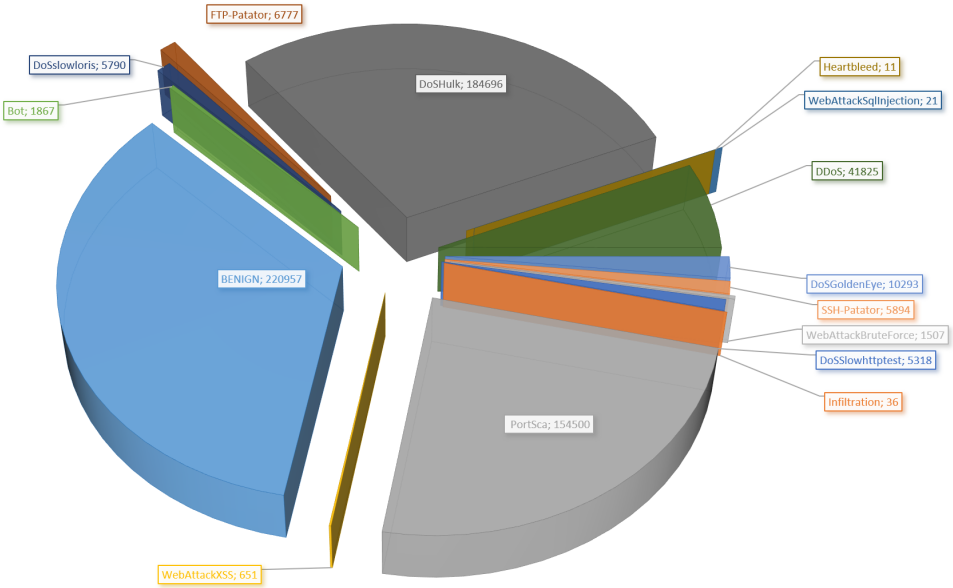


Figura 2: Distribuzione Classi Down Sampling BENIGN

Successivamente è stato effettuato il merge di questi file e poi applicate le seguenti operazioni di filtraggio:

- rimozione delle righe aventi il campo "Protocol" pari a 0;
- rimozione delle righe con "Flow Duration" minore o uguale a 10;

- rimozione delle colonne corrispondenti a “Flow ID”, alla quintupla {Source IP, Source Port, Destination IP, Destination Port, Protocol}, a “Timestamp” e a “Fwd Header Length”.

2. Classificatori allo stato dell’arte

I classificatori allo stato dell’arte implementati sono *Taylor Random Forest*, *Decision Tree (CART)* e *Gaussian Naive Bayes*. Ogni classificatore prende in ingresso un insieme di features (ottenute nella fase 2 del workflow) e produce una decisione di tipo *hard* o *soft*.

3. Combinatori

Un Combiner combina in maniera intelligente le decisioni provenienti da più classificatori e fornisce in uscita la classe predetta. I combinatori si possono categorizzare secondo varie viste, una è quella relativa alla divisione tra *hard* e *soft* combiner e l’altra è relativa alla divisione in *trainable* e *no-trainable*. I combinatori di tipo hard prendono in ingresso il vettore delle decisioni dei classificatori, mentre quelli di tipo soft ricevono in ingresso la matrice *Decision Profile*, costruita a partire dai vettori di confidenza dei classificatori. Per quanto riguarda l’altra categorizzazione, i combinatori di tipo no-trainable non prevedono una fase di addestramento, nei trainable invece è presente una fase di addestramento del combinatore.

I combinatori implementati sono:

- **Hard No-Trainable Combiner**: *Majority Voting* (MV);
- **Hard Trainable Combiner:** *Weighted Majority Voting* (WMV), *Recall Combiner* (RC), *Naive Bayes* (NB), *Behavior Knowledge Space* (BKS);
- **Soft No-Trainable Combiner**: *Avg, Max, Min*;
- **Soft Trainable Combiner**: *Decision Template* (DT).

3.1 Majority Voting Combiner

L’MV Combiner prende in ingresso, per ogni oggetto di classificazione (biflusso), le decisioni di tipo hard dei singoli classificatori e decide per la classe con il maggior numero di voti; in caso di parità si sceglie in maniera random tra le classi che presentano lo stesso numero di voti (Tie-Breaking).

3.2 Weighted Majority Voting Combiner

Il principio alla base di questo combinatore consiste nell’assegnare un peso ad ogni classificatore in modo che la decisione finale del combinatore sarà influenzata da tali pesi. Il peso di ciascun classificatore è il logaritmo della sua accuratezza totale, ossia il numero di decisioni corrette sul totale delle decisioni. Il combinatore quindi voterà per la decisione del classificatore a peso maggiore; in caso di parità si sceglie in maniera random tra le decisioni dei classificatori che presentano lo stesso peso massimo.

3.3 Recall Combiner

Il Recall Combiner decide utilizzando come pesi l'accuratezza dei classificatori in funzione della classe, vale a dire la Recall per classe. Qualora si abbiano più classi con il medesimo peso massimo si sceglie in maniera random tra una di queste.

3.4 Naive Bayes Combiner

Il Naive Bayes Combiner determina la confidenza di una classe combinando l'accuratezza e il pattern di errore di ogni classificatore per quella classe. Quindi il combinatore sceglie per la classe a confidenza maggiore e, ancora una volta, se si hanno più classi con la medesima confidenza, si decide in maniera random.

3.5 Behavior Knowledge Space Combiner

Il BKS Combiner decide per una classe dopo aver appreso il comportamento assunto dai classificatori quando sottoposti al medesimo input: in particolare, per ogni nuovo biflusso da classificare, il pool di classificatori genera un pattern di decisioni e si conta, per ciascuna classe, quante volte questo pattern ricorre tra le predizioni dei classificatori quando in input ricevono il validation set. Tale conteggio serve a determinare i pesi delle classi. La classe predetta sarà quella avente il peso più grande. Nella situazione in cui più classi ottengono lo stesso punteggio si adotta la regola di Tie-Breaking già vista.

3.6 Soft Avg, Max, Min Combiner

Questo combinatore riceve dai classificatori i soft output, vettori costituiti dalle confidenze delle classi, e a seconda della funzione di combinazione (avg, max o min) genara un nuovo soft output e sceglie per la classe avente la confidenza più alta. Vale, come nei combinatori precedenti, la stessa regola di Tie-Breaking.

3.7 Decision Template Combiner

Il combinatore Decision Template calcola per ciascuna classe il Decision Profile medio e lo confronta con il Decision Profile corrispondente ad un nuovo oggetto di classificazione, in termini di distanza Euclidea. La decisione ricade sulla classe che presenta la distanza minima. Se più classi hanno la stessa distanza si sceglie in modo casuale tra queste.

4. Valutazione delle prestazioni

Le performance del sistema di Multi-classificazione sono state valutate attraverso la tecnica *Stratified Ten-Fold Validation*, ossia una procedura iterativa che consta di 10 passi e ad ogni passo il dataset viene suddiviso in due parti, di cui 9/10 rappresentano il training set e 1/10 il testing set. Essendo stratificata inoltre si ha la sicurezza che in ogni set cada almeno un campione per ciascuna classe.

Nel caso di combinatore trainable, dal momento che è prevista una fase di addestramento del combiner, ad ogni passo della Ten-Fold il training set viene ulteriormente diviso in un set per

l'addestramento dei classificatori (traning set, 70%) e un set per l'addestramento del combinatore (validation set, 30%).

Le metriche utilizzate per l'analisi delle prestazioni sono: Accuracy, Macro-Recall, Macro-Precision, Macro F-Measure, Macro G-Mean e matrice di confusione.

4.1 Prestazioni Classificatori allo stato dell'arte

Di seguito sono riportate le matrici di confusione e le prestazioni dei classificatori allo stato dell'arte implementati.

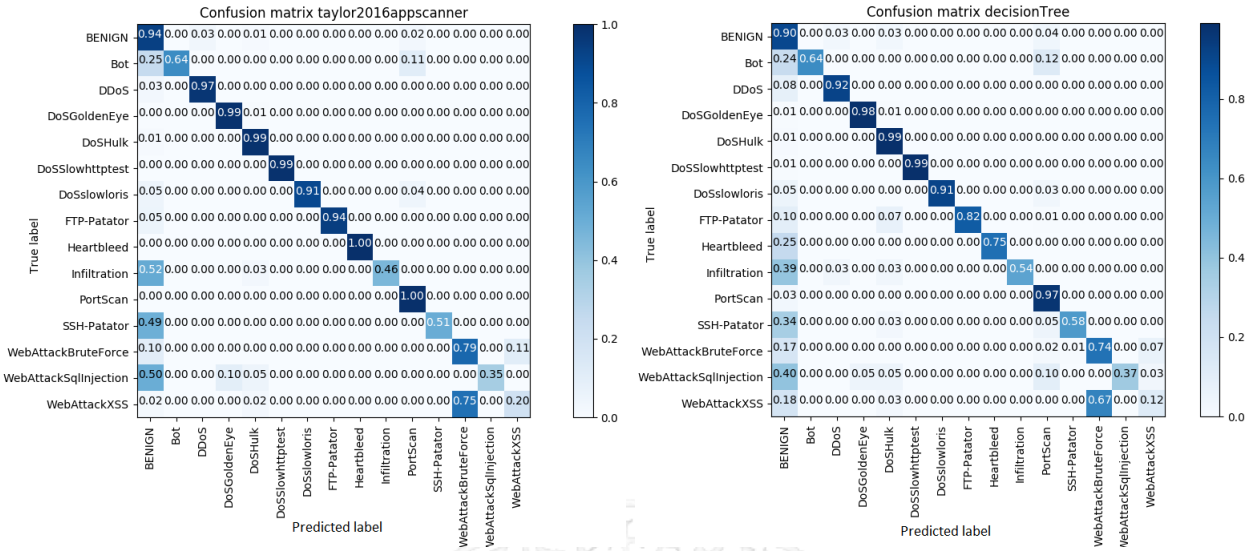


Figura 3: Taylor Random Forest - Decision Tree

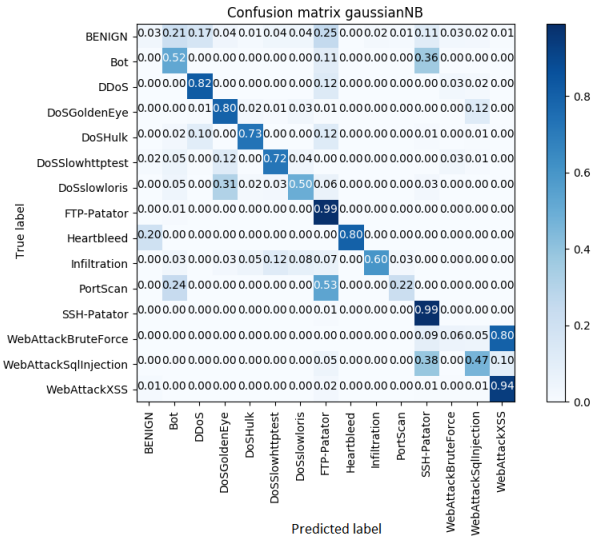


Figura 4: Gaussian Naive Bayes

	RF	DT	GNB
Accuracy	96,3916% (±0,1852%)	94,1258% (±1,5997%)	37,5018% (±0,3952%)
Macro Recall	77,8098% (±1,7432%)	74,8354% (±3,7565%)	61,2452% (±4,5342%)
Macro Precision	87,7715% (±4,3218%)	83,7143% (±4,5437%)	35,9937% (±2,7610%)
Macro F-Measure	81,2561% (±1,8488%)	76,9217% (±3,5473%)	30,6621% (±2,8385%)
Macro G-Mean	85,0944% (±2,0944%)	83,0383% (±3,7869%)	71,0135% (±4,4354%)

Figura 5: Performance Classificatori

Dalle figure di sopra si evince che il classicatore con le migliori prestazioni è il Taylor Random Forest e per tale motivo verrà utilizzato nella comparazione delle performance con i combinatori.

4.2 Prestazioni Combinatori

Al fine di valutare le performance è stato introdotto il Combinatore Oracolo, che costituisce un Upper-Bound per le prestazioni. Si ricorda che l'oracolo è un combinatore che classifica correttamente se almeno uno dei classificatori del pool decide correttamente.

	RF	MV	WMV	Recall	NB	BKS	Soft (Avg)	DecisionTemplate	Oracle
Accuracy	96,3916%	95,0842%	96,4570%	95,2923%	89,1658%	96,4738%	95,1292%	93,5589%	97,7143%
Macro Recall	77,8098%	79,5122%	76,5710%	72,8777%	81,8163%	75,7648%	81,8291%	81,9548%	92,3056%
Macro Precision	87,7715%	76,5456%	85,4383%	83,3967%	68,8274%	74,9014%	80,4080%	74,7851%	
Macro F-Measure	81,2561%	76,7014%	79,4361%	74,2107%	67,8333%	70,0351%	79,5032%	73,7842%	
Macro G-Mean	85,0944%	86,5626%	84,2504%	79,4728%	87,2426%	80,8487%	87,9877%	88,2338%	

Figura 6: Performance

Osservando le prestazioni si nota che sia il classificatore Taylor Random Forest che alcuni combinatori presentano dei valori che non si discostano molto dalle prestazioni del combinatore Oracolo. In particolare, per il classificatore di base l'incremento massimo che si può ottenere è circa del 1,32% e 14,50% rispettivamente per l'Accuracy e la Macro-Recall.

4.2.1 Prestazioni Hard Combiner

Considerando i combinatori di tipo Hard, trainable e no-trainable, si osserva che i trainable presentano prestazioni migliori. Nello specifico, il BKS ha un'Accuracy maggiore delle altre, il Weighted Majority Voting ha Macro Precision e Macro F-Measure più alte, mentre per quanto riguarda Macro Recall e Macro G-Mean, i migliori valori sono ottenuti dal Naive Bayes.

In definitiva, sulla base del fatto che la metrica Macro F-measure tiene conto sia della Recall che della Precision e poiché il BKS e il WMV hanno un'Accuracy simile, si è scelto come combinatore Hard più performante il Weighted Majority Voting.

	MV	WMV	Recall	NB	BKS	Oracle
Accuracy	95,0842%	96,4570%	95,2923%	89,1658%	96,4738%	97,7143%
Macro Recall	79,5122%	76,5710%	72,8777%	81,8163%	75,7648%	92,3056%
Macro Precision	76,5456%	85,4383%	83,3967%	68,8274%	74,9014%	
Macro F-Measure	76,7014%	79,4361%	74,2107%	67,8333%	70,0351%	
Macro G-Mean	86,5626%	84,2504%	79,4728%	87,2426%	80,8487%	

Figura 7: Performance Hard Combiner

4.2.2 Prestazioni Soft Combiner

Per quanto concerne il combinatore Soft di tipo no-trainable, esistendo varie funzioni di combinazione, quella che presenta le migliori prestazioni è l'Average, pertanto sarà considerato questo come combinatore Soft no-trainable di riferimento.

Come mostrato nella figura sottostante, in termini di Accuracy, Macro Precision e Macro F-Measure, l'Average Soft Combiner è quello con risultati migliori, il Decision Template lo è per le altre metriche.

Con ragionamenti analoghi fatti per gli Hard Combiner, decidiamo per l'Average Soft Combiner come Soft combiner più performante.

	Soft (Avg)	DecisionTemplate	Oracle
Accuracy	95,1292%	93,5589%	97,7143%
Macro Recall	81,8291%	81,9548%	92,3056%
Macro Precision	80,4080%	74,7851%	
Macro F-Measure	79,5032%	73,7842%	
Macro G-Mean	87,9877%	88,2338%	

Figura 8: Performance Soft Combiner

4.2.3 Confronto Prestazioni

Una prima considerazione può essere fatta sui tempi di esecuzione: i classificatori e combinatori implementati hanno impiegato tempi dell'ordine delle decine di minuto, pertanto non si è percepita una considerevole differenza tra loro. Un'eccezione è rappresentata dal BKS Combiner, il quale richiederebbe più di 4 ore per fold se non fosse prevista alcuna ottimizzazione. Infatti, parallelizzando la fase di apprendimento del combinatore BKS si è riusciti a ridurre la durata di ogni fold a meno di 2 ore e mezzo, per una durata totale di 23 ore circa.

Si confrontano ora le perfomance del miglior classificatore con quelle dei migliori combinatori Hard e Soft. Oltre alle metriche già viste in precedenza, consideriamo le matrici di confusione che ci consentono di avere una panoramica anche sui pattern di errore.

	RF	WMV	Soft (Avg)	Oracle
Accuracy	96,3916%	96,4570%	95,1292%	97,7143%
Macro Recall	77,8098%	76,5710%	81,8291%	92,3056%
Macro Precision	87,7715%	85,4383%	80,4080%	
Macro F-Measure	81,2561%	79,4361%	79,5032%	
Macro G-Mean	85,0944%	84,2504%	87,9877%	

Figura 9: Performance

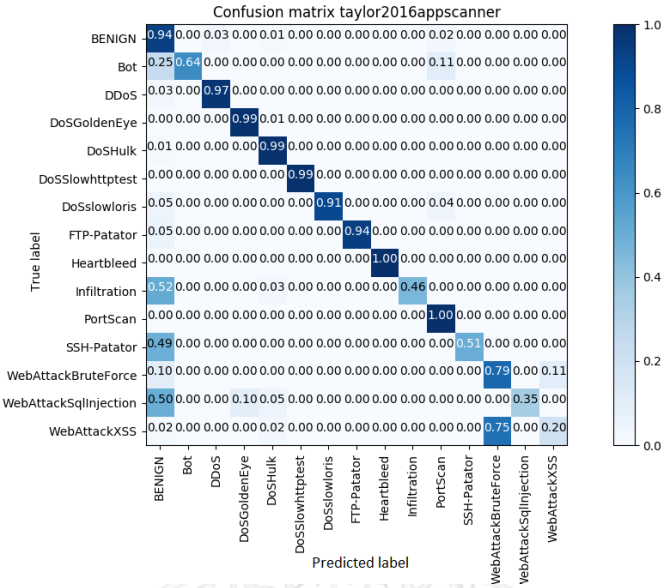


Figura 10: Matrice di confusione Taylor Random Forest

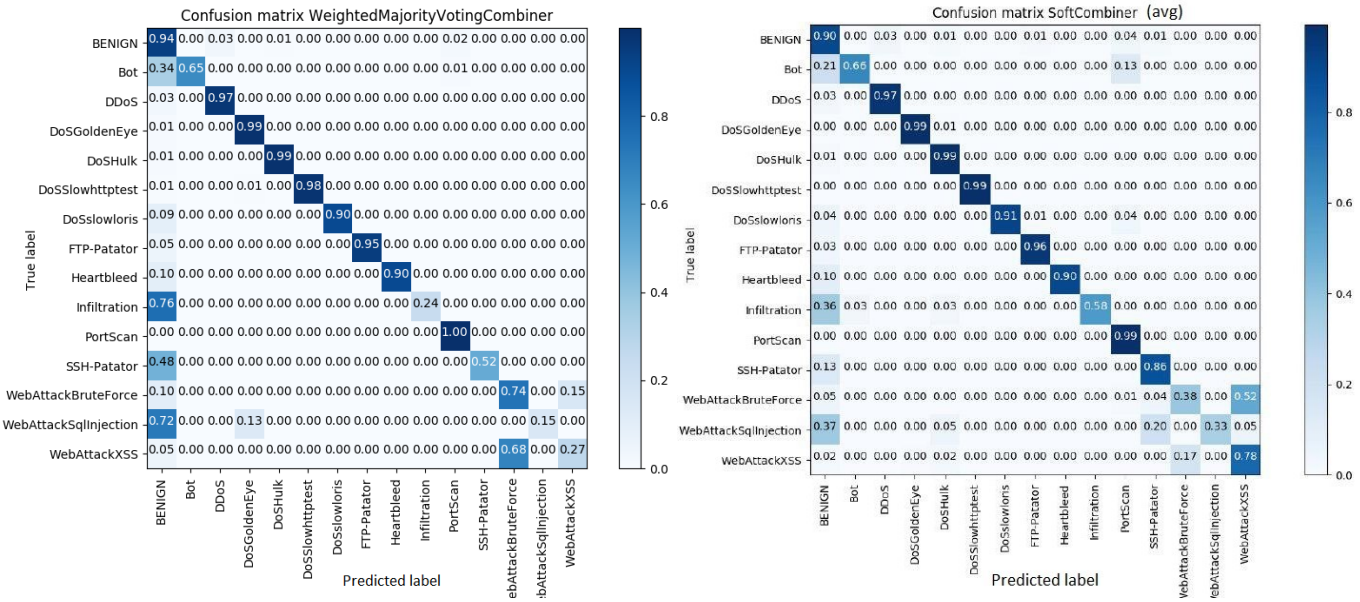


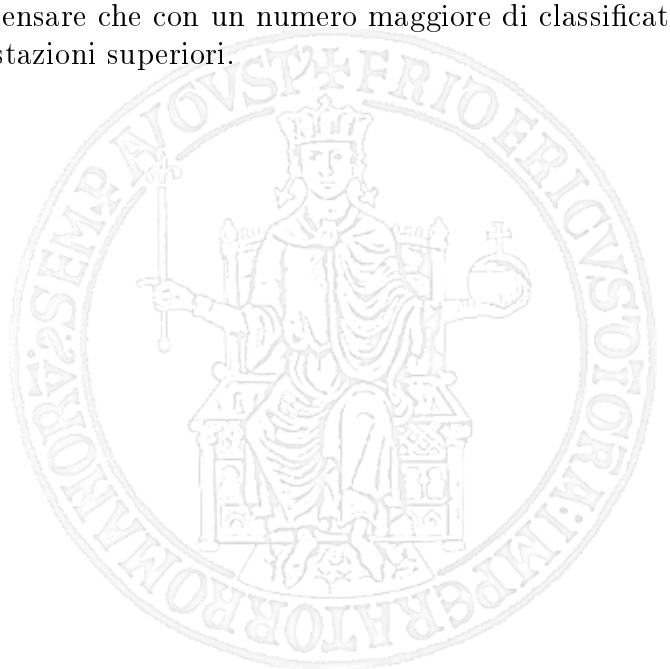
Figura 11: Matrici di confusione Weighted Majority Voting e Average Soft

Dai valori di performance mostrati in Figura 9, si nota che per alcuni valori (Accuracy e Macro-Recall) il sistema di Multi-classificazione è migliore rispetto al classificatore di base; nonostante ciò, non si riscontra un notevole incremento delle prestazioni. Prendendo in considerazione le matrici di confusione, è possibile però giungere a conclusioni più significative riguardo la classifizione del traffico. Le tre matrici di confusione manifestano un comportamento abbastanza simile sulla diagonale principale, la quale da' informazioni su quanto un classificatore decida bene per le classi. La differenza sostanziale sta nella colonna relativa alla predizione della classe BENIGN: il Taylor Random Forest e il Weighted Majority Voting tendono a classificare biflussi relativi ad attacchi (Bot, Infiltration, SSH-Patetor, WebAttackBruteForce e WebAttackSQLInjection) come traffico benigno; l'Average Soft, invece, mitiga questo comportamento discriminando meglio il traffico malevolo da quello benigno. Alla luce di queste ultime considerazioni, è possibile giungere alla conclusione che il sistema di Multi-classificazione costituito dai tre classificatori di base ed il combinatore Average Soft è preferibile rispetto ad un sistema di classificazione con il solo classificatore di base.

Conclusioni

I risultati ottenuti hanno messo in luce i vantaggi nell'utilizzo di un sistema di Multi-classificazione. Inoltre, si osserva che il combinatore Average Soft (no-trainable) non solo è il migliore tra i combinatori implementati ma è anche uno dei più semplici. Quindi, non è necessario avere complessità maggiore per performare meglio.

Infine, si precisa che i risultati sono stati ottenuti con un pool limitato di tre classificatori e pertanto si è portati a pensare che con un numero maggiore di classificatori allo stato dell'arte si potrebbero ottenere prestazioni superiori.



Appendice

Tutti gli scripts sono stati realizzati in Python e le principali librerie utilizzate sono *numpy*, *scipy*, *sklearn* e *imblearn*.

Di seguito viene fornito l'elenco degli script utilizzati e le funzioni realizzate per implementare i combinatori.

Script:

- **genPickle.sh**, script che richiama:
  1. *down_sampling.py*, down sampling della classe BENING;
  2. *mergeDataset_script.py*, merge dei file che compongono il dataset;
  3. *delRow.py*, rimozione delle righe per la fase di pre-processing;
  4. comando *cut*, rimozione delle colonne per la fase di pre-processing;
  5. *dataset_preparation.py*, generazione del file in formato *pickle*.
- **machine_learning_architectures.py**, script usato per l'implementazione dei classificatori e combinatori e la valutazione delle prestazioni;
- **plot_confusion_matrix.py**, script per la generazione delle matrici di confusione.

Per quanto riguarda l'implementazione dei combinatori, per ognuno di essi sono state previste le funzioni di:

- Fit_*nomeCombiner*, per addestrare il combinatore di tipo trainable;
- Predict_*nomeCombiner*, per produrre le predizioni hard/soft dei combinatori.

È prevista inoltre una funzione (*nomeCombiner*_10Fold) che definisce il passo della Ten-Fold Validation per il combinatore: invoca la Fit e la Predict di quel combinatore e al contempo stima le performance per quel particolare passo.

Nella figura di sotto sono illustrate un esempio di funzioni che implementano un combinatore.

```
def Fit Recall (self, prediction list, validation labels):

def Predict Recall (self, prediction list, confidence class matrix):

def Recall Combiner 10Fold (self,samples train, categorical labels train, samples test, categorical labels test):
```

Figura 12: Funzioni Recall Combiner