



Università Politecnica delle Marche

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica e
dell'Automazione

CORSO DI DATA SCIENCE

**Classificazione e Clustering sulla tipologia di traffico
Internet e Regressione su serie temporale per la
previsione del traffico di rete**

Professore

Prof. Domenico Ursino

Studenti

Matteo Risolo

Niccolò De Pascali

Indice

1	Dataset CIC-IDS-2017	5
1.1	Descrizione del dataset	5
1.2	ETL	7
1.3	Data Visualizzazione	10
2	Classificazione	16
2.1	Pre-processing	17
2.2	Training	23
2.3	Analisi dei risultati	25
2.4	Greed Search	30
2.5	Model Ensembling	33
2.5.1	Random Forest + Gradient Boosting	33
2.5.2	Random Forest + Logistic Regression	36
3	Clustering	40
3.1	Pre-processing	41
3.2	K-Means	41
3.2.1	Clustering Gerarchico	45
3.3	PCA	49
3.4	DBSCAN	52
4	Dataset UGR'16	55
4.1	Contesto di riferimento	55

4.1.1	L'importanza del Network Traffic Forecasting	55
4.1.2	Natura delle Serie Temporali nel Traffico Dati	56
4.2	Il dataset	57
4.3	Struttura del dataset	58
5	Regessione su serie temporale	65
5.1	Pre-processing	65
5.1.1	Parsing temporale e indicizzazione	65
5.1.2	Analisi della regolarità temporale	66
5.1.3	Definizione della variabile target	69
5.1.4	Creazione e analisi della serie al minuto	70
5.1.5	Creazione e analisi della serie oraria	75
5.1.6	Ricostruzione del profilo della serie oraria	77
5.1.7	Taglio della serie	84
5.2	Analisi preliminare della stabilità	86
5.3	Decomposizione STL	87
5.4	Verifica della stazionarietà	90
5.5	Analisi di ACF e PACF	93
5.6	Scelta del modello migliore	96
5.7	Forecast finale e valutazione risultati	100
5.8	Inserimento variabili esogene	103

Capitolo 1

Dataset CIC-IDS-2017

1.1 Descrizione del dataset

Il dataset **CIC-IDS-2017** (Intrusion Detection Evaluation Dataset) è un dataset pubblico per la ricerca in sicurezza informatica, sviluppato dal Canadian Institute for Cybersecurity (CIC) e rilasciato nel 2017. È concepito come standard realistico per l'addestramento e la valutazione di modelli di Intrusion Detection System (IDS) basati su tecniche di machine learning e data mining.

Il dataset è stato generato attraverso la cattura di traffico di rete in un ambiente controllato, includendo sia traffico legittimo, sia molteplici tipologie di attacchi informatici, tra cui attacchi Brute Force, Denial of Service (DoS / DDoS), Heartbleed, Web Attack, Infiltration e Botnet, in modo da riflettere scenari reali di minacce. Il traffico è stato registrato come pacchetti e successivamente elaborato in flussi etichettati, con annotazioni relative a indirizzi IP, porte, protocolli e classi di attacco. La raccolta si è svolta in un periodo di 5 giorni consecutivi (dal 3 al 7 luglio 2017), includendo un giorno dedicato esclusivamente a traffico legittimo e quattro giorni con vari attacchi eseguiti in orari specifici per simulare attività malevole. Il dataset è reso disponibile al seguente link : <https://www.unb.ca/cic/datasets/ids-2017.html>, ed è diviso in più file CSV opportunamente etichettati per facilitare l'utilizzo in

analisi supervisionate e non supervisionate.

Il dataset è composto complessivamente da **79 feature**, le quali descrivono proprietà temporali, statistiche sui pacchetti, informazioni di protocollo e indicatori di attività del flusso. Ai fini dell’analisi, risulta opportuno concentrarsi sulle feature più rilevanti dal punto di vista descrittivo e discriminativo:

- **Destination Port**: porta di destinazione del flusso; fornisce indicazioni sul servizio di rete coinvolto (es. HTTP, HTTPS, FTP), risultando utile per distinguere traffico lecito da attività anomale.
- **Flow Duration**: durata complessiva del flusso; valori estremi o anomali sono spesso associati ad attacchi DoS/DDoS o a tentativi di scansione.
- **Total Fwd Packets / Total Backward Packets**: numero totale di pacchetti inviati e ricevuti nel flusso.
- **Total Length of Fwd/Bwd Packets**: volume di byte trasmessi nelle due direzioni.
- **Fwd/Bwd Packet Length**: statistiche sulla dimensione dei pacchetti, fondamentali per caratterizzare pattern di traffico anomalo, come flussi molto piccoli e ripetitivi tipici di attacchi automatici.
- **Flow Bytes/s e Flow Packets/s**: indicano la velocità di trasmissione in termini di byte e pacchetti al secondo; valori elevati sono indicatori tipici di attacchi di tipo flooding.
- **Fwd Packets/s / Bwd Packets/s**: distribuzione direzionale del traffico nel tempo.
- **SYN, ACK, FIN, RST, PSH, URG Flag Count**: il conteggio dei flag TCP consente di individuare comportamenti sospetti come SYN flood, connessioni incomplete o reset frequenti, tipici di diversi attacchi di rete.

- **Packet Length/Average Packet Size**: offrono una visione complessiva della distribuzione dimensionale dei pacchetti all'interno del flusso.
- **Active/Idle**: descrivono i periodi di attività e inattività del flusso; sono particolarmente utili per individuare comportamenti intermittenti o burst tipici di botnet e attività di scansione.
- **Label**: identifica la classe del flusso, distinguendo traffico Benign e diverse tipologie di attacco (DoS, DDoS, Brute Force, Web Attack, Botnet, ecc.). Tale colonna costituisce il riferimento per le analisi supervisionate e viene esclusa nelle tecniche di clustering non supervisionato.

1.2 ETL

La fase di **ETL (Extract, Transform, Load)** rappresenta un passaggio fondamentale nel processo di analisi dei dati, in quanto consente di rendere il dataset idoneo alle successive attività di data exploration, classification e clustering. Nel contesto del dataset CIC-IDS-2017, l'ETL è particolarmente rilevante a causa dell'elevato numero di istanze, della presenza di valori anomali, come ad esempio valori infiniti o nulli, e della forte eterogeneità delle feature di rete.

La prima operazione eseguita nella fase di ETL ha riguardato l'unificazione dei file CSV che compongono il dataset CIC-IDS-2017. Come è stato detto precedentemente, il dataset, infatti, è distribuito sotto forma di più file distinti, ciascuno relativo a specifiche sessioni temporali di acquisizione del traffico di rete. Per poter effettuare un'analisi coerente e uniforme sull'intero insieme di dati, si è reso necessario consolidare tali file in un'unica struttura tabellare. Il processo di unificazione è stato realizzato mediante la lettura sequenziale di tutti i file CSV presenti nella directory di lavoro, garantendo un ordine deterministico nella loro elaborazione. Ogni file è stato caricato mantenendo inalterata la struttura originale delle feature, e successivamente, i singoli data-

set sono stati aggregati per righe, ottenendo un unico DataFrame contenente l'intero volume di osservazioni del traffico di rete registrato nel periodo di riferimento. Questa operazione consente quindi di trattare il dataset come un insieme omogeneo di flussi di rete, semplificando le successive fasi di pulizia, trasformazione e analisi dei dati.

```
# percorso base della cartella che contiene i CSV
base_dir = Path(
    r"C:\Users\nicde\OneDrive\Desktop\Università\Ancona\DS\Data-Science\datasets"
) / "MachineLearningCSV" / "MachineLearningCVE"

# elenco ordinato di tutti i file CSV
csv_files = sorted(base_dir.glob("*.csv"))

print(f"Numero di file CSV trovati: {len(csv_files)}")

dfs = []

# lettura sequenziale dei CSV
for i, file in enumerate(csv_files, 1):
    print(f"Caricamento file {i}/{len(csv_files)} - {file.name}")
    df = pd.read_csv(file, low_memory=False)
    dfs.append(df)

# concatenazione row-wise
data = pd.concat(dfs, ignore_index=True)

# controlli rapidi
data.head()
```

Figura 1.1: Unificazione dei file CSV

	Destination Port	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Fwd Packet Length Std	... min_seg_size_forward	Active Mean	Active Std	Active Max	Active Min
0	54865	3	2	0	12	0	6	6	6.0	0.0	...	20	0.0	0.0	0
1	55054	109	1	1	6	6	6	6	6.0	0.0	...	20	0.0	0.0	0
2	55055	52	1	1	6	6	6	6	6.0	0.0	...	20	0.0	0.0	0
3	46236	34	1	1	6	6	6	6	6.0	0.0	...	20	0.0	0.0	0
4	54863	3	2	0	12	0	6	6	6.0	0.0	...	20	0.0	0.0	0

5 rows × 79 columns

Figura 1.2: Visualizzazione delle prime 5 righe del dataset unificato

Successivamente all'unificazione dei file CSV, è stata eseguita una fase di pulizia e normalizzazione strutturale del dataset, finalizzata a garantire coerenza sintattica, correttezza semantica delle feature e compatibilità con le successive tecniche di analisi esplorativa e machine learning. In primo luogo, è stata effettuata una normalizzazione dei nomi delle colonne, rimuovendo eventuali spazi accidentali presenti all'inizio o alla fine delle intestazioni. Un'analogia operazione di pulizia sintattica è stata applicata alla variabile target Label, al fine di garantire la corretta identificazione delle classi e prevenire duplicazio-

ni semantiche dovute a differenze puramente formali. Successivamente, sono state rimosse alcune colonne ritenute non informative o non direttamente utili ai fini dell’analisi, in particolare, la variabile Destination Port è stata esclusa per ridurre la dipendenza da informazioni di tipo identificativo e favorire l’apprendimento basato su caratteristiche comportamentali del traffico di rete. In questa versione del dataset, già predisposta per lavorare con il machine learning, molte variabili ritenute non informative sono state già precedente rimosse da chi ha sviluppato il dataset. Per questo motivo nel seguente lavoro si è deciso di eliminare solo la variabile Destination Port. Una volta definite le feature di input, tutte le variabili esplicative sono state convertite in formato numerico. Eventuali valori non convertibili, o infiniti, sono stati automaticamente trasformati in valori mancanti (NaN). Successivamente poi, le osservazioni contenenti almeno un valore NaN sono state rimosse, producendo un dataset finale completamente privo di valori mancanti. Questa fase di pulizia e trasformazione costituisce un passaggio cruciale della pipeline ETL, in quanto assicura che i dati utilizzati nelle successive fasi siano coerenti e privi di anomalie strutturali.

```
# Normalizzazione nomi colonne
data.columns = data.columns.str.strip()

# Pulizia sintattica Label (tenuta solo per valutazione ex post)
if "Label" in data.columns:
    data["Label"] = data["Label"].astype(str).str.strip()

# Rimozione colonne identificative / temporali non utili
# questa versione del dataset è già priva di molte colonne non utili per machine learning
cols_to_drop = [
    "Destination Port"
]
data.drop(columns=[c for c in cols_to_drop if c in data.columns], inplace=True)

# Conversione numerica solo delle feature, label rimane string
feature_cols = [c for c in data.columns if c != "Label"]
data[feature_cols] = data[feature_cols].apply(pd.to_numeric, errors="coerce")

# Sostituzione inf con NaN
data.replace([np.inf, -np.inf], np.nan, inplace=True)

data.info()

# colonne con più NaN
data[feature_cols].isna().sum(axis=0).sort_values(ascending=False).head(20)

# Drop righe con NaN sulle sole feature
data = data.dropna(subset=feature_cols).copy()

# separazione features e label
if "Label" in data.columns:
    y_label = data["Label"].reset_index(drop=True)
    X = data.drop(columns=["Label"]).reset_index(drop=True)
else:
```

Figura 1.3: Operazioni di pulizia del dataset

	Flow Duration	Total Fwd Packets	Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets	Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean	Bwd Packet Length Std	... min seg size forward	Active Mean	Active Std	Active Max	Active Min	I M
0	3	2	0	12	0	6	6	6.0	0.0	0	20	0.0	0.0	0	0
1	109	1	1	6	6	6	6	6.0	0.0	6	20	0.0	0.0	0	0
2	52	1	1	6	6	6	6	6.0	0.0	6	20	0.0	0.0	0	0
3	34	1	1	6	6	6	6	6.0	0.0	6	20	0.0	0.0	0	0
4	3	2	0	12	0	6	6	6.0	0.0	0	20	0.0	0.0	0	0

5 rows × 78 columns

Figura 1.4: Visualizzazione delle prime 5 righe del dataset dopo le operazioni di pulizia effettuate

1.3 Data Visualization

A seguito delle operazioni di estrazione, pulizia e trasformazione dei dati, è stata condotta una fase di **Exploratory Data Analysis (EDA)** con l’obiettivo di analizzare la struttura del dataset e valutare la distribuzione delle feature e delle classi. L’EDA rappresenta un essenziale strumento per comprendere il comportamento statistico delle variabili, rilevare eventuali anomalie residue e guidare le scelte metodologiche relative alla selezione delle feature e agli algoritmi di classificazione e clustering adottati.

In questo ambito sono state realizzati diversi grafici con l’obiettivo di supportare l’analisi descrittiva del dataset e di fornire una comprensione preliminare delle sue caratteristiche principali. In particolare, i grafici consentono infatti di affiancare all’analisi numerica una lettura intuitiva del dataset, rendendo immediatamente evidenti fenomeni come la predominanza di alcune classi di traffico, la rarità di specifiche tipologie di attacco e la presenza di forti squilibri tra classi, aspetti particolarmente rilevanti nel contesto dei dataset per Intrusion Detection.

Il primo grafico analizzato rappresenta la distribuzione delle classi presenti nel dataset CIC-IDS-2017, espressa in termini di numero di flussi per ciascuna categoria di traffico.

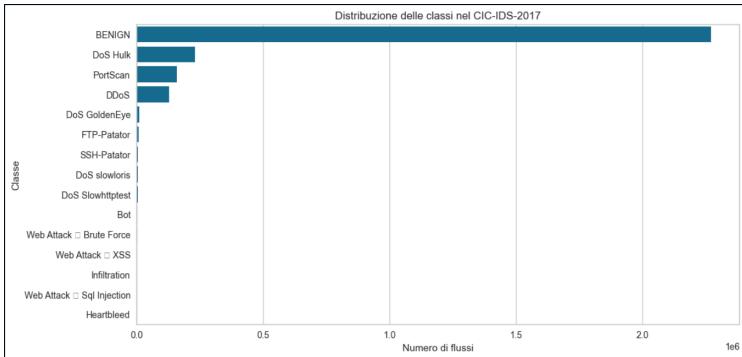


Figura 1.5: Distribuzione delle classi presenti nel dataset

La visualizzazione evidenzia in modo immediato un forte sbilanciamento del dataset, con una netta predominanza della classe **BENIGN**, che costituisce la maggior parte delle osservazioni disponibili. Tra le classi di attacco, le più rappresentate risultano essere **DoS Hulk**, **PortScan** e **DDoS**, mentre le restanti tipologie presentano una numerosità significativamente inferiore. Alcune classi risultano estremamente rare, configurandosi come eventi a bassa frequenza. Questa distribuzione riflette fedelmente la natura realistica del traffico di rete, in cui gli eventi malevoli costituiscono una frazione limitata rispetto al traffico lecito. Tuttavia, dal punto di vista metodologico, lo sbilanciamento tra classi rappresenta un aspetto critico per le successive fasi di analisi poiché può influenzare le prestazioni dei modelli e richiedere l'adozione di strategie specifiche di gestione del class imbalance.

La stessa analisi può essere effettuata sul secondo grafico che rappresenta la distribuzione percentuale delle classi presenti nel dataset e costituisce una normalizzazione del grafico precedente, consentendo una lettura immediata del peso relativo di ciascuna categoria.

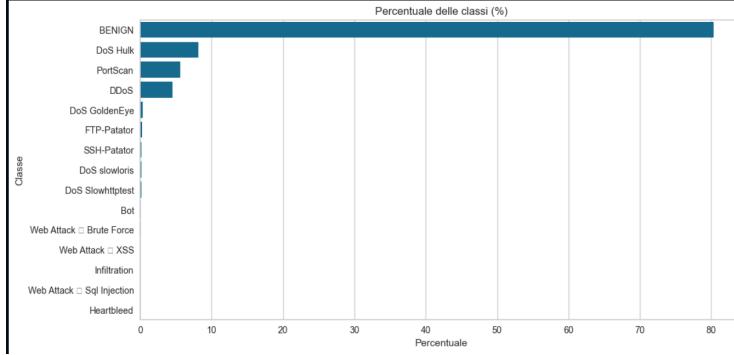


Figura 1.6: Distribuzione percentuale delle classi

La visualizzazione conferma il marcato sbilanciamento del dataset, con la classe BENIGN che rappresenta la quota largamente predominante delle osservazioni, seguita a distanza dalle classi DoS Hulk, PortScan e DDoS, mentre le restanti tipologie di attacco incidono in misura marginale.

Il terzo grafico rappresenta la distribuzione della durata dei flussi (Flow Duration) per ciascuna classe di traffico, consentendo di confrontare il comportamento temporale del traffico benigno rispetto alle diverse tipologie di attacco. L'utilizzo della scala logaritmica sull'asse delle ascisse permette di evidenziare la presenza di flussi con durate che variano su più ordini di grandezza.

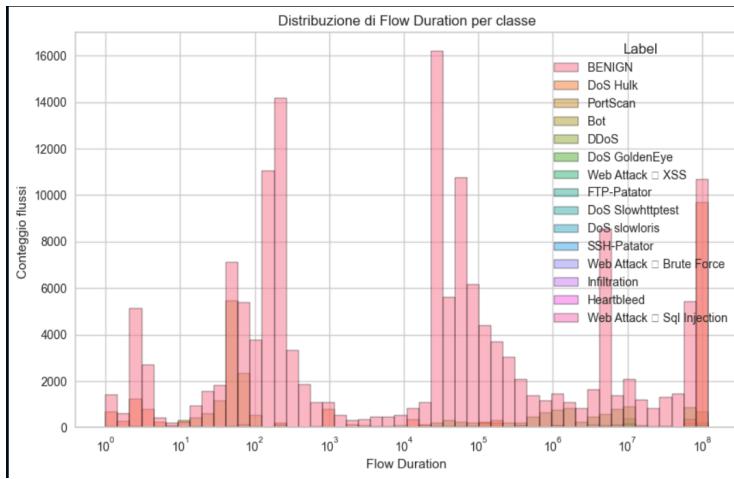


Figura 1.7: Distribuzione della durata dei flussi per classe

Dal confronto emerge come il traffico BENIGN presenti una distribuzione molto più ampia e dispersa, indicativa di sessioni di comunicazione eterogenee

e di durata variabile. Al contrario, molte classi di attacco risultano caratterizzate da distribuzioni più concentrate, con durate dei flussi tipicamente più brevi o ripetitive, coerenti con comportamenti automatici e non interattivi, in particolare attacchi di tipo DoS/DDoS e PortScan tendono a generare flussi di durata relativamente breve o molto uniforme, mentre altre tipologie risultano caratterizzate da durate più limitate e meno variabili. Questa differenza nel profilo temporale dei flussi suggerisce che la Flow Duration rappresenti una variabile informativa rilevante per distinguere traffico lecito e malevolo.

Il grafico seguente mostra la distribuzione del numero di pacchetti inviati in direzione forward (Total Fwd Packets) per ciascuna classe di traffico, rappresentata in scala logaritmica per evidenziare la forte asimmetria dei valori. La visualizzazione consente di confrontare il comportamento volumetrico del traffico benigno rispetto alle diverse tipologie di attacco.

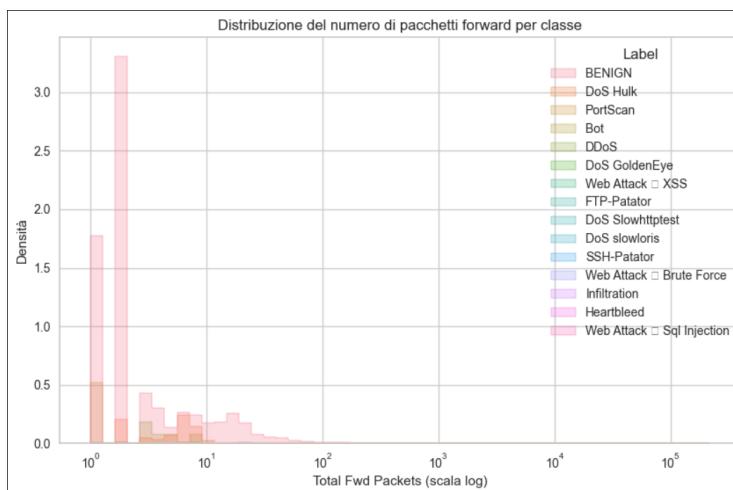


Figura 1.8: Distribuzione dei Total Fwd Packets per classe

Dal grafico emerge come il traffico BENIGN presenti una distribuzione più ampia e dispersa, con flussi che possono includere un numero elevato di pacchetti, coerentemente con sessioni applicative di durata variabile. Al contrario, molte classi di attacco risultano concentrate su valori più bassi del numero di pacchetti forward, suggerendo flussi brevi e ripetitivi, tipici di attività automatizzate come scansioni, brute force o attacchi di tipo DoS. La separazione

parziale tra le distribuzioni delle diverse classi indica che il numero di pacchetti forward costituisce una feature informativa rilevante. In particolare, la diversa densità dei flussi in specifici intervalli di valori rende questa variabile utile sia per la discriminazione supervisionata tra traffico benigno e malevolo, sia per l'individuazione di pattern omogenei nelle tecniche di clustering non supervisionato.

L'ultimo grafico analizzato rappresenta invece la relazione tra la durata del flusso (Flow Duration) e il numero totale di pacchetti forward (Total Fwd Packets), mediante una visualizzazione, anche in questo caso, in scala logaritmica su entrambi gli assi. L'obiettivo di questa rappresentazione è quello di analizzare il comportamento congiunto di due feature chiave e valutare la presenza di pattern distintivi tra traffico benigno e diverse tipologie di attacco.

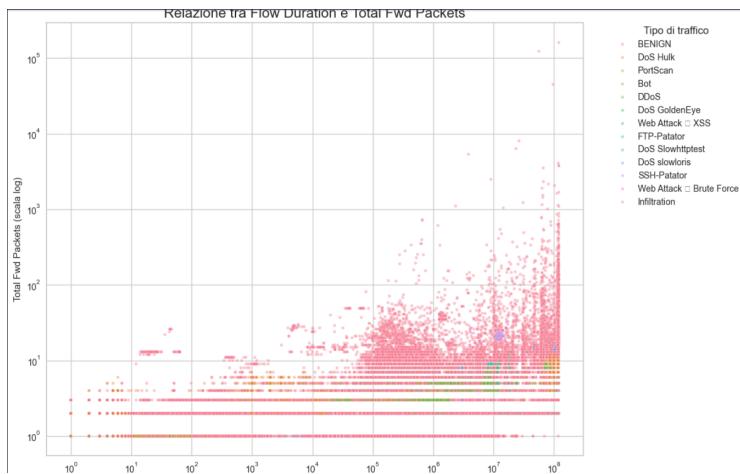


Figura 1.9: Relazione tra Flow Duration e Total Fwd Packets

Dal grafico emerge una forte concentrazione del traffico BENIGN su un'area ampia del piano, con flussi caratterizzati da durate elevate e da un numero variabile di pacchetti, riflettendo la naturale eterogeneità delle comunicazioni lecite. Al contrario, molte classi di attacco risultano distribuite in regioni più circoscritte, spesso associate a un numero limitato di pacchetti e a durate più brevi o ripetitive, coerentemente con comportamenti automatici e non interattivi.

La visualizzazione evidenzia inoltre una parziale separazione tra le classi, suggerendo che la combinazione di Flow Duration e Total Fwd Packets possa fornire un buon potere discriminante. Questo aspetto risulta particolarmente rilevante in ottica di clustering non supervisionato, dove la presenza di strutture aggregative nel piano delle feature facilita l'individuazione di gruppi omogenei, e di classificazione supervisionata, dove la relazione tra più variabili contribuisce a migliorare la capacità predittiva dei modelli.

Nel complesso, il grafico conferma l'importanza di considerare relazioni multivariate tra le feature di rete, oltre alle singole distribuzioni univariate, per una comprensione più completa dei pattern di traffico presenti nel dataset CIC-IDS-2017.

La fase di data visualization ha consentito di ottenere una visione complessiva e strutturata del dataset CIC-IDS-2017, evidenziandone le principali caratteristiche statistiche e comportamentali. In particolare, l'analisi grafica ha messo in luce il marcato sbilanciamento tra le classi, la forte eterogeneità del traffico benigno e la presenza di pattern distintivi nelle feature temporali e volumetriche associate alle diverse tipologie di attacco. Tali evidenze hanno fornito indicazioni fondamentali per la successiva fase di classificazione supervisionata, orientando le scelte relative alla selezione delle feature, alla gestione dello sbilanciamento delle classi e alla definizione delle strategie di modellazione, che verranno affrontate nel capitolo successivo.

Capitolo 2

Classificazione

La fase di classificazione supervisionata costituisce uno degli obiettivi principali dell’analisi, in quanto mira a distinguere automaticamente il traffico di rete lecito da quello malevolo sulla base delle feature estratte dai flussi. Nel contesto del dataset CIC-IDS-2017, tale approccio risulta particolarmente appropriato poiché ogni istanza è associata a una variabile target etichettata (Label), che identifica in modo esplicito la classe di appartenenza del flusso. Il dataset presenta un insieme ricco e strutturato di feature numeriche, comprendenti metriche temporali, volumetriche e di protocollo, che descrivono in maniera dettagliata il comportamento del traffico di rete. Questa caratteristica rende il CIC-IDS-2017 particolarmente adatto all’applicazione di algoritmi di classificazione basati su machine learning, in grado di apprendere pattern complessi e non lineari associati alle diverse tipologie di attacco. Inoltre, la presenza di classi multiple emersa durante la fase di EDA, rende la classificazione un banco di prova realistico per la valutazione delle prestazioni dei modelli in scenari reali. La classificazione supervisionata consente pertanto di valutare in modo quantitativo la capacità dei modelli di generalizzare su dati eterogenei e di identificare efficacemente comportamenti anomali all’interno del traffico di rete.

2.1 Pre-processing

Prima di procedere con l’addestramento dei modelli di classificazione, è stata effettuata una fase di pre-processing finalizzata a rendere il dataset coerente e adeguato all’analisi supervisionata. In particolare, come descritto nella fase di ETL, il dataset è stato sottoposto a operazioni di pulizia strutturale e sintattica, comprendenti la normalizzazione dei nomi delle colonne, la conversione delle feature in formato numerico, la gestione dei valori infiniti e la rimozione delle osservazioni contenenti valori NaN. Al termine di tali operazioni, il dataset risultava privo di anomalie strutturali e composto esclusivamente da feature numeriche consistenti. Successivamente, è stata effettuata la separazione tra variabili di input e variabile target, distinguendo le feature descrittive del traffico di rete dalla classe associata a ciascun flusso. In particolare, tutte le colonne esplicative sono state raccolte nella matrice delle feature (X), mentre la variabile Label è stata isolata come vettore target (y). Questa separazione ha permesso di analizzare in modo esplicito la distribuzione numerica delle classi, mediante il conteggio delle istanze appartenenti a ciascuna tipologia di traffico. Tale analisi preliminare si è resa necessaria per valutare la presenza e l’entità dello sbilanciamento tra classi, già evidenziato nella fase di EDA, e per determinare l’eventuale necessità di applicare tecniche di undersampling o oversampling nella fase successiva.

Label	
BENIGN	2271320
Dos Hulk	230124
PortScan	158804
DDoS	128025
Dos GoldenEye	10293
FTP-Patator	7935
SSH-Patator	5897
Dos slowloris	5796
Dos Slowhttptest	5499
Bot	1956
Web Attack ⚡ Brute Force	1507
Web Attack ⚡ XSS	652
Infiltration	36
Web Attack ⚡ Sql Injection	21
Heartbleed	11

Figura 2.1: Distribuzione delle classi

Come si evince dall’immagine precedente, è presente un marcato sbilanciamento tra le classi, con una netta predominanza del traffico BENIGN, che conta oltre due milioni di istanze, rispetto alle classi di attacco. Anche alcune tipologie di attacco rilevanti, come DoS Hulk, PortScan e DDoS, presentano una numerosità significativamente inferiore, mentre altre classi risultano estremamente rare, con un numero di osservazioni dell’ordine di poche decine o unità. In particolare, le classi Infiltration e Heartbleed presentano una numerosità estremamente ridotta, tale da renderle poco rappresentative ai fini dell’addestramento di modelli di classificazione supervisionata. La presenza di classi con così poche istanze può infatti compromettere la stabilità dei modelli, favorire fenomeni di overfitting e rendere non significative le metriche di valutazione. Per questo motivo, prima di applicare tecniche di riequilibrio del dataset, si è deciso di effettuare un raggruppamento delle classi più rare. In particolare:

- tutte le varianti di Web Attack sono state unificate in un’unica classe (Web_Attack);
- le classi Infiltration e Heartbleed sono state accorpate in una classe residuale (Other_Attack).

Questa scelta consente di ridurre la frammentazione della variabile target, mantenendo al contempo una distinzione semantica coerente tra le principali famiglie di attacco, e di aumentare la numerosità delle classi minoritarie, rendendole più adatte all'apprendimento supervisionato.

A seguito del raggruppamento delle classi rare, si è proceduto ad applicare una strategia di undersampling della classe BENIGN, riducendone in modo controllato la numerosità. In particolare, è stato fissato un numero massimo di istanze per la classe BENIGN pari a 300.000 flussi, selezionati in maniera casuale ma riproducibile tramite l'utilizzo di un random seed costante.

```
target_benign = 300_000
rng = 42

benign_mask = (data["Label"] == "BENIGN")
benign_df = data[benign_mask]
other_df = data[~benign_mask]

if len(benign_df) > target_benign:
    benign_down = benign_df.sample(n=target_benign, random_state=rng)
else:
    benign_down = benign_df

balanced_d = pd.concat([other_df, benign_down], axis=0) \
    .sample(frac=1, random_state=rng) \
    .reset_index(drop=True)

print("Distribuzione dopo undersampling BENIGN:")
print(balanced_d["Label"].value_counts())
```

Figura 2.2: Undersampling della classe BENIGN

Dopo l'operazione di undersampling della classe BENIGN, si è optato per un'operazione di oversampling delle altre classi di attacco. Al fine di procedere con tali tecniche, è stato innanzitutto necessario effettuare una codifica numerica della variabile target. Le etichette testuali associate alle classi di traffico sono state convertite in valori numerici mediante un'operazione di label encoding. Questa trasformazione consente di rappresentare la variabile target in una forma compatibile con gli algoritmi di machine learning e, in particolare, con le tecniche di oversampling basate su distanza, come SMOTE. Contestualmente, è stata costruita una mappatura esplicita tra le classi originali e i corrispondenti valori numerici, al fine di mantenere la tracciabilità

semantica delle etichette. Una volta completata la codifica, è stata definita una strategia di oversampling mirata, specificando per ciascuna classe minoritaria il numero desiderato di istanze finali. Tale scelta è stata effettuata in modo controllato, evitando un riequilibrio completo tra tutte le classi e privilegiando invece un compromesso tra bilanciamento del dataset e realismo dello scenario di traffico. In particolare, sono state incrementate le classi di attacco meno rappresentate, come Other_Attack, Web_Attack, Bot, DoS slowloris, DoS Slowhttptest, SSH-Patator e FTP-Patator, portandole a una numerosità tale da renderle statisticamente significative per l'addestramento dei modelli. L'oversampling è stato realizzato mediante l'algoritmo SMOTE, che genera nuove istanze sintetiche interpolando i campioni esistenti delle classi minoritarie nello spazio delle feature. È stato utilizzato un numero ridotto di vicini (k-neighbors = 3) per limitare la generazione di osservazioni artificiali troppo distanti dai dati originali e ridurre il rischio di introdurre rumore.

```
{'BENIGN': 0,
 'Bot': 1,
 'DDoS': 2,
 'DoS GoldenEye': 3,
 'DoS Hulk': 4,
 'DoS Slowhttptest': 5,
 'DoS slowloris': 6,
 'FTP-Patator': 7,
 'Other_Attack': 8,
 'PortScan': 9,
 'SSH-Patator': 10,
 'Web_Attack': 11}
```

Figura 2.3: Mappatura tra le classi originali e i corrispondenti valori numerici

```

sampling_strategy_enc = {
    label_mapping['Other_Attack']: 5000,
    label_mapping['Web_Attack']: 15000,
    label_mapping['Bot']: 10000,
    label_mapping['DoS slowloris']: 12000,
    label_mapping['DoS Slowhttptest']: 12000,
    label_mapping['SSH-Patator']: 15000,
    label_mapping['FTP-Patator']: 15000
}

smote = SMOTE(
    sampling_strategy=sampling_strategy_enc,
    random_state=42,
    k_neighbors=3
)

x_res, y_res_enc = smote.fit_resample(x, y_enc)
print("Distribuzione dopo SMOTE:")
print(pd.Series(y_res_enc).value_counts())

```

Figura 2.4: Overampling delle classi di attacco

```

Distribuzione dopo SMOTE:
0      300000
4     230124
9     158804
2     128025
7      15000
11     15000
10     15000
6      12000
5      12000
3      10293
1      10000
8       5000

```

Figura 2.5: Distribuzione finale delle classi

Come ultima operazione della fase di preprocessing, è stata calcolata la matrice di correlazione tra le feature numeriche del dataset. Tale analisi ha avuto lo scopo di individuare relazioni lineari significative tra le variabili e di evidenziare eventuali ridondanze informative all'interno dello spazio delle feature. La matrice di correlazione consente infatti di valutare il grado di dipendenza tra le variabili e fornisce un supporto alle scelte successive di modellazione, in particolare per quanto riguarda la selezione delle feature e l'applicazione di algoritmi sensibili alla multicollinearità. Questa analisi preliminare contribuisce a migliorare l'interpretabilità dei modelli e la stabilità del processo di classificazione.

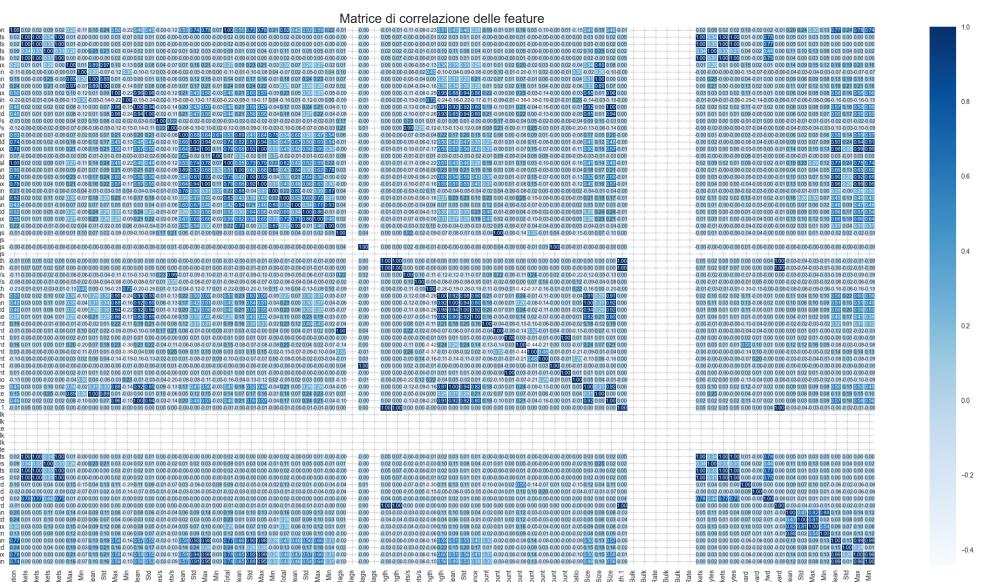


Figura 2.6: Matrice di correlazione

L’analisi della matrice di correlazione, calcolata sulle feature numeriche dopo le operazioni di pulizia, è stata utilizzata come strumento di riduzione della ridondanza informativa. In questa fase sono state innanzitutto eliminate le variabili a varianza nulla, in quanto prive di potere discriminante. Inoltre per ciascuna coppia di variabili caratterizzata da un coefficiente di correlazione superiore a 0,8 è stata rimossa una delle due feature, in quanto una forte correlazione tra attributi può indurre i classificatori a concentrarsi su informazioni ridondanti, riducendo il contributo di altre caratteristiche potenzialmente rilevanti e compromettendo la capacità di generalizzazione del modello. La matrice di correlazione finale mostra una struttura più compatta e bilanciata dello spazio delle feature, costituendo una base più solida per la successiva fase di addestramento e valutazione dei modelli di classificazione, come si evince dall’immagine seguente.

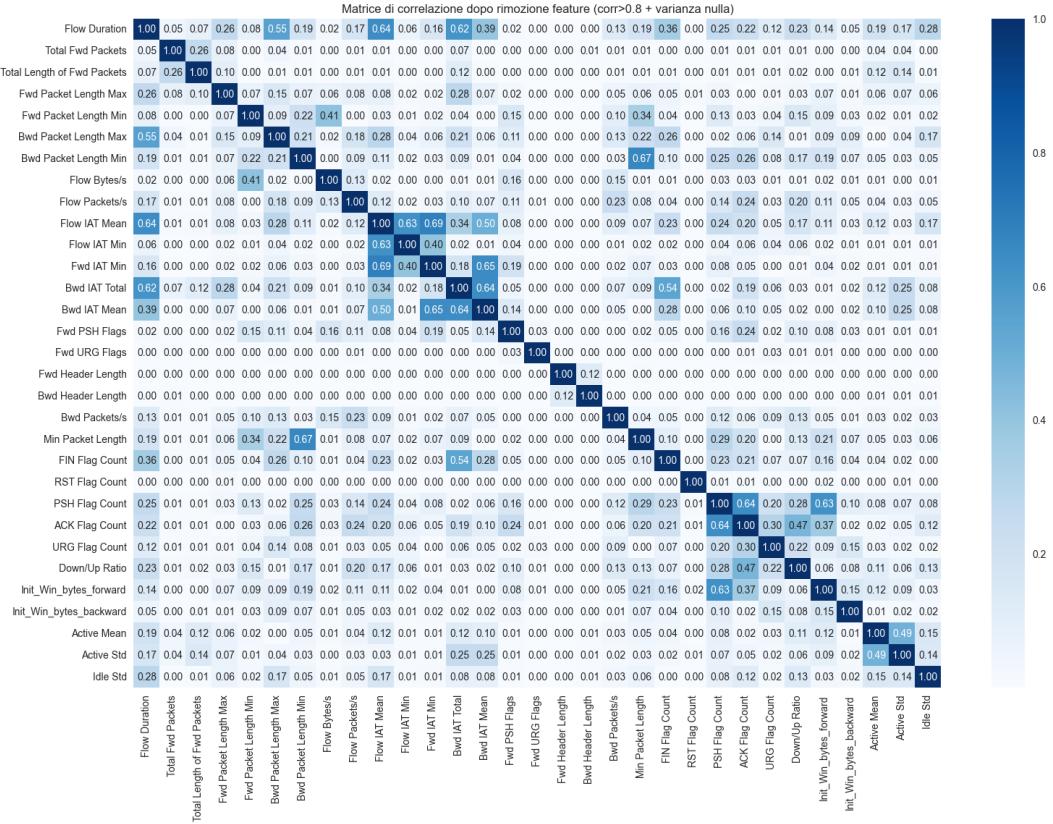


Figura 2.7: Matrice di correlazione finale

2.2 Training

In questa fase del progetto è stata effettuata la suddivisione del dataset in training set e test set mediante la funzione `train_test_split` della libreria `scikit-learn`. In particolare, l'80% delle osservazioni è stato assegnato al training set, mentre il restante 20% è stato riservato al test set. La suddivisione è stata eseguita in modalità stratificata rispetto alla variabile target, al fine di preservare la distribuzione delle classi anche nei due sottoinsiemi e garantire una valutazione più affidabile delle prestazioni dei modelli. Il processo ha prodotto due coppie di dataset distinti:

- x_{train} , y_{train} , utilizzati per l'addestramento dei modelli di classificazione;

- x_{test} , y_{test} , utilizzati esclusivamente per la valutazione delle performance su dati non visti durante il training.

Successivamente, sono stati definiti e confrontati diversi algoritmi di classificazione supervisionata, selezionati in modo da includere modelli lineari, modelli basati su alberi e metodi di ensemble. In particolare, sono stati considerati i seguenti classificatori:

- **Logistic Regression;**
- **Decision Tree Classifier;**
- **Random Forest Classifier;**
- **AdaBoost Classifier;**
- **Gradient Boosting Classifier;**
- **Linear Discriminant Analysis;**
- **Support Vector Classifier (SVC).**

I modelli Logistic Regression, Linear Discriminant Analysis e Support Vector Classifier sono stati integrati all'interno di una pipeline comprendente una fase preliminare di standardizzazione delle feature tramite *StandardScaler*. Tale scelta è motivata dal fatto che questi algoritmi sono sensibili alla scala delle variabili di input, e la standardizzazione consente di evitare che differenze di ordine di grandezza tra le feature influenzino negativamente il processo di apprendimento. Al contrario, i modelli basati su alberi decisionali e metodi ensemble (Decision Tree, Random Forest, AdaBoost e Gradient Boosting) non richiedono una fase di scaling delle feature, poiché operano tramite suddivisioni basate su soglie e risultano intrinsecamente invarianti rispetto alla scala delle variabili. Pertanto, tali modelli sono stati utilizzati direttamente senza standardizzazione.

2.3 Analisi dei risultati

Le prestazioni dei modelli di classificazione sono state valutate sul test set mediante le metriche di Accuracy e F1-macro. In particolare, l'uso della F1-macro risulta fondamentale in questo contesto, poiché consente di attribuire uguale importanza a tutte le classi, indipendentemente dalla loro numerosità, rendendo la valutazione più affidabile in presenza di dataset precedentemente sbilanciati.

Modello	Accuracy	F1-Score
Logistic Regression	0.9263	0.7816
Decision Tree	0.9983	0.9953
Random Forest	0.9987	0.9966
AdaBoost	0.5300	0.2507
Gradient Boosting	0.9954	0.9818
LDA	0.7471	0.5505
SVC	0.9597	0.9142

Tabella 2.1: Risultati dei modelli

Come si evince dalla tabella precedente, i risultati ottenuti mostrano differenze significative tra i modelli considerati. I classificatori basati su alberi decisionali e metodi di ensemble hanno fornito le prestazioni migliori. In particolare, *DecisionTreeClassifier* e *RandomForestClassifier* hanno raggiunto valori di Accuracy prossimi a 0,999 e valori di *F1-macro* superiori a 0,995, indicando un'elevata capacità di discriminare correttamente sia il traffico benigno sia le diverse tipologie di attacco. Anche *GradientBoostingClassifier* ha mostrato performance elevate, sebbene leggermente inferiori rispetto ai due modelli precedenti.

Al contrario, i modelli lineari e a margine massimo hanno evidenziato prestazioni più contenute. *Logistic Regression* e *Linear Discriminant Analysis*

presentano valori di *F1-macro* sensibilmente inferiori, suggerendo una minore capacità di catturare le relazioni non lineari presenti nelle feature del dataset. *SVC* ottiene risultati complessivamente buoni, ma inferiori rispetto ai modelli ensemble, probabilmente a causa dell'elevata complessità e dimensionalità dello spazio delle feature. *AdaBoostClassifier* mostra invece prestazioni nettamente inferiori, indicando una difficoltà nel gestire efficacemente la struttura del dataset e la complessità del problema multi-classe.

Sulla base delle prestazioni ottenute sul test set, si è deciso di applicare una cross-validation stratificata solo ai modelli che hanno mostrato i risultati più promettenti, ovvero *DecisionTreeClassifier*, *RandomForestClassifier* e *GradientBoostingClassifier*. Questa scelta è stata motivata da considerazioni di costo computazionale, evitando di applicare la cross-validation a modelli con prestazioni nettamente inferiori.

La cross-validation è stata eseguita con una strategia *Stratified K-Fold* a 3 fold, utilizzando come metrica di valutazione la *F1-macro*, ritenuta la più rappresentativa per il problema in esame.

Modello	F1-Score	Dev. Standard
Decision Tree	0.994266	0.000099
Random Forest	0.996545	0.0000107
Gradient Boosting	0.981126	0.015385

Tabella 2.2: Risultati della cross-validation

I risultati ottenuti mostrano una notevole stabilità delle prestazioni per *DecisionTreeClassifier* e *RandomForestClassifier*, entrambi caratterizzati da una deviazione standard estremamente ridotta, indice di una buona capacità di generalizzazione e di una bassa variabilità delle performance sui diversi sottoset di training.

In particolare, *RandomForestClassifier* risulta il modello più performante e stabile, confermando quanto emerso nella valutazione sul test set. *Gradient-*

BoostingClassifier, pur mantenendo un valore medio di *F1-macro* elevato, presenta una maggiore variabilità, suggerendo una sensibilità più marcata alla composizione dei dati di training.

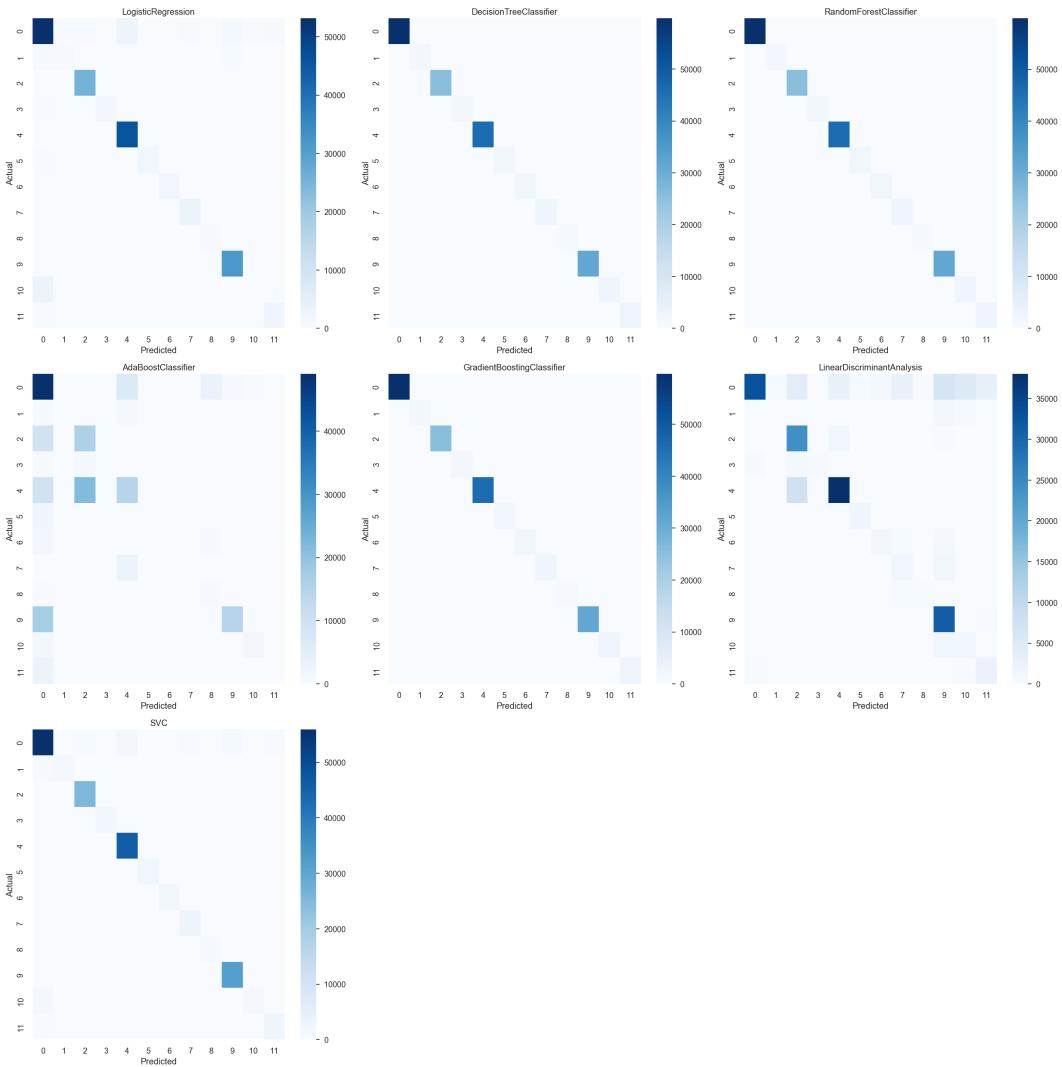


Figura 2.8: Matrici di confusione dei modelli

Le matrici di confusione presenti nell’immagine precedente, consentono di analizzare in modo dettagliato il comportamento dei classificatori sulle singole classi, evidenziando errori di classificazione e pattern di confusione residui. Dai grafici emerge una chiara differenziazione tra i modelli con prestazioni elevate e quelli meno efficaci. I modelli *DecisionTreeClassifier*, *RandomForestClas-*

sifier e *GradientBoostingClassifier* mostrano matrici fortemente concentrate lungo la diagonale principale, indicando un'elevata capacità di classificare correttamente la maggior parte delle istanze per tutte le classi. In particolare, *RandomForestClassifier* presenta un livello di errore estremamente contenuto anche sulle classi minoritarie, confermando la robustezza osservata nelle metriche aggregate e nella cross-validation. Al contrario, *LogisticRegression* e *LinearDiscriminantAnalysis* evidenziano una maggiore dispersione dei valori fuori dalla diagonale, segnalando difficoltà nel distinguere alcune classi di attacco tra loro. Questo comportamento è coerente con la natura prevalentemente lineare di tali modelli, meno adatta a catturare le relazioni complesse presenti nel dataset. Infine *AdaBoostClassifier* mostra la matrice di confusione meno strutturata, con numerosi errori di classificazione e una marcata confusione tra classi, a conferma delle prestazioni nettamente inferiori rilevate tramite *Accuracy* e *F1-macro*. *SVC* si colloca in una posizione intermedia, con buone prestazioni sulle classi principali ma una maggiore incertezza su alcune categorie minoritarie.

Successivamente all'analisi delle metriche aggregate e delle matrici di confusione, le prestazioni dei modelli migliori sono state ulteriormente valutate mediante la **curva ROC (Receiver Operating Characteristic)** in un contesto multi-classe, adottando l'approccio One-vs-Rest e considerando sia la media macro sia la media micro dell'Area Under the Curve (AUC). Dal grafico emerge che entrambi i modelli analizzati, *DecisionTreeClassifier* e *RandomForestClassifier*, presentano curve ROC fortemente prossime all'angolo superiore sinistro del piano, indicando un'elevata capacità discriminante. In particolare, *RandomForestClassifier* raggiunge valori di AUC pari a 1.000 sia in media macro sia in media micro, mentre *DecisionTreeClassifier* mostra valori leggermente inferiori ma comunque prossimi all'unità. Questi risultati confermano l'elevata efficacia dei modelli nel distinguere correttamente tra le diverse classi di traffico, anche considerando l'intero spettro dei possibili trade-off tra true

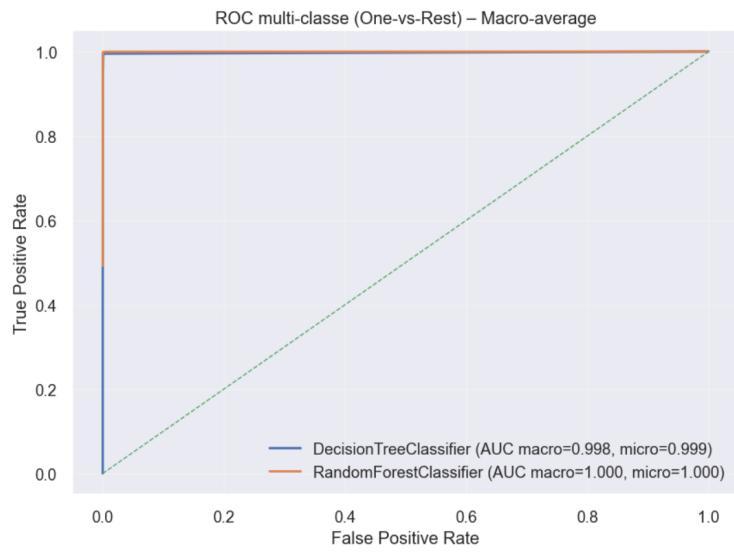


Figura 2.9: Curva ROC

positive rate e false positive rate.

Tuttavia è opportuno osservare che le elevate prestazioni ottenute dai modelli, in particolare dai classificatori basati su alberi decisionali ed ensemble, sono verosimilmente influenzate anche dalla strategia di riequilibrio del dataset adottata in fase di preprocessing. L'applicazione combinata di undersampling della classe BENIGN e oversampling delle classi minoritarie tramite SMOTE ha contribuito a fornire ai modelli un numero maggiore di esempi rappresentativi delle classi di attacco, facilitando l'apprendimento di pattern discriminanti. In particolare, l'oversampling può aver ridotto il bias verso la classe maggioritaria e migliorato la recall delle classi minoritarie, con un impatto positivo sulla metrica F1-macro. Pertanto, i risultati ottenuti devono essere interpretati tenendo conto del contributo del preprocessing, e non esclusivamente della capacità intrinseca dei modelli, sottolineando l'importanza di una valutazione critica delle prestazioni in scenari realistici di Intrusion Detection.

2.4 Greed Search

Un elemento determinante per il miglioramento delle prestazioni dei modelli di machine learning è la corretta configurazione degli iperparametri, ossia quelle variabili esterne al processo di apprendimento che devono essere definite prima dell’addestramento. Una scelta appropriata degli iperparametri può incidere in modo significativo sull’accuratezza, sulla capacità di generalizzazione e sulla robustezza complessiva del modello.

A tal fine, è stata adottata la tecnica della **Grid Search**, un approccio sistematico che esplora un insieme predefinito di combinazioni di iperparametri al fine di individuare la configurazione ottimale rispetto a una metrica di valutazione. Per questo lavoro la selezione è stata guidata dalla *F1-macro media* ottenuta tramite validazione incrociata, ritenuta la metrica più rappresentativa per il problema multi-classe considerato.

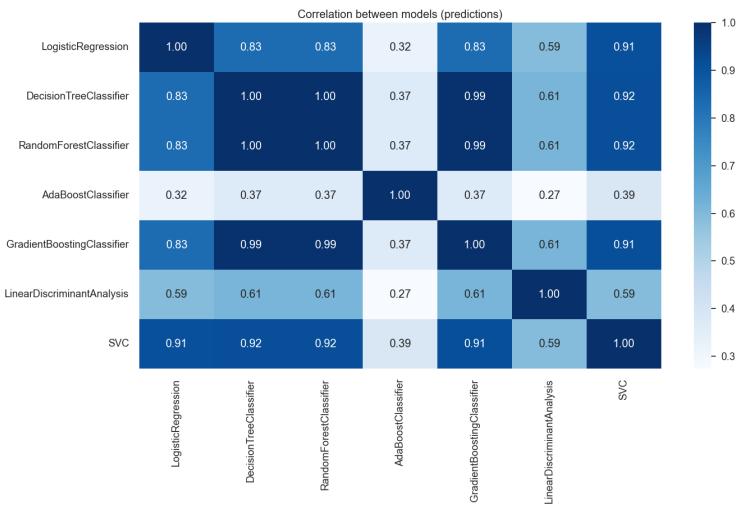


Figura 2.10: Matrice di correlazione tra i modelli

Prima di procedere con la Grid Search, è stata analizzata la matrice di correlazione tra le predizioni dei classificatori, al fine di valutare il grado di similarità nel comportamento dei diversi modelli. Tale analisi ha evidenziato una forte correlazione tra i modelli basati su alberi decisionali, in particolare

DecisionTreeClassifier, *RandomForestClassifier* e *GradientBoostingClassifier*, a conferma del fatto che essi apprendono pattern simili dal dataset. Sebbene l’analisi della matrice di correlazione tra le predizioni abbia evidenziato una correlazione molto elevata (0,99) tra *RandomForestClassifier* e *GradientBoostingClassifier*, si è comunque deciso di includere entrambi i modelli nella fase di ottimizzazione degli iperparametri. Questa scelta è motivata dal fatto che la correlazione tra le predizioni, pur indicando un comportamento simile sul dataset, non implica equivalenza dei modelli dal punto di vista algoritmico, dato che il primo utilizza un approccio di bagging con alberi indipendenti, mentre il secondo adotta una strategia di boosting sequenziale, in cui ciascun modello corregge gli errori del precedente. Inoltre, entrambi i modelli hanno ottenuto le migliori prestazioni assolute nella fase di valutazione baseline, in termini di *F1-macro*, stabilità e capacità di generalizzazione. Pertanto, l’elevata correlazione è stata considerata un indicatore di efficacia comune sul dataset, ma non un criterio di esclusione, in quanto l’obiettivo della Grid Search era quello di individuare il miglior compromesso prestazionale tra i modelli più promettenti. Gli altri modelli come i modelli lineari (*Logistic Regression* e *Linear Discriminant Analysis*), sono stati invece esclusi dato che avevano mostrato prestazioni inferiori, così come il *Support Vector Classifier (SVC)*, che, nonostante buoni risultati, avrebbe comportato costi computazionali eccessivi in fase di Grid Search.

```
# GRID SEARCH RANDOM FOREST
RF_param = {
    "n_estimators": [100, 200],
    "max_depth": [None, 20, 40],
    "min_samples_split": [2, 5],
    "min_samples_leaf": [1, 2],
    "max_features": ["sqrt", "log2"]
}
# GRID SEARCH GRADIENT BOOSTING (subset stratificato)
GB_param = {
    "n_estimators": [100, 200],
    "learning_rate": [0.05, 0.1],
    "max_depth": [3, 5],
    "subsample": [0.8, 1.0]
}
```

Figura 2.11: Parametri dei modelli per la Greed Search

```

==== CONFRONTO BASELINE vs GRID SEARCH (F1_macro) ====
Baseline RF F1_macro: 0.9965
Tuned RF F1_macro: 0.9966

Baseline GB F1_macro: 0.9811
Tuned GB F1_macro (subset CV): 0.9913

```

Figura 2.12: Risultati Greed Search

Per *RandomForestClassifier*, la Grid Search è stata eseguita sull'intero training set, individuando come configurazione ottimale un numero elevato di alberi e una selezione casuale delle feature ($max_features = \sqrt{r}$). Il valore di *F1-macro* in *cross-validation* ottenuto risulta pari a 0.9966, sostanzialmente in linea con il modello baseline.

Per *GradientBoostingClassifier*, al fine di contenere i tempi computazionali, la Grid Search è stata effettuata su un sottoinsieme stratificato del training set. L'ottimizzazione ha portato a un incremento significativo della *F1-macro* in *cross-validation*, che passa da 0.9811 (baseline) a 0.9913, evidenziando un beneficio concreto del tuning degli iperparametri per questo modello.

Dopo la selezione degli iperparametri ottimali, i modelli sono stati rifittati sull'intero training set e valutati sul test set. I risultati mostrano prestazioni molto elevate per entrambi i modelli:

- **RandomForestClassifier** mantiene prestazioni eccellenti, con una *F1-macro* sostanzialmente invariata rispetto al baseline;
- **GradientBoostingClassifier** raggiunge prestazioni comparabili a quelle di Random Forest, riducendo significativamente il gap osservato nella fase iniziale.

Modello	Accuracy	F1-Macro
Random Forest	0.998694	0.996616
Gradient Boosting	0.998656	0.996060

Tabella 2.3: Risultati finali della Greed Search

2.5 Model Ensembling

2.5.1 Random Forest + Gradient Boosting

Dopo aver ottimizzato singolarmente i modelli tramite Grid Search, è stata implementata una strategia di **model ensembling** combinando i classificatori *Random Forest* e *Gradient Boosting* mediante un *VotingClassifier* con *voting soft*. Tale approccio consente di aggregare le probabilità predette dai singoli modelli, sfruttandone i punti di forza e migliorando la robustezza complessiva del sistema di classificazione. L'uso di un ensemble è motivato dal principio secondo cui la combinazione di più modelli può contribuire a ridurre la varianza e a mitigare errori specifici dei singoli classificatori. Il modello ensemble è stato addestrato sul training set e valutato sul test set, ottenendo i seguenti risultati:

- **Accuracy** pari a 0.9988
- **F1-macro** pari a 0.9967

Questi valori risultano in linea con quelli ottenuti dai migliori modelli singoli ottimizzati, senza introdurre un miglioramento sostanziale in termini di performance numerica. Tuttavia, l'ensemble fornisce un vantaggio in termini di robustezza decisionale, poiché combina le predizioni di due modelli ad alte prestazioni, riducendo la dipendenza da un singolo classificatore.

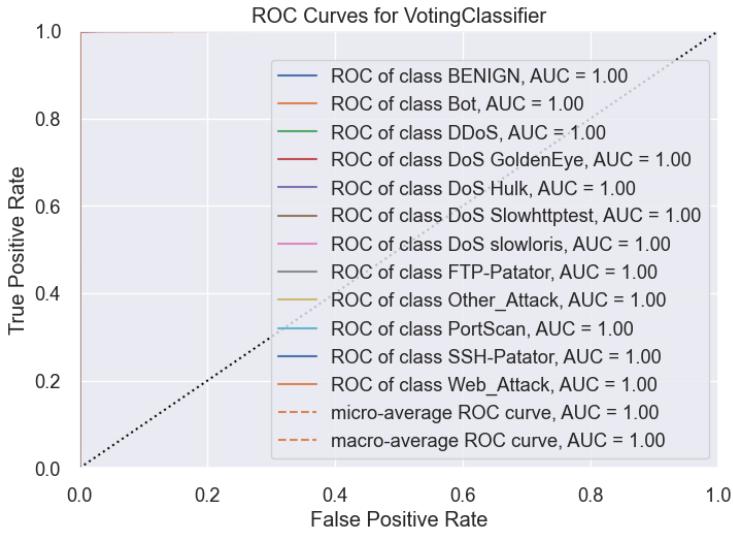


Figura 2.13: Curva ROC

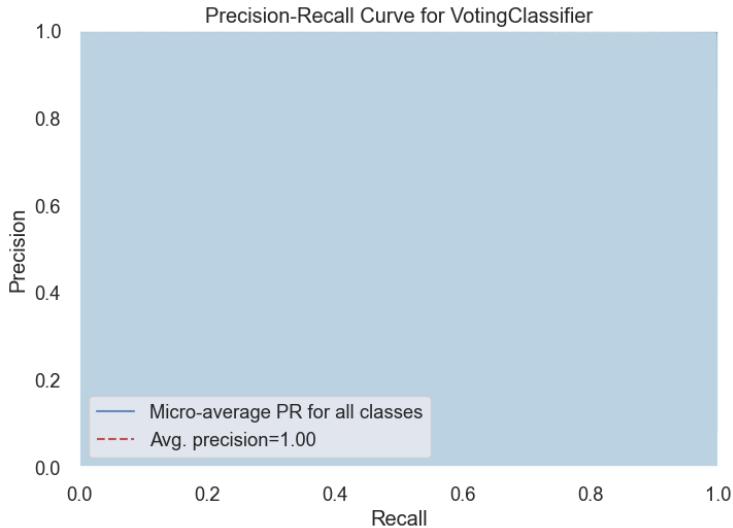


Figura 2.14: Curva Precision Recall

La valutazione del modello ensemble è stata ulteriormente approfondita tramite l'analisi delle **curve ROC** multi-classe (One-vs-Rest) e delle curve **Precision-Recall**. Le curve ROC mostrano valori di AUC pari a 1.00 per tutte le classi, nonché per le medie macro e micro, indicando una capacità discriminante estremamente elevata del modello ensemble su tutte le tipologie di traffico considerate. Analogamente, la curva Precision-Recall evidenzia un'average precision pari a 1.00, suggerendo un eccellente bilanciamento tra preciso-

ne e richiamo anche sulle classi minoritarie. Tali risultati confermano quanto osservato nelle metriche aggregate, pur dovendo essere interpretati alla luce delle operazioni di riequilibrio del dataset effettuate in fase di preprocessing.

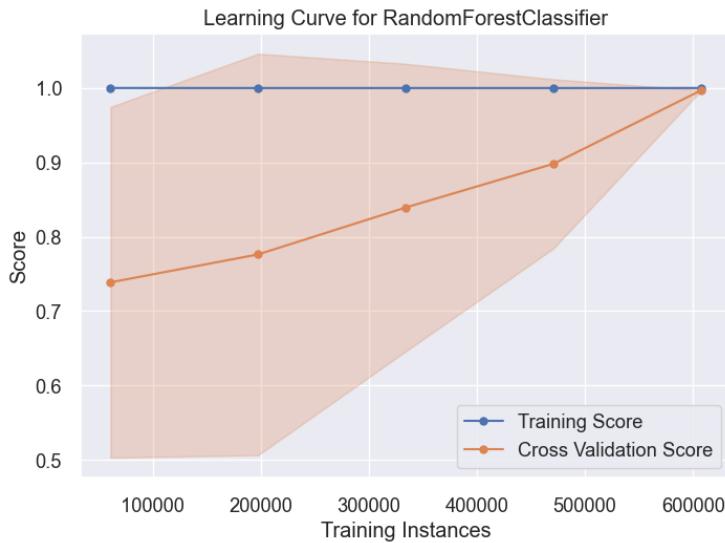


Figura 2.15: Learning Curve

Come ulteriore analisi, è stata costruita la Learning Curve, al fine di valutare il comportamento del modello al variare della dimensione del training set. Tuttavia tale analisi non è stata effettuata sul Voting Classifier, poiché il costo computazionale associato all’addestramento ripetuto dell’ensemble avrebbe comportato tempi di esecuzione eccessivi. La Learning Curve è stata quindi calcolata esclusivamente sul Random Forest, in quanto rappresenta il modello singolo con le migliori prestazioni complessive emerse dalle precedenti analisi. Questa scelta consente comunque di ottenere indicazioni affidabili sulla capacità di generalizzazione del sistema, considerando che il Random Forest costituisce il principale contributore alle prestazioni dell’ensemble. Dall’analisi della Learning Curve invece, viene mostrato l’andamento delle prestazioni al variare del numero di istanze di training. Il grafico evidenzia un gap contenuto e stabile tra training score e cross-validation score, indicando che il modello non soffre né di underfitting né di overfitting. Inoltre, la curva di cross-validation mostra una tendenza crescente all’aumentare dei dati di training, suggerendo

che l'aggiunta di ulteriori dati potrebbe portare a un ulteriore, seppur marginale, miglioramento delle prestazioni. Questo comportamento è indicativo di una buona capacità di generalizzazione del modello ensemble. Nel complesso, il model ensembling tra Random Forest e Gradient Boosting non ha prodotto un incremento significativo delle prestazioni rispetto al miglior modello singolo già ottimizzato, ma ha confermato l'elevata affidabilità del sistema di classificazione. L'ensemble risulta pertanto particolarmente vantaggioso in ottica di stabilità e robustezza, piuttosto che di puro incremento prestazionale, e rappresenta una valida soluzione per contesti applicativi reali di Intrusion Detection.

2.5.2 Random Forest + Logistic Regression

In continuità con l'approccio adottato in precedenza, in cui è stato costruito un ensemble tra Random Forest e Gradient Boosting, è stata successivamente implementata una seconda strategia di model ensembling combinando **Random Forest** e **Logistic Regression** mediante un VotingClassifier con voting soft. Questa scelta è motivata da finalità comparative, poiché la Logistic Regression presenta un meccanismo decisionale sostanzialmente diverso rispetto ai modelli ad albero già considerati e, di conseguenza, una correlazione più bassa nelle predizioni rispetto alla coppia Random Forest–Gradient Boosting. L'obiettivo di questo secondo ensemble è quindi valutare se l'integrazione di un modello lineare e regolarizzato con un classificatore non lineare ad alta capacità possa apportare benefici in termini di robustezza e stabilità delle decisioni, anche in assenza di un incremento significativo delle metriche aggregate. Il modello ensemble è stato addestrato sul training set e valutato sul test set, ottenendo i seguenti risultati:

- **Accuracy** pari a 0.9976
- **F1-macro** pari a 0.9924

Tali valori risultano molto elevati e coerenti con le prestazioni dei migliori modelli singoli precedentemente ottimizzati. Anche in questo caso, l'ensembling non introduce un incremento sostanziale delle metriche aggregate, ma garantisce una maggiore stabilità decisionale, rendendo il sistema meno dipendente dal comportamento di un singolo classificatore.

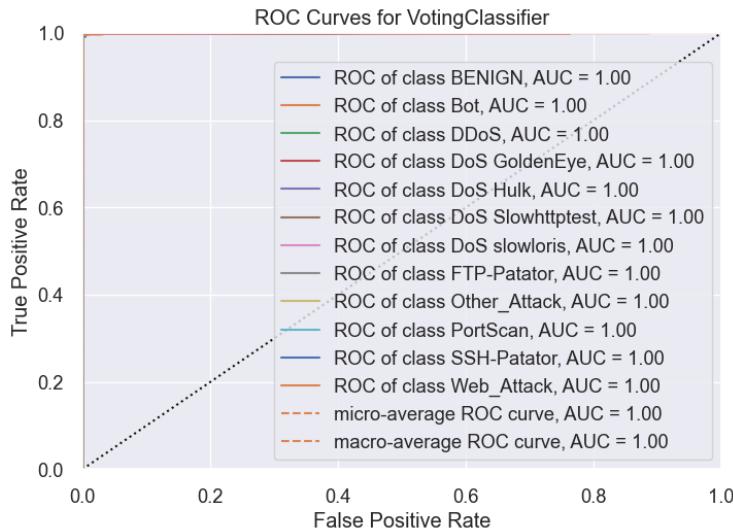


Figura 2.16: Curva ROC

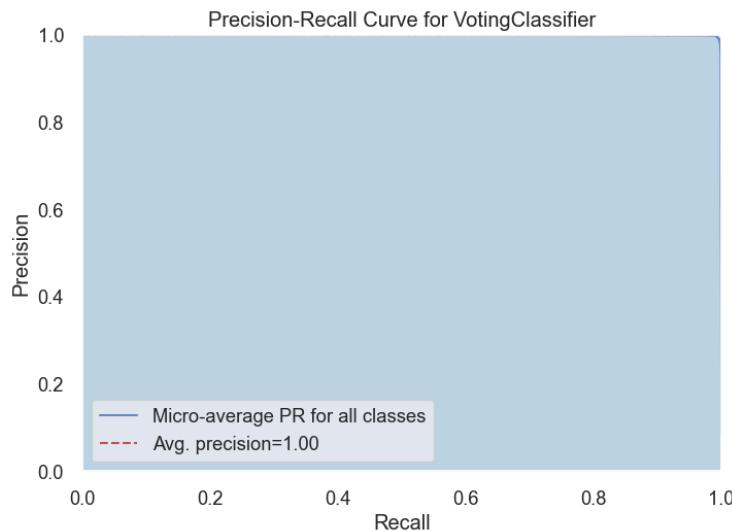


Figura 2.17: Curva Precision Recall

Anche in questo caso la valutazione del modello ensemble è stata ulterior-

mente approfondita tramite l’analisi delle curve ROC multi-classe (One-vs-Rest) e delle curve Precision-Recall. Le curve ROC mostrano valori di AUC pari a 1.00 per tutte le classi, nonché per le medie macro e micro, indicando una capacità discriminante estremamente elevata del modello ensemble su tutte le tipologie di traffico considerate.

Analogamente, la curva Precision-Recall evidenzia un’average precision pari a 1.00, suggerendo un eccellente compromesso tra precisione e richiamo anche in presenza di classi minoritarie. Questi risultati confermano l’efficacia dell’approccio ensemble, pur dovendo essere interpretati alla luce delle tecniche di riequilibrio del dataset applicate in fase di preprocessing.

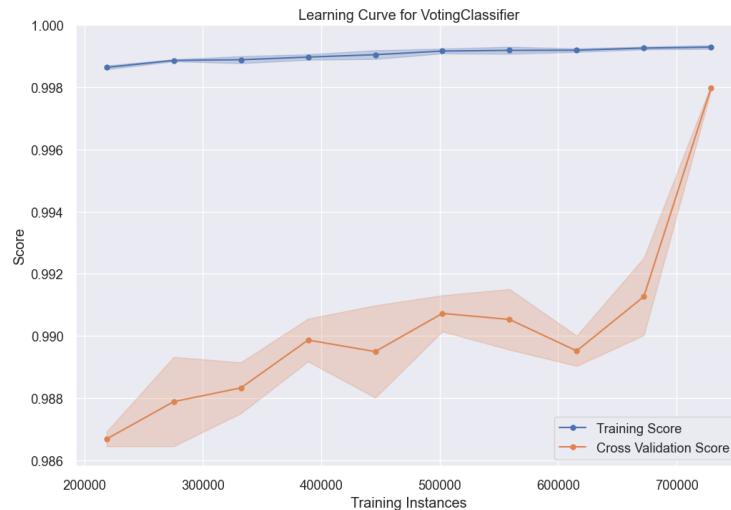


Figura 2.18: Learning Curve

L’analisi della Learning Curve evidenzia un gap contenuto e stabile tra il training score e il cross-validation score, indicando l’assenza di fenomeni significativi di overfitting o underfitting. La curva di validazione mostra inoltre una tendenza crescente all’aumentare del numero di istanze di training, suggerendo che il modello beneficia dell’aggiunta di nuovi dati e presenta una buona capacità di generalizzazione. Nel complesso, l’ensembling tra Random Forest e Logistic Regression conferma l’elevata affidabilità del sistema di classificazione, senza introdurre miglioramenti numerici marcati rispetto al miglior

modello singolo. Tuttavia, l’ensemble risulta particolarmente vantaggioso in termini di robustezza, stabilità e bilanciamento delle decisioni, rappresentando una soluzione solida e affidabile per applicazioni reali di Intrusion Detection in scenari multi-classe complessi. In conclusione, dopo l’analisi dell’ensemble Random Forest + Gradient Boosting, l’ensemble Random Forest + Logistic Regression è stato introdotto come termine di confronto, sfruttando la minore correlazione tra i due modelli. Il confronto evidenzia che, sebbene l’eterogeneità contribuisca alla diversificazione delle decisioni, l’ensemble RF + GB garantisce prestazioni complessivamente superiori, risultando la scelta più efficace nel contesto considerato.

Capitolo 3

Clustering

Accanto alla classificazione supervisionata, il progetto ha previsto una fase di **Clustering non supervisionato** con l'obiettivo di esplorare la struttura interna del dataset e individuare gruppi di flussi di rete caratterizzati da comportamenti simili, senza fare uso delle etichette di classe. Questo tipo di analisi consente di evidenziare pattern latenti nei dati e di valutare in che misura le informazioni contenute nelle feature siano in grado di separare naturalmente il traffico benigno da quello potenzialmente malevolo. Nel contesto del dataset CIC-IDS-2017, il clustering rappresenta uno strumento complementare alla classificazione, utile sia per l'analisi esplorativa sia per l'identificazione di possibili anomalie non note a priori. L'approccio non supervisionato permette infatti di osservare il comportamento dei dati in uno scenario più vicino a contesti reali, in cui le etichette non sono sempre disponibili. In questa fase sono state adottate tre tecniche principali:

- **K-Means**, utilizzato per individuare cluster compatti basati sulla minimizzazione della distanza intra-cluster;
- **PCA (Principal Component Analysis)**, impiegata come tecnica di riduzione della dimensionalità per semplificare lo spazio delle feature, migliorare la visualizzazione dei dati e supportare il clustering;

- **DBSCAN**, utilizzato per individuare cluster di forma arbitraria e per identificare flussi anomali o rumorosi, sfruttando un approccio basato sulla densità.

L'utilizzo combinato di queste tecniche consente di confrontare metodologie con ipotesi differenti e di ottenere una visione più articolata della distribuzione dei dati, evidenziando sia strutture compatte sia comportamenti isolati o anomali all'interno del traffico di rete.

3.1 Pre-processing

Il pre-processing adottato per la fase di clustering ricalca in larga parte quello utilizzato per la classificazione supervisionata, al fine di garantire coerenza metodologica tra le due analisi. In particolare, il dataset è stato sottoposto alle stesse operazioni di pulizia, selezione delle feature e rimozione delle variabili ridondanti e delle variabili a varianza 0. Successivamente, tutte le feature sono state sottoposte a standardizzazione tramite *StandardScaler*, in modo da ottenere variabili con media zero e varianza unitaria. Lo scaling risulta un passaggio essenziale nel contesto del clustering, poiché algoritmi come *K-Means* e *DBSCAN* si basano sul calcolo di distanze tra punti nello spazio delle feature e sono pertanto sensibili alle differenze di scala tra le variabili. Al termine di queste operazioni, si ottiene una matrice delle feature scalata e priva di variabili costanti, idonea all'applicazione delle tecniche di clustering e di riduzione della dimensionalità.

3.2 K-Means

Come primo passo, è stato applicato l'**Elbow Method** sui dati scalati, con l'obiettivo di individuare un numero appropriato di cluster (k). Tale metodo analizza l'andamento della *Within-Cluster Sum of Squares* (inertia) al variare

di k , consentendo di identificare un punto oltre il quale l'aumento del numero di cluster produce benefici marginali in termini di riduzione della varianza intra-cluster.

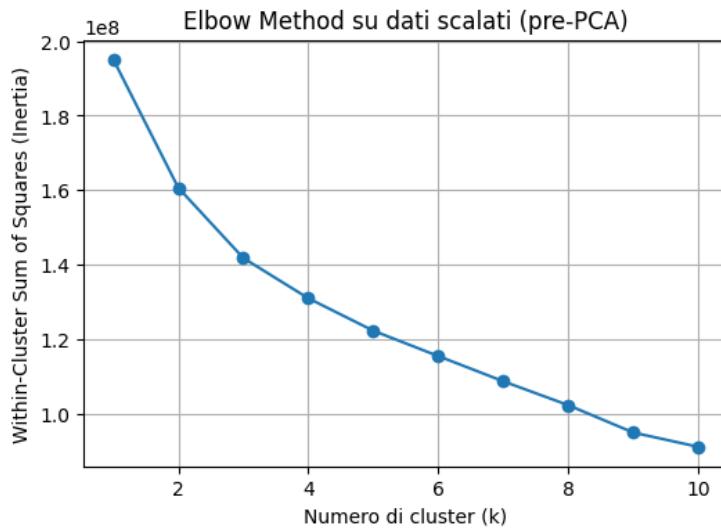


Figura 3.1: Elbow Methodod

Dal grafico si osserva una riduzione marcata dell'inertia passando da $k = 1$ a $k = 2$, mentre per valori successivi il decremento diventa progressivamente più graduale. In particolare, il punto di flesso non risulta nettamente definito, collocandosi in prossimità di $k = 2$. Tuttavia, l'incremento del numero di cluster a $k = 3$ consente di ottenere una maggiore capacità descrittiva, permettendo di distinguere ulteriori sottostrutture nel traffico di rete che non emergono chiaramente con una suddivisione binaria. Una volta definito il numero di cluster, è stato applicato l'algoritmo *K-Means* sulla matrice delle feature scalate. L'algoritmo assegna ciascuna osservazione al cluster il cui centroide risulta più vicino, secondo una metrica di distanza euclidea, producendo una partizione dei dati in gruppi compatti e mutuamente esclusivi. Come è stato detto in precedenza, l'utilizzo di dati standardizzati risulta fondamentale in questa fase, poiché K-Means è fortemente sensibile alla scala delle variabili e basa il processo di clustering esclusivamente sulle distanze nello spazio delle feature. Poiché il clustering è stato eseguito su uno spazio delle feature ad alta

dimensionalità, non è possibile rappresentare direttamente i risultati in modo intuitivo. Per questo motivo, al fine di interpretare visivamente la struttura dei cluster individuati da K-Means, è stato necessario ricorrere a proiezioni bidimensionali, selezionando coppie di feature rappresentative del traffico di rete. La scelta di utilizzare due sole feature per volta è motivata da esigenze di leggibilità e interpretabilità: una rappresentazione bidimensionale consente infatti di osservare in modo diretto la separazione tra i cluster e di analizzare eventuali sovrapposizioni, cosa che non sarebbe possibile in spazi di dimensione superiore senza ricorrere a tecniche di riduzione della dimensionalità. Le feature selezionate per la visualizzazione non sono state scelte in modo arbitrario, ma in base alla loro rilevanza semantica e operativa nel contesto del traffico di rete:

- la proiezione **Flow Duration vs Flow Bytes/s** consente di analizzare simultaneamente la durata temporale dei flussi e la quantità di dati trasferiti, due caratteristiche fondamentali per distinguere tra traffico normale, traffico intensivo e comportamenti anomali. In particolare, si osservano gruppi di flussi caratterizzati da durata ridotta e basso volume di dati, tipici di comunicazioni brevi o di richieste isolate, e cluster associati a flussi più persistenti o ad alta intensità, che possono indicare trasferimenti massivi di dati o attività anomale prolungate. Questa separazione suggerisce che la combinazione di durata e intensità del traffico costituisce un criterio informativo rilevante per distinguere comportamenti eterogenei.
- la rappresentazione **Total Fwd Packets vs Total Backward Packets** permette invece di valutare il volume e la direzionalità del traffico, evidenziando differenze tra flussi bilanciati e flussi caratterizzati da comunicazioni prevalentemente unidirezionali, tipiche di alcune tipologie di attacco. In questo spazio, i cluster individuati da K-Means risultano differenziati in base al grado di simmetria del traffico: alcuni clu-

ster comprendono flussi con un numero bilanciato di pacchetti forward e backward, tipici di comunicazioni bidirezionali regolari, mentre altri sono caratterizzati da una marcata predominanza di pacchetti in una sola direzione. Quest'ultimo comportamento è spesso indicativo di interazioni anomale, come richieste ripetute senza risposte proporzionate o trasmissioni unidirezionali di dati.

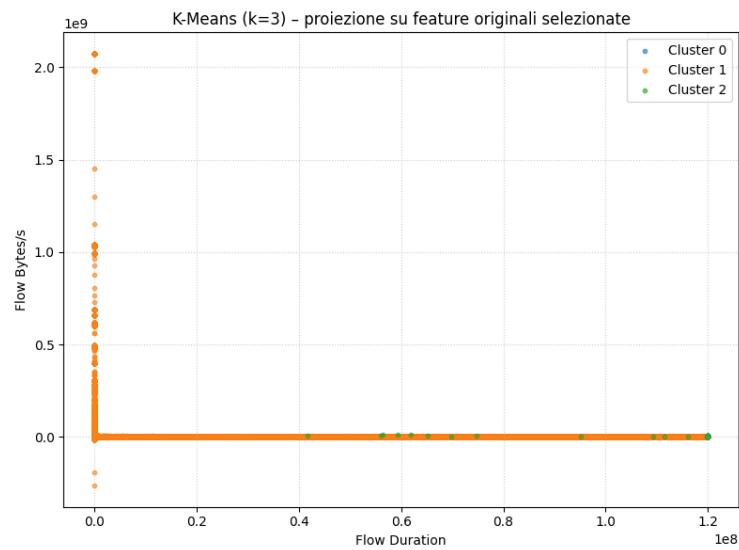


Figura 3.2: Flow Duration vs Flow Bytes/s

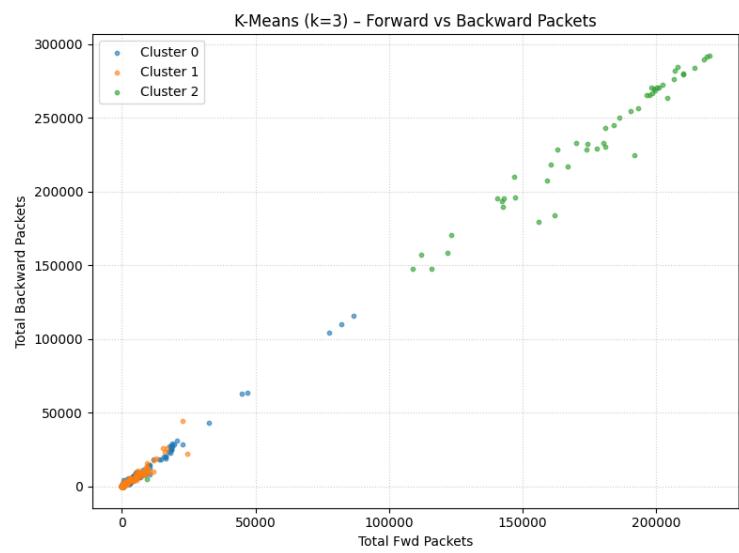


Figura 3.3: Total Fwd Packets vs Total Backward Packets

A completamento dell'analisi, è stata effettuata una **Silhouette Analysis**, utilizzata per valutare la qualità della partizione ottenuta. Il coefficiente di silhouette misura quanto ciascun punto sia ben assegnato al proprio cluster rispetto agli altri, assumendo valori compresi tra -1 e 1.

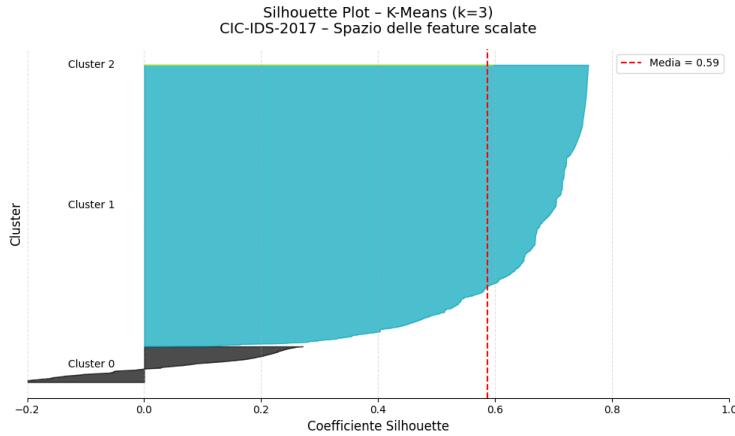


Figura 3.4: Silhouette Plot

Il grafico della silhouette mostra che il valore medio della silhouette risulta pari a 0,59, indicando una buona separazione complessiva tra i cluster e una coerenza interna soddisfacente. Tale valore suggerisce che la maggior parte delle osservazioni è assegnata correttamente al proprio cluster e che i gruppi individuati presentano una struttura ben definita, pur non essendo completamente separati. Dal grafico emerge inoltre che la distribuzione dei coefficienti di silhouette è prevalentemente positiva, con un numero limitato di osservazioni prossime allo zero e senza la presenza significativa di valori negativi. Questo indica l'assenza di assegnazioni fortemente errate e conferma che i cluster individuati risultano globalmente stabili.

3.2.1 Clustering Gerarchico

Il **clustering gerarchico** è una tecnica di apprendimento non supervisionato che costruisce una struttura gerarchica di raggruppamento dei dati, organizzando le osservazioni in una sequenza di cluster annidati. Nel caso del clu-

stering gerarchico agglomerativo, adottato in questo lavoro, ogni osservazione viene inizialmente considerata come un cluster singolo e, a ogni iterazione, i cluster più simili vengono progressivamente fusi secondo un criterio di distanza e di collegamento (linkage). Questo approccio risulta particolarmente utile per l'analisi esplorativa dei dati, poiché permette di evidenziare la struttura interna del dataset senza imporre ipotesi rigide sulla forma dei cluster. L'algoritmo di clustering gerarchico agglomerativo presenta una complessità computazionale elevata, che lo rende difficilmente applicabile all'intero dataset CIC-IDS-2017. Per questo motivo, prima dell'applicazione dell'algoritmo è stato effettuato un campionamento casuale stratificato di dimensione limitata, pari a 20.000 osservazioni, mantenendo la distribuzione generale dei dati e garantendo la riproducibilità tramite l'impostazione di un random seed. Il campione così ottenuto è stato utilizzato esclusivamente per l'analisi non supervisionata, consentendo di ridurre significativamente i tempi computazionali senza compromettere la rappresentatività delle strutture principali presenti nel dataset. Il clustering è stato eseguito utilizzando l'algoritmo **Agglomerative Clustering** con linkage di tipo *Ward* e distanza euclidea. Il metodo di *Ward* è progettato per minimizzare la varianza intra-cluster ad ogni fase di fusione, favorendo la creazione di cluster compatti e omogenei. Coerentemente con quanto emerso nella fase di K-Means, il numero di cluster è stato fissato a $k = 3$, al fine di rendere confrontabili i risultati tra i diversi approcci di clustering.

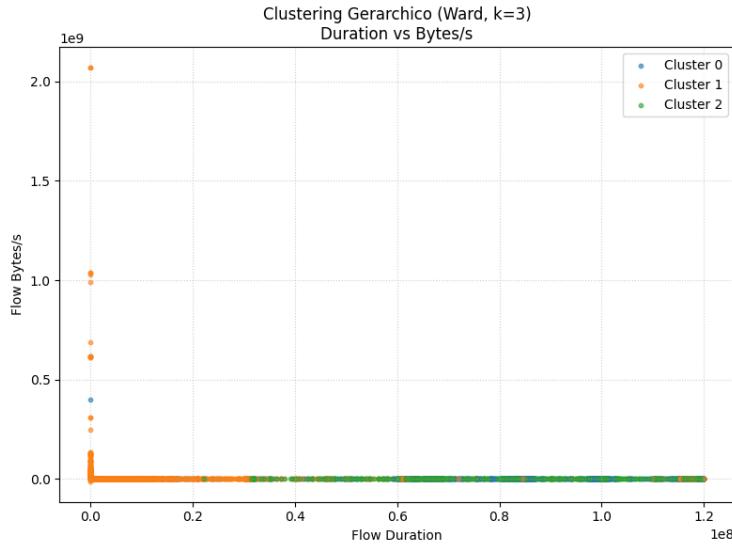


Figura 3.5: Flow Duration vs Flow Bytes/s Clustering Gerarchico

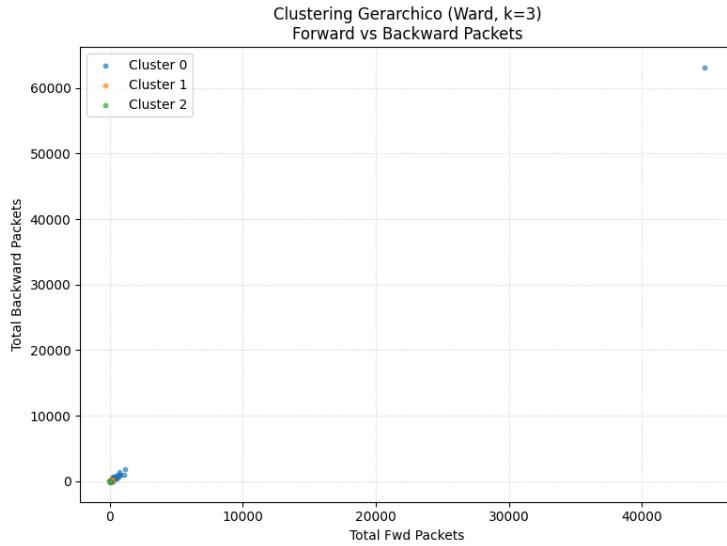


Figura 3.6: Total Fwd Packets vs Total Backward Packets Clustering Gerarchico

Le visualizzazioni bidimensionali ottenute mostrano una struttura dei cluster in larga parte coerente con quanto osservato tramite K-Means:

- nella proiezione **Flow Duration vs Flow Bytes/s**, i cluster risultano separati principalmente in base all'intensità del traffico e alla durata dei

flussi, distinguendo flussi brevi e poco intensivi da flussi più persistenti o ad alto volume;

- nella rappresentazione **Total Fwd Packets vs Total Backward Packets**, emerge una separazione basata sul volume complessivo e sulla direzionalità del traffico, con cluster associati a comunicazioni più bilanciate e cluster caratterizzati da una maggiore asimmetria.

Tali risultati suggeriscono che il clustering gerarchico è in grado di individuare pattern strutturali simili a quelli rilevati da K-Means, pur adottando una logica di aggregazione differente.

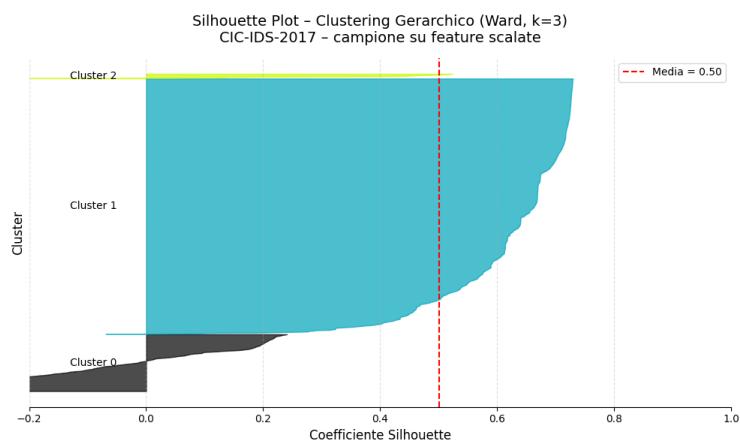


Figura 3.7: Silhouette Plot Clustering Gerarchico

La qualità del clustering gerarchico è stata valutata mediante **Silhouette Analysis**. Il valore medio del coefficiente di silhouette risulta pari a 0,50, indicando una separazione moderata tra i cluster e una coesione interna complessivamente accettabile.

Dal grafico emerge che la maggior parte delle osservazioni presenta valori di silhouette positivi, mentre una porzione limitata di punti mostra valori prossimi allo zero o lievemente negativi. Questo comportamento suggerisce la presenza di aree di sovrapposizione tra i cluster, coerente con la natura eterogenea del traffico di rete e con l'assenza di confini netti tra alcuni comportamenti.

Nel complesso, il valore medio di silhouette pari a 0,50 indica una qualità del clustering inferiore rispetto a quella ottenuta con K-Means (silhouette media pari a 0,59), ma comunque adeguata per un'analisi esplorativa in un contesto non supervisionato.

3.3 PCA

Considerata l'elevata dimensionalità dello spazio delle feature, è stata applicata la **Principal Component Analysis (PCA)** con l'obiettivo di ridurre la dimensionalità dei dati, mantenendo al contempo la maggior parte dell'informazione originale. La PCA consente di trasformare le feature originali in un nuovo insieme di variabili ortogonali (componenti principali), ordinate in base alla varianza spiegata.

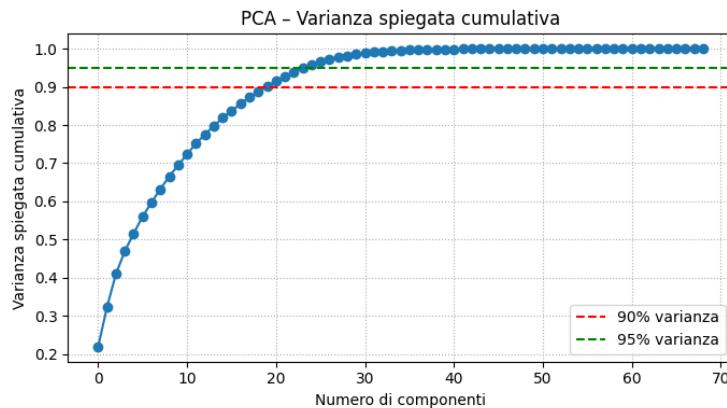


Figura 3.8: Varianza spiegata cumulativa

In questo caso sono state selezionate 20 componenti principali, numero ritenuto adeguato per preservare una quota significativa della varianza totale, riducendo al contempo la complessità dello spazio dei dati.

Una volta ottenuta la rappresentazione dei dati nello spazio PCA, è stato nuovamente applicato l'*Elbow Method* al fine di verificare se la riduzione della dimensionalità influisse sulla scelta del numero ottimale di cluster. Il grafico dell'inertie mostra un andamento analogo a quello osservato nello spazio ori-

ginale, con un punto di flesso non nettamente definito ma collocato tra $k = 2$ e $k = 3$. Anche in questo caso, si è scelto di fissare $k = 3$, in continuità con le analisi precedenti e per mantenere coerenza nel confronto tra i diversi approcci di clustering.

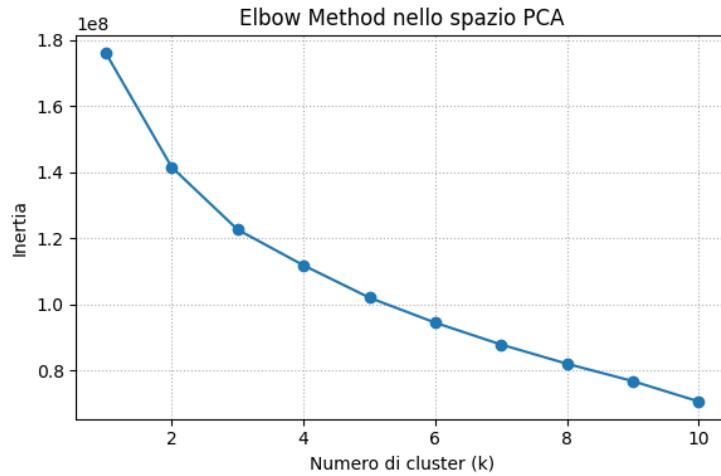


Figura 3.9: Elbow Method PCA

Successivamente, l'algoritmo *K-Means* è stato applicato ai dati trasformati tramite PCA.

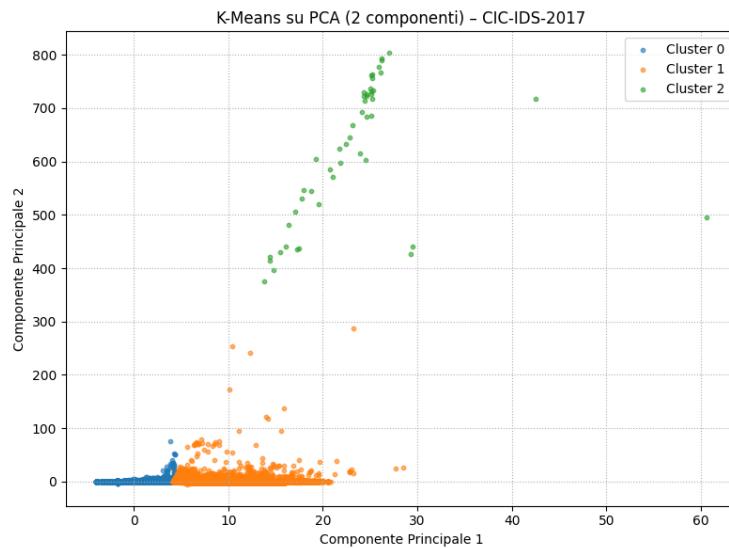


Figura 3.10: k-Means su PCA (2 componenti)

La visualizzazione dei cluster sulle prime due componenti principali è stata effettuata esclusivamente a scopo di visualizzazione. Tale scelta è necessaria per consentire una rappresentazione grafica bidimensionale dei cluster, ma non riflette l'intero spazio informativo utilizzato dall'algoritmo di clustering. Le due componenti rappresentano quindi una proiezione parziale dei dati, utile per l'interpretazione visiva ma non esaustiva della struttura del dataset. La visualizzazione mostra comunque una separazione più evidente rispetto a quella osservabile nello spazio delle feature originali. Nello spazio bidimensionale delle prime due componenti, alcune strutture latenti del dataset risultano più facilmente osservabili rispetto allo spazio originale, in cui l'elevata dimensionalità e la correlazione tra le feature rendono complessa la lettura diretta dei dati. Tuttavia, questa maggiore leggibilità visiva non implica una separazione completa dei cluster, ma riflette una proiezione parziale della struttura appresa nello spazio PCA a dimensione superiore.

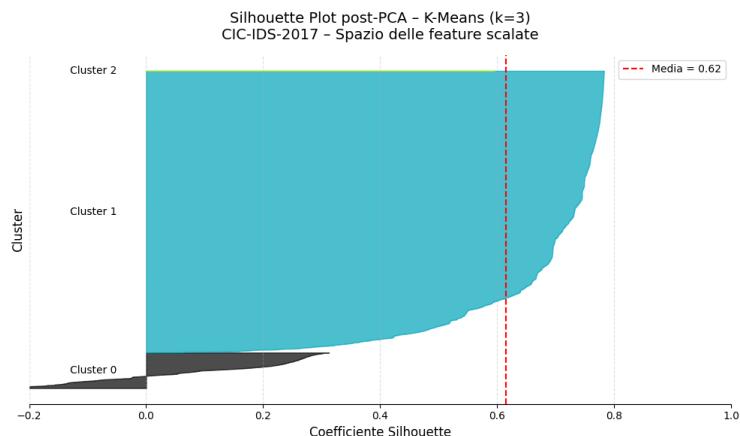


Figura 3.11: Silhouette Plot PCA

Anche in questo caso, la qualità del clustering è stata valutata mediante Silhouette Analysis. Il valore medio del coefficiente di silhouette ottenuto risulta pari a 0,62, superiore a quello registrato nello spazio delle feature originali (0,59). Questo incremento indica una migliore separazione tra i cluster e una maggiore coesione interna, suggerendo che la riduzione della dimensionalità ha

avuto un effetto positivo sulla struttura del clustering.

3.4 DBSCAN

Per completare l'analisi di clustering è stato applicato l'algoritmo **DBSCAN** (**Density-Based Spatial Clustering of Applications with Noise**), una tecnica basata sulla densità che consente di individuare automaticamente gruppi di osservazioni caratterizzati da elevata concentrazione di punti, distinguendoli dal rumore. A differenza di K-Means e del clustering gerarchico, DBSCAN non richiede di fissare a priori il numero di cluster, risultando particolarmente adatto a dataset complessi e sbilanciati come CIC-IDS-2017. L'algoritmo è stato applicato nello spazio delle componenti principali (PCA), al fine di ridurre la dimensionalità, mitigare l'effetto della correlazione tra feature e rendere più stabile il calcolo delle distanze euclidean. La scelta dei parametri fondamentali, ε (*eps*) e *min_samples*, è stata guidata tramite l'analisi del k-distance graph, costruito calcolando la distanza dall'8° vicino più prossimo per ciascun punto. Il valore di *min_samples* = 8 è stato selezionato come compromesso tra sensibilità ai cluster densi e robustezza al rumore, mentre il valore di ε è stato individuato osservando il punto di flesso della curva.

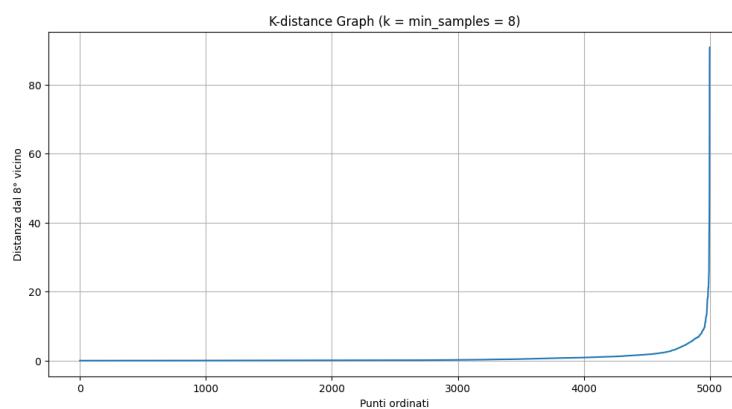


Figura 3.12: K-distance Graph

Con $\varepsilon = 2.5$, DBSCAN ha individuato 41 cluster, con una percentuale di

rumore pari a circa 1.46%, indicando una buona capacità dell'algoritmo di assegnare la maggior parte delle osservazioni a regioni dense dello spazio delle feature. Per valutare la sensibilità del metodo rispetto al parametro ε , sono stati eseguiti ulteriori esperimenti:

- con $\varepsilon = 2.8$, il numero di cluster si riduce a 36, con una leggera diminuzione del rumore;
- con $\varepsilon = 2.2$, il numero di cluster aumenta a 47, accompagnato da un incremento della percentuale di rumore.

Questi risultati confermano il comportamento atteso di DBSCAN: valori di ε più piccoli producono una frammentazione maggiore, mentre valori più elevati tendono ad accorpare cluster distinti. La percentuale di rumore rimane comunque contenuta in tutti gli scenari analizzati, suggerendo che la struttura latente dei dati è caratterizzata da numerose regioni dense ben definite.

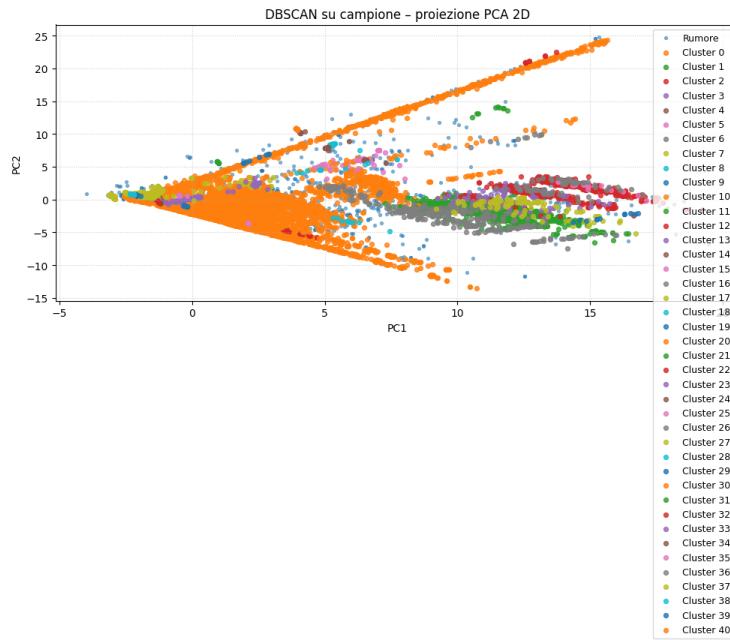


Figura 3.13: DBSCAN su 2 componenti

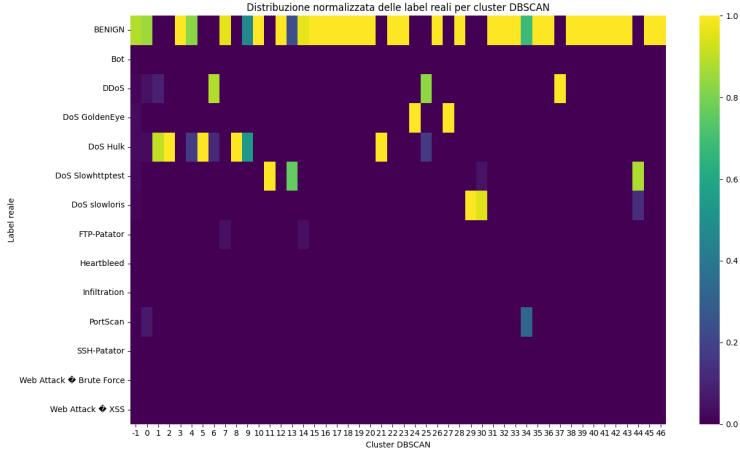


Figura 3.14: Distribuzione delle label reali per cluster

Dall’analisi del grafico precedente emerge che alcune classi presentano una forte concentrazione in un numero limitato di cluster, suggerendo pattern di traffico relativamente omogenei. In particolare, il traffico BENIGN risulta suddiviso in più cluster distinti, ciascuno caratterizzato da densità elevate, a indicare la presenza di sottogruppi strutturali all’interno del traffico lecito (ad esempio flussi brevi ad alta intensità, flussi lunghi a bassa intensità, ecc.).

Le principali classi DoS (come DoS Hulk, DoS slowloris e DoS Slowhttptest) mostrano una distribuzione più concentrata, con cluster specifici in cui la quota normalizzata è prossima all’unità. Questo indica che tali attacchi generano pattern di traffico ben definiti, facilmente separabili nello spazio delle feature utilizzato.

Al contrario, alcune classi meno frequenti o semanticamente più eterogenee risultano frammentate su più cluster o presentano valori di concentrazione più bassi. Ciò suggerisce che tali attacchi possono manifestarsi attraverso modalità operative differenti, che DBSCAN tende a separare in cluster distinti in base alla densità locale.

Capitolo 4

Dataset UGR'16

4.1 Contesto di riferimento

4.1.1 L'importanza del Network Traffic Forecasting

Nell'era digitale odierna, le infrastrutture di rete rappresentano la spina dorsale di servizi critici, dal cloud computing allo streaming multimediale. Per gli Internet Service Provider (ISP) e gli amministratori di rete, la capacità di monitorare e, soprattutto, *prevedere* l'evoluzione del traffico non è più un'opzione, ma un requisito operativo stringente.

Il problema del *Network Traffic Forecasting* (NTF) si inserisce in un contesto di gestione proattiva delle risorse. A differenza degli approcci reattivi, che intervengono solo al verificarsi di una congestione o di un guasto, un approccio predittivo abilita diverse funzionalità critiche:

1. **Cybersecurity e Rilevamento Anomalie:** Il traffico di rete segue pattern comportamentali umani (cicli sonno-veglia, orari lavorativi). Una deviazione significativa e improvvisa rispetto alla previsione del modello (es. un picco di traffico alle 3:00 del mattino) è spesso il primo indicatore di un'anomalia di sicurezza, come un attacco DDoS (Distributed Denial of Service) o un'esfiltrazione di dati, oppure di un guasto tecnico.

2. **Pianificazione della Capacità (Capacity Planning):** Le reti moderne devono rispettare rigorosi Service Level Agreements (SLA). Prevedere i picchi di carico permette di dimensionare correttamente l'infrastruttura fisica e virtuale, evitando sia il sovra-dimensionamento (costoso e inefficiente) che il sotto-dimensionamento (che causa perdita di pacchetti e latenza).
3. **Gestione Dinamica delle Risorse (Auto-scaling):** In ambienti virtualizzati e SDN (Software Defined Networks), la larghezza di banda può essere allocata dinamicamente. Un modello predittivo accurato funge da oracolo per i controller SDN, permettendo di scalare le risorse pochi minuti prima che il traffico reale aumenti.

4.1.2 Natura delle Serie Temporali nel Traffico Dati

Dal punto di vista statistico, il traffico internet è una serie temporale complessa. Essa esibisce tipicamente:

- **Non-linearietà e Non-stazionarietà:** Il volume di traffico cambia nel tempo a causa dell'aumento naturale dell'adozione di internet (trend di lungo periodo).
- **Stagionalità Multiple:** Esistono cicli sovrapposti: giornalieri (attività diurna vs notturna), settimanali (feriali vs weekend) e persino annuali.
- **Effetto "Heavy Tail":** La distribuzione del traffico presenta spesso code pesanti, con eventi rari di grandissima intensità.

In questo contesto, il presente lavoro si propone di affrontare un problema di regressione e forecasting su serie temporali, utilizzando dati reali di traffico di rete. L'obiettivo principale è la previsione del numero di flussi di rete su base temporale, adottando modelli statistici classici per serie temporali, in

particolare ARIMA, SARIMA e SARIMAX, e valutandone le prestazioni su un set di test temporale separato.

4.2 Il dataset

Il dataset selezionato per questo studio è l'**UGR'16** (University of Granada 2016 Dataset). Si tratta di uno dei dataset di riferimento nel campo della ricerca sulla sicurezza delle reti e dell'analisi del traffico. A differenza di dataset più datati (come *KDD'99*) che erano fortemente simulati, UGR'16 è stato costruito catturando traffico reale da un ISP spagnolo di livello 3 (Tier-3).

Questo conferisce ai dati un valore aggiunto significativo: il "rumore di fondo" e i pattern di traffico non sono generati da algoritmi sintetici, ma riflettono il comportamento reale di utenti umani e sistemi automatizzati, includendo tutte le imperfezioni e le complessità del mondo reale.

I dati non sono forniti come cattura completa dei pacchetti (PCAP), ma in formato **NetFlow/IPFIX**, infatti la versione del dataset utilizzata è quella denominata **UGR'16 Feature Data**, basata proprio su *flussi* e disponibile al seguente link: https://github.com/josecamachop/UGR16_FeatureData/tree/main

Un *flusso* (flow) è definito come una sequenza di pacchetti che condividono le stesse proprietà chiave (la cosiddetta 5-tupla: IP Sorgente, IP Destinazione, Porta Sorgente, Porta Destinazione, Protocollo) in una finestra temporale.

Questo è necessario perchè nel contesto del traffico di rete, i dati grezzi (**raw data**) sono tipicamente costituiti da:

- pacchetti individuali (packet-level),
- flussi elementari (flow-level),
- informazioni di basso livello come indirizzi IP, porte, protocolli, dimensioni dei pacchetti e timestamp ad alta risoluzione.

Tali dati, pur essendo estremamente dettagliati, risultano:

- molto voluminosi,
- difficili da modellare direttamente come serie temporali,
- più adatti ad approcci di tipo packet inspection o machine learning supervisionato.

La versione UGR'16 Feature Data, invece, fornisce statistiche aggregate su finestre temporali prefissate di **1 minuto**, trasformando il traffico grezzo in un insieme di variabili numeriche riassuntive. Questo processo di feature extraction è stato effettuato a monte dagli autori del dataset e consente di:

- ridurre drasticamente la dimensionalità,
- preservare le informazioni rilevanti sul volume e l'intensità del traffico,
- rendere i dati direttamente utilizzabili per analisi temporali e modelli di forecasting.

Di conseguenza, il dataset Feature Data rappresenta un compromesso efficace tra fedeltà informativa e trattabilità statistica, risultando particolarmente adatto a modelli di regressione e serie temporali.

4.3 Struttura del dataset

Prima di iniziare a lavorare con il dataset, trattandosi di una raccolta di dimensioni elevate, si è scelto di utilizzare il file `UGR16v4.Xtrain.csv` che è un sottoinsieme della raccolta totale dei dati e analizza un periodo di tempo che va dalle ore 00:00:00 del 19-03-2016 alle 23:59:00 del 31-05-2016.

A questo punto si è passati al caricamento e alla prima esplorazione del dataset con il seguente comando:

```

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Configurazione stile grafici
plt.style.use('seaborn-v0_8-whitegrid')
# Aumenta la dimensione dei font per leggibilità
sns.set_context("talk")

file_path = "../datasets/UGR16v4.Xtrain.csv"
traffic_raw = pd.read_csv(file_path)

traffic_raw.info()
print(traffic_raw.head())
traffic_raw.describe()

```

Figura 4.1: Comandi per il caricamento del dataset e la visualizzazione delle prime informazioni

Il risultato dei comandi `info` e `describe` è il seguente:

```

RangeIndex: 72644 entries, 0 to 72643
Columns: 135 entries, Row to nbytesveryhigh
dtypes: int64(135)
memory usage: 74.8 MB

      Row srcipprivate srcippublic srcipdefault dstipprivate \
0  201603190000      3     38587      3066          0
1  201603190001      4     36836      2938          0
2  201603190002      3     37349      2981          0
3  201603190003      3     38515      3024          0
4  201603190004      4     37519      2929          0

      dstippublic dstipdefault sportzero sportmultiplex sportecho ... \
0        41656          0       781          0          0 ...
1        39778          0       664          0          0 ...
2        40333          0       751          0          0 ...
3        41542          0       627          0          0 ...
4        40452          0       641          0          0 ...

      npacketsverylow npacketslow npacketsmedium npacketshigh \
0           65985     16076       1093       208
1           63934     14120       1249       229
2           64281     14970       1159       220
3           65731     15837       1290       192
4           64282     15098       1329       175

      npacketsveryhigh nbytesverylow nbyteslow nbytesmedium nbyteshigh \
0             30       62572     11883      8419      329
1              24       60519     10747      7871      283
2              36       61194     11268      7819      278
3              34       62476     11973      8229      287

```

Figura 4.2: Risultati dei comandi `info()` e `describe()`

Il dataset è composto da 72.644 osservazioni e 135 colonne, tutte di tipo numerico intero (`int64`), per un utilizzo complessivo di memoria pari a circa 75 MB. Non sono presenti valori mancanti nelle feature fornite, poiché l'estrazione delle caratteristiche è stata effettuata a monte dagli autori del dataset. Questo conferma l'elevata dimensionalità del dataset, pertanto si procede alla visualizzazione del nome delle colonne al fine di individuare quale possa fare al caso nostro per definire la variabile target. Si lancia dunque il comando:

```
1 list(traffic_raw.columns)
```

Listing 4.1: Visualizzazione colonne

Pur essendo estremamente numerose, le feauture possono essere raggruppate in gruppi logici che ne facilitano la comprensione:

- **IP-based feautures:** Queste feature rappresentano il numero di flussi o pacchetti associati a indirizzi IP privati, pubblici o di default, **sorgenti e destinazione**, fornendo una caratterizzazione generale della provenienza e della destinazione del traffico di rete in ciascun intervallo temporale. In questo gruppo troviamo per esempio `srcipprivate`, `srcippublic`, `srcipdefault`, `dstipprivate`, `dstipppublic` e `dstipdefault`. In poche parole, rappresentano il numero di flussi (o connessioni) che, in quel minuto, hanno come IP sorgente o destinazione un IP privato, pubblico o nessuno dei due (indirizzi non validi, muticast, broadcast, ecc).
- **Port-based feature:** Queste sono le feauture più consistenti nel dataset ed è legata alle porte di rete sorgente (`sport*`) e destinazione `dport*`. Ciascuna variabile è associata a una porta o a un servizio noto, tra cui HTTP, HTTPS, DNS, FTP, SSH, SMTP e molti altri. Questo insieme di feature consente di catturare il comportamento applicativo del traffico di rete, distinguendo tra servizi legittimi, servizi amministrativi e potenziali fonti di traffico anomalo. L'elevato numero di variabili legate alle porte riflette l'obiettivo originale del dataset, orientato principalmente a compiti di classificazione del traffico e rilevamento di anomalie, più che al forecasting temporale. Anche in questo caso il dato è espresso in numero di flussi.
- **Protocol-based feautures:** Un ulteriore gruppo di variabili descrive la distribuzione del traffico sui principali protocolli di rete (`protocoltcp`, `protocoludp`, `protocolicmp`, `protocoligmp` e `protocolother`). Queste

feature forniscono una visione aggregata dell'utilizzo dei protocolli nel tempo, risultando utili per analisi di tipo descrittivo e diagnostico.

- **TCP Flags features:** Il dataset include inoltre variabili relative ai flag TCP, quali: `tcpflagsSYN`, `tcpflagsACK`, `tcpflagsRST`, `tcpflagsFIN`, ecc. Queste informazioni sono particolarmente rilevanti in contesti di intrusion detection, poiché permettono di individuare comportamenti anomali come SYN flood o tentativi di scansione. Tuttavia, nel contesto di questo lavoro, tali variabili non verranno utilizzate direttamente come input del modello di forecasting, in quanto l'obiettivo è la previsione del volume complessivo di traffico piuttosto che l'identificazione di specifici pattern di attacco.
- **Traffic Volume feautures:** Il gruppo di feature più rilevante per l'obiettivo di forecasting è costituito dalle variabili che descrivono il volume di traffico:

- `npacketsverylow`, `npacketslow`, `npacketsmedium`, `npacketshigh`,
`npacketsveryhigh`
- `nbytesverylow`, `nbyteslow`, `nbytesmedium`, `nbyteshigh`,
`nbytesveryhigh`

Queste variabili rappresentano la distribuzione del numero di pacchetti e dei byte trasferiti, suddivisi per classi di intensità (da `very low` a `very high`). Dunque Per ogni finestra temporale di 1 minuto, il sistema di monitoraggio osserva tutti i flussi di rete e per ciascun flusso misura numero di pacchetti e numero di byte trasferiti e assegna il flusso a una classe di volume. Le classi sono discrete e rappresentano range predefiniti di pacchetti o byte, quindi per esempio da flussi molto piccoli (pochi byte/pacchetti) a flussi più grandi (molti byte/pacchetti). A partire da tali feature è stata poi definita la variabile target utilizzata in questo

lavoro, rappresentativa del volume totale di traffico (flussi) per intervallo temporale. Le statistiche descrittive di queste variabili evidenziano una forte variabilità, un ampio range di valori e la presenza di picchi significativi, suggerendo l'esistenza di stagionalità marcata e comportamenti non stazionari, aspetti che hanno guidato le scelte successive in fase di esplorazione e modellazione della serie temporale.

Ovviamente, oltre a tutte queste variabili, come si vede dalla figura, è presente anche la feature `Row` che sta ad indicare proprio il `timestamp`, fondamentale dunque per il nostro obiettivo. Già dalla visualizzazione si vede che la distanza tra una riga e l'altra è appunto di 1 minuto, a conferma di quanto detto.

Prima di iniziare la fase di *pre-processing* vera e propria, si svolge una verifica per rassicurarsi che le colonne con prefisso `npacket*` e `nbytes*` siano realmente basate su un conteggio dei flussi e non rappresentino la vera somma dei bytes o dei pacchetti in transito sulla rete minuto dopo minuto. Per verificare questo si esegue il seguente codice:

```
packet_cols = [
    'npacketsverylow',
    'npacketslow',
    'npacketsmedium',
    'npacketshigh',
    'npacketsveryhigh'
]

byte_cols = [
    'nbytesverylow',
    'nbyteslow',
    'nbytesmedium',
    'nbyteshigh',
    'nbytesveryhigh'
]

traffic_raw['total_packets'] = traffic_raw[packet_cols].sum(axis=1)
traffic_raw['total_bytes'] = traffic_raw[byte_cols].sum(axis=1)

traffic_raw[['total_packets', 'total_bytes']].head(10)
```

Figura 4.3: Codice di verifica per colonne `nbytes*` e `npacket*`

Il codice consiste quindi nel raggruppare in `total_packets` la somma di tutte le colonne riguardanti i pacchetti, indipendentemente dalla classe di volume e in `total_bytes` l'analogo ma considerando le colonne riguardanti i byte. Ovviamente quello che ci si aspetta è che `total_bytes` e `total_packets` ab-

biano sempre, minuto per minuto, lo stesso valore, in quanto si riferiscono appunto al numero totale di flussi osservati al minuto che è ovviamente unico e non al numero assoluto di bytes e di pacchetti che sarebbe ovviamente diverso tra loro. Il risultato ci dà la conferma che cercavamo:

	total_packets	total_bytes
0	83312	83312
1	79556	79556
2	80666	80666
3	83084	83084
4	80904	80904
5	88812	88812
6	79316	79316
7	79108	79108
8	78468	78468
9	78334	78334

Figura 4.4: Esito della verifica

Dal confronto diretto tra le due grandezze emerge che `total_packets` e `total_bytes` assumono valori identici per ogni osservazione. Questo comportamento è coerente con la struttura del dataset: entrambe le famiglie di variabili rappresentano conteggi di flussi, non quantità fisiche di pacchetti o byte. Di conseguenza, la loro somma restituisce lo stesso numero totale di flussi osservati nell'intervallo temporale.

Per il proseguimento dell'analisi, una sola delle due variabili è quindi sufficiente a rappresentare il volume complessivo di traffico, non come quantità fisica trasmessa, bensì come intensità di attività della rete, espressa come numero di flussi. Tuttavia, questa rappresentazione rimane coerente, valida e

metodologicamente giustificata per diversi motivi:

- **Coerenza con la struttura del dataset:** Il dataset UGR'16 è progettato per descrivere il traffico di rete a livello di flusso, non di pacchetto individuale. Tutte le informazioni di volume sono già aggregate e discretizzate per classi. Utilizzare il numero totale di flussi è quindi pienamente coerente con il livello di astrazione del dato disponibile.
- **Proxy affidabile dell'intensità di traffico:** In contesti di monitoraggio e sicurezza di rete, il numero di flussi è comunemente utilizzato come proxy dell'intensità di traffico. Un aumento del numero di flussi è fortemente correlato a un aumento dell'utilizzo della rete, anche se non misura direttamente i byte trasmessi.
- **Robustezza rispetto a valori estremi:** Modellare il traffico tramite il numero di flussi riduce la sensibilità a flussi eccezionalmente grandi o anomali, che potrebbero dominare una misura basata esclusivamente sui byte. Questo aspetto è particolarmente rilevante in un dataset che include eventi di natura anomala o malevola.
- **Allineamento con l'obiettivo di previsione:** L'obiettivo del lavoro è la previsione del comportamento normale del traffico e l'analisi delle sue dinamiche stagionali. In questo contesto, il numero di flussi rappresenta una grandezza più stabile e prevedibile rispetto al volume fisico, rendendola più adatta a modelli statistici come ARIMA/SARIMA.

In sintesi, sebbene quella che vedremo più avanti diventare la variabile target, non rappresenti il traffico in termini fisici assoluti, essa costituisce una misura consistente dell'attività di rete, pienamente adeguata allo scopo di analisi e previsione temporale perseguito in questo lavoro.

Capitolo 5

Regressione su serie temporale

5.1 Pre-processing

5.1.1 Parsing temporale e indicizzazione

Conclusa l'esplorazione preliminare del dataset si può passare alla fase di *pre-processing* vero e proprio dove si costruirà la serie temporale vera e propria con tutte le caratteristiche adatte a svolgere l'operazione di forecasting.

Un primo problema è rappresentato dalla colonna `Row` che abbiamo visto contenere l'informazione temporale codificata però come numero intero nel formato YYYYMMDDHHMM. Tale rappresentazione non è direttamente utilizzabile per analisi di tipo `time series`. È stata quindi convertita in un oggetto `datetime` tramite parsing esplicito del formato, ottenendo un timestamp con risoluzione al minuto.

Il timestamp così ricostruito è stato impostato come indice del `DataFrame`, permettendo:

- ordinamento cronologico corretto,
- operazioni di differenziazione temporale,
- resampling e aggregazioni successive.

Dopo l'indicizzazione temporale, la serie è stata ordinata in modo crescente rispetto al tempo per garantire la coerenza delle analisi sequenziali. Di seguito codice e output di questa sezione che conferma anche gli estremi temporali della serie (dal 19-03-2016 ore 00:00 al 31-05-2016 ore 23:59).

```
Parsing del timestamp e setting dell'indice

traffic_data = traffic_raw.copy()
traffic_data['timestamp'] = pd_to_datetime(
    traffic_data['Row'].astype(str),
    format='%Y%m%d %H%M'
)
traffic_data.set_index('timestamp', inplace=True)

Ordinamento e verifica

traffic_data.sort_index(inplace=True)
traffic_data.index.min(), traffic_data.index.max()

(Timestamp('2016-03-19 00:00:00'), Timestamp('2016-05-31 23:59:00'))
```

Figura 5.1: Paring dell'indice

È stata inoltre effettuata una verifica esplicita della presenza di righe duplicate. Il controllo ha mostrato assenza di duplicati, confermando che ogni osservazione temporale è rappresentata una sola volta nel dataset.

5.1.2 Analisi della regolarità temporale

Un aspetto critico per l'analisi delle serie temporali è la regolarità del campionamento. Per verificarla, è stata calcolata la differenza temporale tra ciascun timestamp e il precedente come segue:

```
# Calcolo della differenza tra un timestamp e quello precedente
diffs = traffic_data.index.to_series().diff()

# Conteggio della frequenza delle differenze
diff_counts = diffs.value_counts().sort_index()

# Visualizzazione delle differenze
with pd.option_context('display.max_rows', None):
    print(diff_counts)
```

Figura 5.2: Codice per l'analisi della regolarità temporale

Il risultato è invece il seguente:

timestamp	
0 days 00:01:00	71793
0 days 00:02:00	586
0 days 00:03:00	79
0 days 00:04:00	23
0 days 00:05:00	12
0 days 00:06:00	9
0 days 00:07:00	4
0 days 00:08:00	5
0 days 00:09:00	7
0 days 00:10:00	3
0 days 00:11:00	3
0 days 00:12:00	6
0 days 00:13:00	2
0 days 00:14:00	1
0 days 00:15:00	3
0 days 00:16:00	2
0 days 00:17:00	5
0 days 00:19:00	5
0 days 00:20:00	2
0 days 00:21:00	3
0 days 00:22:00	1
0 days 00:23:00	3
0 days 00:24:00	2
0 days 00:25:00	2
...	
0 days 09:16:00	1
5 days 17:19:00	1
11 days 00:18:00	1

Figura 5.3: Visualizzazione della frequenza delle distanze tra le righe

L’analisi delle differenze temporali rivela che:

- la maggior parte delle osservazioni presenta un passo temporale di 1 minuto (71793 su 72644 osservazioni totali), coerente con la risoluzione

nominale del dataset;

- sono presenti numerosi intervalli superiori a 1 minuto (che vanno da pochi minuti fino a diverse ore, alcuni non visibili direttamente in figura perché ad output ridotto);
- sono identificabili due gap temporali estremamente estesi, rispettivamente di circa 5 giorni e 11 giorni, che risultano essere i più lunghi e dunque i più problematici da gestire.

Questi risultati indicano che il dataset non costituisce una serie temporale continua, ma presenta buchi temporali di varia ampiezza, probabilmente dovuti a interruzioni di acquisizione o a problemi di logging, misti a periodi di inattività totale, durante i quali non è stato registrato alcun traffico.

La presenza di tali discontinuità ha implicazioni rilevanti per la costruzione di una serie temporale regolare, requisito fondamentale per procedere all'utilizzo di modelli statistici per la previsione. Tali modelli, infatti, modellano le dipendenze tra osservazioni consecutive, assumendo che la distanza temporale tra due punti adiacenti sia costante. La presenza di buchi temporali viola direttamente questa ipotesi, e il modello interpreterebbe tali transizioni come variazioni improvvise del processo e non come assenze di osservazione. Questo porta a stimare dinamiche che non riflettono il comportamento reale del traffico, ma piuttosto artefatti introdotti dalla mancanza di dati. Inoltre, da un'ulteriore analisi svolta per cercare la sezione di serie temporale più lunga senza buchi che superassero i 15 minuti (soglia empirica provvisoria scelta al solo fine di esempio, al di sotto della quale si potrebbe provare a risolvere il problema con l'interpolazione lineare) si scopre che questo segmento di serie è lungo appena 9 giorni e 14 ore. Sarebbe dunque particolarmente corto da usare per training + test.

Per questo motivo, nelle fasi successive del lavoro, particolare attenzione verrà dedicata alla gestione dei buchi temporali e alla scelta di segmenti di

serie più affidabili.

Infine, dopo aver convertito la colonna temporale in un indice `DatetimeIndex` e aver ordinato cronologicamente le osservazioni, è stato necessario effettuare una reindicizzazione della serie in modo tale da averla campionata regolarmente al minuto, anche nei punti che abbiamo visto mancare. Le righe sono dunque passate da 72644 a 106560. La differenza tra questi valori indica che circa il 32% dei minuti teorici non è stato osservato, confermando che il dataset non è temporalmente completo. L'operazione di reindicizzazione sul nuovo asse temporale (con tutti i timestamp presenti) introduce valori `NaN` in corrispondenza dei minuti mancanti, trasformando un problema "nascosto" (timestamp mancanti) in un problema esplicito (missing values) e che dovrà essere gestito adeguatamente.

5.1.3 Definizione della variabile target

A questo punto si è finalmente pronti per la definizione della variabile target. Come detto, l'obiettivo della modellazione è la previsione del traffico di rete espresso in termini di numero totale di flussi osservati per unità di tempo. Nel dataset UGR'16 tale informazione non è disponibile in forma diretta, ma viene fornita in maniera aggregata attraverso diverse variabili che descrivono il numero di pacchetti suddivisi in classi di intensità del traffico. In particolare, per ogni istante temporale t , il dataset fornisce il numero di pacchetti appartenenti alle seguenti classi:

- very low
- low
- medium
- high
- very high

indicate rispettivamente dalle variabili:

$$npackets_{\text{verylow}}^{(t)}, npackets_{\text{low}}^{(t)}, npackets_{\text{medium}}^{(t)}, npackets_{\text{high}}^{(t)}, npackets_{\text{veryhigh}}^{(t)} \quad (5.1)$$

La variabile target, denominata total_flows, è stata quindi definita come la somma del numero di pacchetti su tutte le classi, ovvero:

$$\text{total_flows}(t) = \sum_{i \in \{\text{verylow}, \text{low}, \text{medium}, \text{high}, \text{veryhigh}\}} npackets_i^{(t)} \quad (5.2)$$

Mappando tutto questo in codice si ricava la feature target della regressione in questo modo:

```
traffic_data['total_flows'] = traffic_data[[
    'npacketsverylow', 'npacketslow', 'npacketsmedium', 'npacketshigh', 'npacketsveryhigh'
]].sum(axis=1, min_count=1)
```

Figura 5.4: Codice per la creazione della variabile target `total_flows`

Da notare che l'opzione `min_count = 1` è importante per garantire che la somma restituisca `NaN` se tutti i valori sono mancanti, mantenendo quindi i missing values laddove già erano presenti in attesa di trovare la strategia adatta per essere gestiti.

5.1.4 Creazione e analisi della serie al minuto

A questo punto si è creata una copia della serie grezza, denominata `traffic_1min` che sarà la prima serie di riferimento su cui lavorare. Fondamentale è ora la scelta sulla gestione dei "buchi" temporali, visto che ovviamente i modelli statistici necessitano di una serie completa e senza `NaN`.

Per quanto riguarda i gap di durata inferiore ai 15 minuti è stato scelto di adottare come strategia quella dell'**interpolazione lineare**. Formalmente, dato un intervallo mancante compreso tra due osservazioni valide $y(t_1)$ e $y(t_2)$, il valore interpolato $\hat{y}(t)$ viene stimato come:

$$\hat{y}(t) = y(t_1) + \frac{y(t_2) - y(t_1)}{t_2 - t_1} \cdot (t - t_1) \quad (5.3)$$

La scelta della soglia di 15 minuti è motivata da considerazioni sia pratiche che statistiche:

- nell’intervallo di pochi minuti, il traffico di rete tende a variare in modo relativamente continuo;
- interpolare su intervalli brevi consente di preservare la struttura locale della serie senza alterare i pattern giornalieri o settimanali;
- soglie più elevate avrebbero introdotto stime arbitrarie su porzioni troppo ampie della serie, rendendo troppo artificiale la ricostruzione.

Il codice che permette l’interpolazione è il seguente:

```

1 traffic_1min[‘total_flows’] = traffic_1min[‘total_flows’].
2     interpolate(
3         method=‘linear’,
4         limit=15
5     )

```

Listing 5.1: Interpolazione lineare dei flussi mancanti

A questo punto rimangono i buchi di grande durata che non possono ovviamente essere ricostruiti tramite interpolazione poichè:

- non esiste informazione locale sufficiente;
- la dinamica del traffico potrebbe essere completamente diversa;
- la ricostruzione introduirebbe una forte componente artificiale.

Per questo motivo, tali intervalli vengono provvisoriamente riempiti con valore 0:

$$total_flows(t) = 0 \quad \text{se } t \in \text{buco esteso} \quad (5.4)$$

che si riflette nel codice con:

```
traffic_1min['total_flows'] = traffic_1min['total_flows'].  
fillna(0)
```

Listing 5.2: Fill a 0 dei gap grandi

Lo zero non viene interpretato come un valore reale di traffico, ma come una codifica esplicita di inattività o indisponibilità del dato. Questa scelta permette di mantenere una serie temporale completa e regolarmente campionata e di distinguere chiaramente tra traffico osservato e periodi non operativi, evitando interpolazioni arbitrarie su buchi estesi.

È importante comunque sottolineare che questa rappresentazione è stata utilizzata solo per l'analisi esplorativa e diagnostica, mentre per la modellazione ARIMA/SARIMAX sono stati successivamente selezionati segmenti temporali continui e affidabili. Dopo aver verificato ulteriormente l'assenza di valori NaN e quindi di gap, si visualizza la serie al minuto per osservarne preliminarmente il suo comportamento.

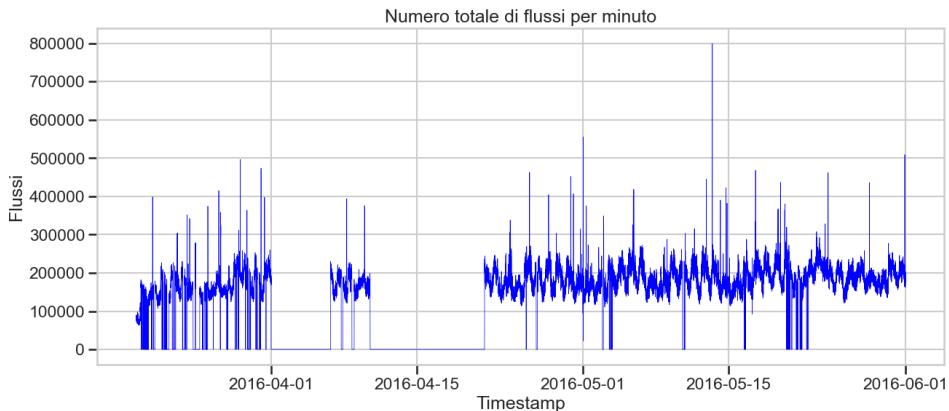


Figura 5.5: Flussi totali di rete al minuto

L'aspetto più evidente del grafico è sicuramente la natura frammentata dell'acquisizione dati. Si osservano chiaramente periodi di attività intervallati da lunghe finestre di silenzio (valori a 0), giustificabili ovviamente dal consapevole pre-processing effettuato e coerentemente con quanto si era già notato con l'analisi dei gap. In particolare, si nota una discontinuità critica di cir-

ca 11 giorni a cavallo tra Aprile e Maggio, anche questo fedele a quanto già analizzato. Questa interruzione rappresenta una rottura strutturale che impedisce l'utilizzo dell'intera serie come un continuum per l'addestramento di modelli autoregressivi, i quali perderebbero la memoria dei lag temporali. È inoltre molto evidente anche l'impatto dei "buchi" minori, con frequenti discese improvvise della serie a 0, soprattutto nella prima parte.

Nonostante queste evidenti criticità nella serie attuale, nelle finestre temporali in cui il dato è presente, emerge in modo inequivocabile un pattern oscillatorio regolare. Si nota abbastanza chiaramente, nonostante la densità dei punti, un **ciclo giorno/notte**: il traffico segue fedelmente i ritmi antropici, con picchi di carico verosimilmente durante le ore lavorative e minimi fisiologici durante le ore notturne. La presenza di questa "onda" regolare conferma che la serie non è stazionaria nella media, ma possiede una forte componente stagionale ($S = 24h$ o multipli) che il modello predittivo dovrà catturare esplicitamente.

Oltre all'oscillazione fisiologica, si notano sporadici picchi verticali (*spikes*) che deviano significativamente dal trend locale. Tali picchi, che superano talvolta di 2-3 volte il volume medio, sono i candidati ideali per rappresentare eventi di sicurezza (es. attacchi DDoS volumetrici o scansioni aggressive). Una delle possibili applicazioni del forecasting del traffico di rete potrebbe essere proprio un modulo di Anomaly Detection che confrontando in tempo reale la propria previsione con il carico reale, individua i residui anomali (i picchi che vediamo qui) e che potrebbero rappresentare problemi di sicurezza.

La conclusione è che in queste condizioni la serie è praticamente inutilizzabile per la presenza massiccia di valori nulli introdotti dai gap.

Per avere comunque una visualizzazione più chiara, ricercando una porzione di serie con interruzioni minime, si visualizza un estratto zoommato di una settimana nel periodo che va dal 25-04-2016 al 01-05-2016:

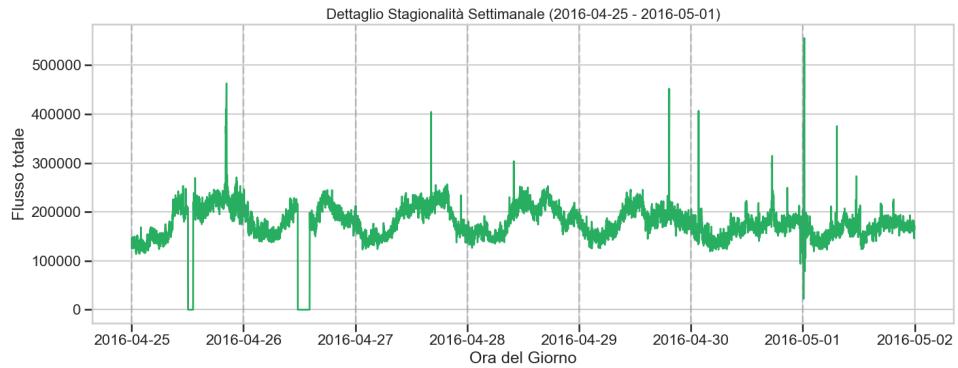


Figura 5.6: Visualizzazione settimanale del flusso totale al minuto

Questa visione settimanale valida ancora di più la ciclicità giornaliera con un pattern oscillatorio persistente di 24 ore. Si nota un incremento ripido del traffico nelle prime ore della mattina, corrispondente all'inizio delle attività lavorative, il carico si mantiene elevato durante le ore centrali, con picchi secondari spesso coincidenti con le fasce serali per poi avere un calo fisiologico netto nella notte, dove il traffico si riduce al "rumore di fondo" dei servizi automatizzati.

La visualizzazione al minuto evidenzia inoltre un'elevata varianza istantanea. La linea non è liscia ("smooth") ma presenta una "rugosità" continua. Questo può far pensare che per la modellazione predittiva, lavorare a 1 minuto potrebbe introdurre troppo rumore, rendendo difficile la convergenza del modello.

Infine, nei giorni festivi (30-04 e 01-05), si nota come il volume di traffico diminuisca in senso assoluto e la curva di salita mattutina è traslata in avanti (ritardo nell'inizio delle attività di rete). Questo potrebbe suggerire l'utilizzo di variabili esogene che indichino i giorni di weekend per evitare una sovrastima del carico di rete in tali giorni.

Questo grafico conferma che, al netto dei grandi buchi strutturali, il processo generatore dei dati è stabile, ripetitivo e prevedibile, validando l'ipotesi di base per l'applicazione di algoritmi di Time Series Forecasting.

5.1.5 Creazione e analisi della serie oraria

Come già osservato, dopo la ricostruzione della serie temporale regolare al minuto, è emerso che, nonostante le operazioni di interpolazione e gestione dei buchi temporali, la serie presentava ancora una forte variabilità ad alta frequenza e una significativa rumorosità. Oltre a questo il vero problema principale però è la presenza di un numero elevato e soprattutto diffuso di intervalli temporali con traffico nullo, frutto ovviamente dei dati non disponibili nel dataset originario come ampiamente discusso.

La soluzione viene dunque individuata nell'operazione di *downsampling* aggregato, passando cioè da una risoluzione al minuto a una serie oraria denominata `traffic_hours`. Il passaggio alla serie oraria permette così di:

- ridurre l'impatto dei buchi a zero;
- aumentare la probabilità che ciascun intervallo temporale contenga almeno una porzione di informazione reale;
- ottenere una serie temporale più densa, stabile e statisticamente significativa.

Aggregando i dati su finestre di 60 minuti, è infatti ragionevole aspettarsi che, anche in presenza di dati mancanti a livello minuto, l'ora complessiva contenga comunque traffico osservato. In questo modo, molti degli zeri artificiali presenti nella serie al minuto vengono assorbiti dall'aggregazione, producendo una rappresentazione più fedele del volume reale di traffico.

Dal punto di vista formale, l'aggregazione oraria consente quindi di trasformare una serie altamente sparsa in una serie più informativa, in cui:

- i valori nulli sono meno frequenti e più semanticamente interpretabili;
- le variazioni riflettono dinamiche reali e non artefatti di acquisizione;

- la struttura stagionale giornaliera (24 osservazioni) e settimanale risulta più chiara e più facilmente modellabile.

Dal punto di vista pratico, la nuova serie viene costruita così:

```
1 traffic_hours = traffic_data[['total_flows']].resample('60min')
   .sum(min_count=1)
```

Listing 5.3: Resampling dei dati su base oraria

L'aggregazione è stata effettuata tramite somma dei flussi all'interno di ciascuna ora, utilizzando `min_count=1` per preservare i valori mancanti nei casi in cui l'intera ora fosse priva di osservazioni valide. Formalmente:

$$Y_h = \sum_{t \in h} total_flows(t) \quad (5.5)$$

dove h indica un intervallo orario e la somma è calcolata solo se almeno un valore valido è presente nell'intervallo. La visualizzazione preliminare della serie ha dato questo esito:

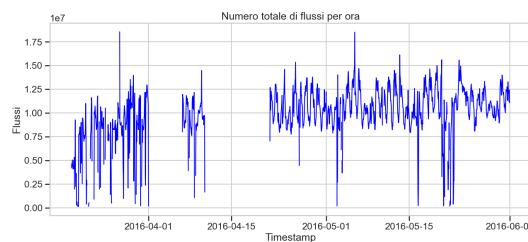


Figura 5.7: Visualizzazione della serie oraria

Quello che si nota è un miglioramento minimo della situazione (si hanno meno tratti di discesa a 0) ma soprattutto un comportamento apparentemente anomalo: la presenza di crolli improvvisi e isolati del traffico (non a 0 ma a valori bassi), non coerenti con la dinamica complessiva della serie.

5.1.6 Ricostruzione del profilo della serie oraria

In realtà, molti di questi crolli non rappresentano una reale diminuzione del traffico, ma sono invece dovuti a registrazioni parziali dell'ora, frutto quindi del resample orario che è stato effettuato ad esempio:

- ore per cui sono disponibili solo pochi minuti di dati;
- interruzioni temporanee del sistema di raccolta;

Per individuare sistematicamente questi artefatti, è stato costruito un profilo di riferimento storico basato sulla struttura settimanale del traffico.

In particolare, per ogni combinazione di:

- giorno della settimana

$$d \in \{0, \dots, 6\} \quad (5.6)$$

,

- ora del giorno

$$h \in \{0, \dots, 23\} \quad (5.7)$$

è stata calcolata la mediana dei flussi osservati nel tempo. La mediana è stata scelta al posto della media perché:

- è robusta rispetto a valori estremi;
- non è influenzata dai crolli artificiali dovuti a registrazioni parziali;
- rappresenta una stima più stabile del comportamento “tipico” del traffico per quella specifica ora e giorno.

Questo profilo mediano rappresenta quindi un baseline affidabile del traffico atteso. A questo punto, un'osservazione è stata etichettata come sospetta (artefatto) se il suo valore risultava inferiore al 50% del profilo di riferimento:

$$Y_h < 0.5 \cdot \text{Mediana}(d, h) \quad (5.8)$$

La soglia del 50% è stata scelta come compromesso tra:

- sensibilità nell'individuare dati parziali,
- robustezza nel non classificare come artefatti reali variazioni fisiologiche del traffico.

Di seguito il codice di riferimento:

```
# Calcolo della mediana storica per ogni (Giorno, Ora) da usare come riferimento
reference_profile = df_fix.groupby([df_fix.index.dayofweek, df_fix.index.hour])['total_flows'].transform('median')

# soglia per identificare i dati parziali (artefatti)
threshold_ratio = 0.50
is_too_low = df_fix['total_flows'] < (reference_profile * threshold_ratio)
```

Figura 5.8: Codice per l'individuazione degli artefatti

Visualizzandoli, i punti candidati ad essere artefatti sono quelli evidenziati in rosso:

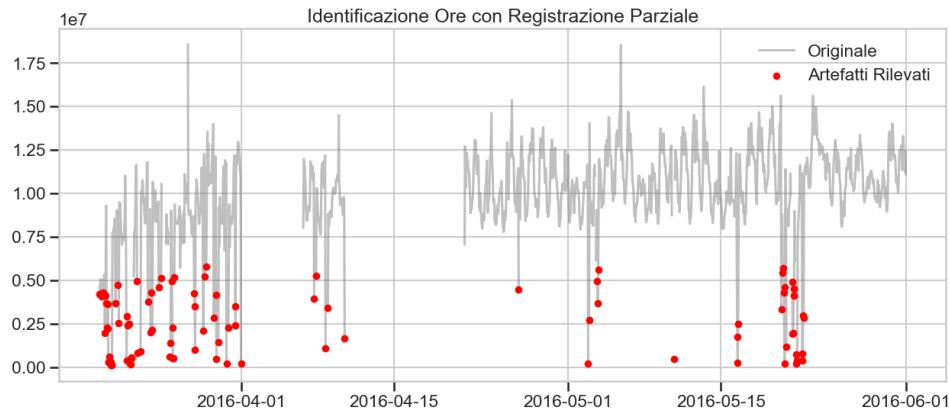


Figura 5.9: Visualizzazione della serie con artefatti rilevati

Come si nota, questo sistema di individuazione è stato particolarmente efficace:

- Si osserva come i punti rossi siano concentrati esclusivamente in corrispondenza di crolli verticali improvvisi. Questi eventi presentano una morfologia a "V" molto stretta, caratteristica tipica di un'interruzione temporanea del servizio di logging (es. il sensore ha registrato solo per 10 minuti su 60 disponibili nell'ora).

- Un aspetto cruciale validato visivamente dal grafico è la capacità di non generare falsi positivi durante le ore notturne. Sebbene i volumi di traffico nelle ore notturne siano bassi in valore assoluto, essi seguono la curva fisiologica del Profilo di Riferimento (la mediana storica che è stata scelta). Di conseguenza, il filtro non marca i minimi notturni come anomalie (nessun punto rosso nelle valli naturali che si identificano chiaramente), ma colpisce solo quando il dato scende significativamente al di sotto di quanto atteso per quella specifica ora.
- La visualizzazione permette inoltre di diagnosticare lo stato di salute del sistema di monitoraggio: la presenza dei punti rossi è infatti sporadica e non periodica e suggerisce che gli errori di acquisizione sono eventi casuali e non sistematici.

Una volta identificati i punti sospetti, questi non sono stati corretti direttamente, ma temporaneamente rimossi, impostando il loro valore a `NaN`. Successivamente, per la ricostruzione dei valori, è stato utilizzato un profilo medio calcolato come media dei flussi per ciascuna coppia (giorno della settimana, ora), escludendo i valori identificati come artefatti. La scelta di usare la media in questa fase, anziché la mediana, è motivata dal fatto che:

- dopo la rimozione degli artefatti, la distribuzione dei dati risulta più pulita;
- la media preserva meglio il livello informativo complessivo del traffico;
- la media consente una ricostruzione coerente con l'intensità media reale osservata.

In questo modo, gli artefatti vengono sostituiti con valori plausibili, coerenti con la struttura settimanale del traffico, senza introdurre discontinuità artificiali. Il codice è il seguente:

```

# Set a NaN dei valori "sporchi" (artefatti).
df_fix.loc[is_too_low, 'total_flows'] = np.nan

# Calcolo del profilo medio (media per ogni (GiornoSettimana, Ora) escludendo i NaN degli artefatti).
profile_means = df_fix.groupby([df_fix.index.dayofweek, df_fix.index.hour])['total_flows'].transform('mean')

# Riempimento artefatti con il profilo medio
df_fix.loc[is_too_low, 'total_flows'] = df_fix.loc[is_too_low, 'total_flows'].fillna(profile_means)

```

Figura 5.10: Codice per la correzione degli artefatti

Oltre agli artefatti, la serie oraria presenta anche buchi temporali reali, ossia ore consecutive completamente prive di osservazioni valide che sono permaste anche dopo il resampling orario (seppur in misura nettamente minore). Per i buchi di durata inferiore o uguale a 4 ore, si è scelto di non utilizzare l’interpolazione lineare come metodo di riempimento finale. Infatti, sebbene l’interpolazione lineare sia adatta a serie continue, in questo contesto presenta diversi limiti:

- non rispetta la struttura stagionale giornaliera e settimanale;
- tende a “smussare” variazioni reali del traffico che è fondamentale cogliere (ancora di più ora che si è ridotta la frequenza di osservazione);
- può produrre valori non coerenti con il profilo tipico dell’ora specifica.

Per questo motivo, l’interpolazione è stata utilizzata solo come strumento diagnostico, per identificare i buchi piccoli. Se un valore poteva essere interpolato entro il limite di 4 ore, il buco è stato considerato “piccolo”.

I valori corrispondenti a questi buchi sono stati quindi riempiti utilizzando il profilo medio orario-settimanale, garantendo coerenza semantica e stagionale.

I buchi temporali di durata superiore alle 4 ore sono stati trattati separatamente. In questi casi, la mancanza prolungata di dati rende qualsiasi tentativo di ricostruzione altamente speculativo ed artificiale. Per questo motivo si è scelto di mantenere la logica del riempimento a 0 per questi buchi temporali grandi. Di seguito il codice per la gestione dei buchi piccoli con il profilo medio e grandi con il fill a 0:

```

1 # Identificazione dei buchi piccoli (max 4 minuti)
mask_small_holes = df_fix['total_flows'].interpolate(method='
    linear', limit=4).notna() & df_fix['total_flows'].isna()

3

# Riempimento dei buchi identificati con il profilo medio
5 df_fix.loc[mask_small_holes, 'total_flows'] = profile_means[
    mask_small_holes]

7 # Chiusura dei buchi estesi rimanenti con valore zero
df_fix['total_flows'] = df_fix['total_flows'].fillna(0)

```

Listing 5.4: Procedura di imputazione ibrida: interpolazione e medie storiche

Il risultato finale visibile è il seguente:

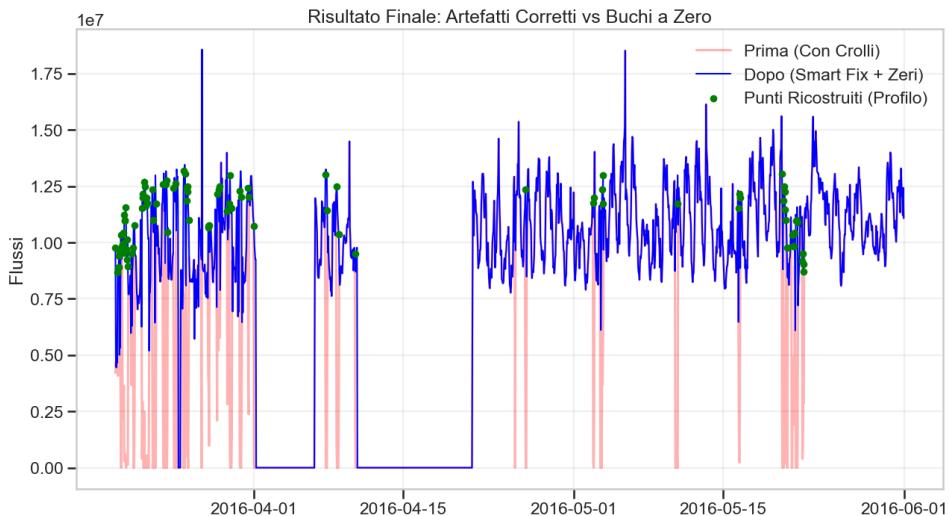


Figura 5.11: Profilo ricostruito della serie

La curva blu rappresenta la serie ricostruita, sovrapposta alla traccia originale (in rosso chiaro) che evidenziava i dati mancanti o corrotti. I punti verdi rappresentano i valori che hanno assunto dopo la ricostruzione col valore medio quelli che erano i punti rossi catalogati come artefatti nella fase precedente. Dall'analisi visiva emergono tre conclusioni fondamentali sulla qualità del dato post-processamento:

- Nelle finestre temporali attive, la strategia di imputazione basata sul Profilo Medio Orario si è dimostrata efficace nel ripristinare la continuità del segnale. Gli artefatti (crolli improvvisi dovuti a registrazioni parziali) e i micro-gap sono stati sostituiti con valori statisticamente coerenti con la stagionalità del giorno e dell'ora. Il risultato è che la curva blu appare ora fluida e priva di quel "rumore negativo" che avrebbe distorto la varianza del modello, preservando la forma naturale dei picchi di traffico diurni.
- Nonostante la pulizia locale, il grafico evidenzia l'impossibilità di recuperare intere settimane di inattività del sensore (il gap di 11 giorni e quello di 5 giorni di cui già eravamo a conoscenza). In accordo con una metodologia conservativa, questi periodi sono stati mantenuti a valore 0, evitando di introdurre dati sintetici su orizzonti temporali troppo estesi, che avrebbero generato un bias di "falsa certezza" nel modello.
- Il grafico rivela una netta dicotomia nella densità informativa della serie:
 - Fase 1 (Marzo - 20 Aprile circa): Sebbene i micro-buchi siano stati curati, questa sezione rimane frammentata da interruzioni sistematiche e presenta ancora residui di discontinuità (es. un buco isolato di alcune ore non completamente recuperabile). La frequenza delle interruzioni resetterebbe continuamente la memoria dei lag autoregressivi, rendendo questa porzione inaffidabile per il training.
 - Fase 2 (21 Aprile circa - 31 Maggio): A partire dalla ripresa dopo l'interruzione più grande, la serie mostra una stabilità e una continuità molto buone. La ricostruzione puntuale ha sanato le piccole imperfezioni, restituendo un blocco compatto di circa 40 giorni di dati ad alta fedeltà.

Prima di proseguire, anche in questo caso si è visualizzato lo zoom settimanale che rileva la stagionalità giornaliera della serie e anche il calo in valore assoluto del traffico nel weekend.

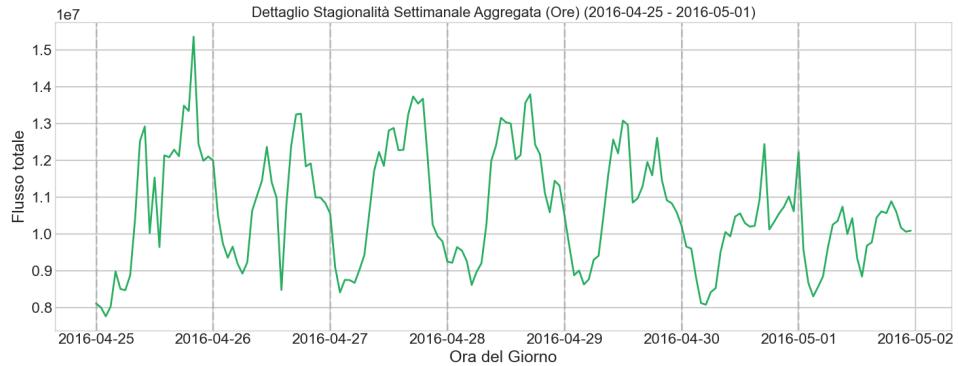


Figura 5.12: Zoom per la settimana dal 25-04 al 01-05 per la serie oraria

A questo punto, per certificare l'inutilità di una serie con innumerevoli valori mancanti riempiti a 0, è stata condotta un'analisi specifica per valutare l'impatto di tali valori nulli sulla stima della stagionalità giornaliera. L'obiettivo di questa fase non è la correzione dei dati, ma la quantificazione dell'effetto distorsivo che la presenza degli zeri introduce nelle statistiche stagionali. Sono stati così costruiti due profili:

- **Profilo “sporco” (dirty profile)** Calcolato come la media oraria dei flussi includendo anche le osservazioni a zero:

$$\bar{x}_{dirty}(h) = \mathbb{E}[X_t \mid hour(t) = h] \quad (5.9)$$

- **Profilo “pulito” (clean profile)** Calcolato escludendo esplicitamente tutte le osservazioni con traffico nullo:

$$\bar{x}_{clean}(h) = \mathbb{E}[X_t \mid hour(t) = h, X_t > 0] \quad (5.10)$$

Questa doppia costruzione consente di confrontare la stagionalità osservata con e senza l'effetto dei buchi temporali, isolando il contributo degli zeri artificiali introdotti durante la ricostruzione della serie. Il risultato è il seguente:

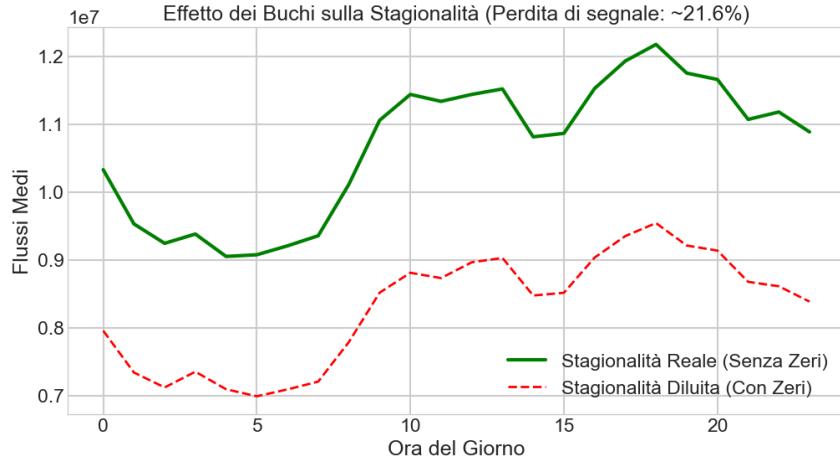


Figura 5.13: Effetto dei gap sulla stagionalità

Questa doppia costruzione consente di confrontare la stagionalità osservata con e senza l'effetto dei buchi temporali, isolando il contributo degli zeri artificiali introdotti durante la ricostruzione della serie. La struttura giornaliera e le ore di minimo e massimo sono preservate ma la differenza riguarda il livello assoluto dei flussi medi. Per misurare quantitativamente questo effetto, è stata calcolata la perdita percentuale del segnale stagionale confrontando i massimi dei due profili:

$$\text{Perdita} = \left(1 - \frac{\max(\bar{x}_{\text{dirty}})}{\max(\bar{x}_{\text{clean}})} \right) \cdot 100 \quad (5.11)$$

La perdita risulta essere del **21.6%** circa. Questo risultato dimostra che l'inclusione degli zeri comporta una sottostima sistematica del livello di traffico. A questo si aggiunge ovviamente che gli zeri, per i modelli statistici che saranno usati vengono interpretati come shock improvvisi, cambi strutturali e innovazioni ad alta varianza. Questo porta a una "rottura" della dinamica autoregressiva, falsando la varianza, rompendo la stagionalità e amplificando il rumore.

5.1.7 Taglio della serie

Come già detto, dopo le fasi di ricostruzione, aggregazione e correzione della serie temporale, si osserva che il dataset completo presenta ancora una forte

eterogeneità strutturale nella parte iniziale della registrazione. In particolare, la porzione iniziale (marzo–metà aprile) è caratterizzata da:

- maggiore presenza di buchi estesi,
- artefatti residui,
- profili di traffico meno stabili,
- discontinuità dovute a problemi di acquisizione.

si è deciso di tagliare la serie temporale a partire da una data di cutoff selezionata manualmente sulla base dell’ispezione visiva e statistica.

```
# Definizione della data di inizio analisi
2 CUT_OFF_DATE = '2016-04-21 12:00:00'

4 # Selezione dei dati puliti a partire dalla data di cut-off
traffic_hours_clean = traffic_hours.loc[CUT_OFF_DATE: ].copy()
```

Listing 5.5: Definizione del periodo di analisi tramite Cut-off

Di seguito le figure che visualizzano il taglio e la serie pulita sulla quale si lavorerà da questo momento in poi, molto più regolare e soprattutto senza gap temporali.

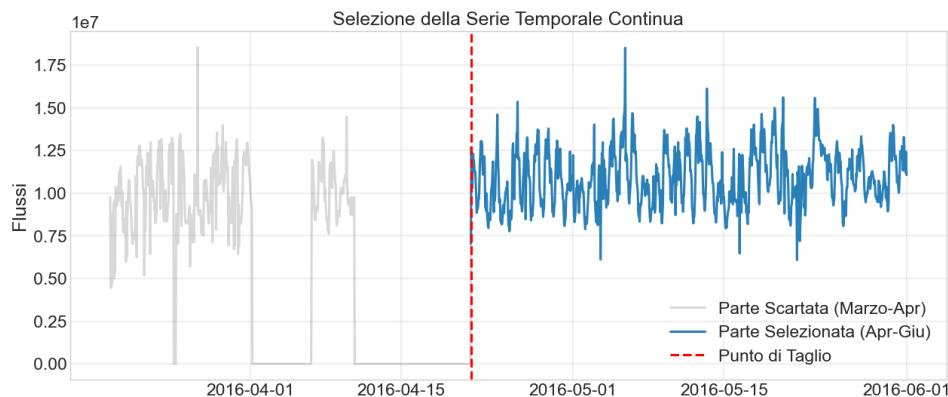


Figura 5.14: Taglio della serie temporale

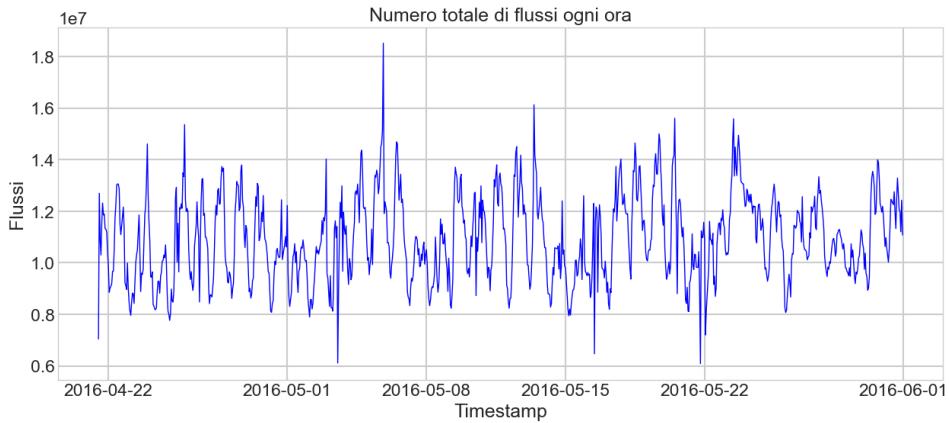


Figura 5.15: Serie finale

5.2 Analisi preliminare della stabilità

Una volta selezionata la nuova serie finale:

```
1 # Estrazione della colonna total_flows come oggetto Series
2 ts = traffic_hours_clean['total_flows']
```

Listing 5.6: Estrazione della serie temporale per l'analisi statistica

si procede con un'analisi di stabilità statistica locale, calcolando **media mobile** e **deviazione standard mobile** su una finestra di 24 ore:

```
1 # Calcolo della media mobile con finestra di 24 ore
2 rolling_mean = ts.rolling(window=24).mean()

4 # Calcolo della deviazione standard mobile con finestra di 24
   ore
rolling_std = ts.rolling(window=24).std()
```

Listing 5.7: Calcolo della media e deviazione standard mobile (finestra 24h)

La scelta di una finestra di 24 ore è coerente con la periodicità giornaliera del traffico che già sappiamo esserci in quanto già osservata in fase esplorativa. L'obiettivo non è verificare la stazionarietà formale (che verrà testata

con ADF), ma osservare se media e varianza fluttuano in modo sistematico nel tempo.

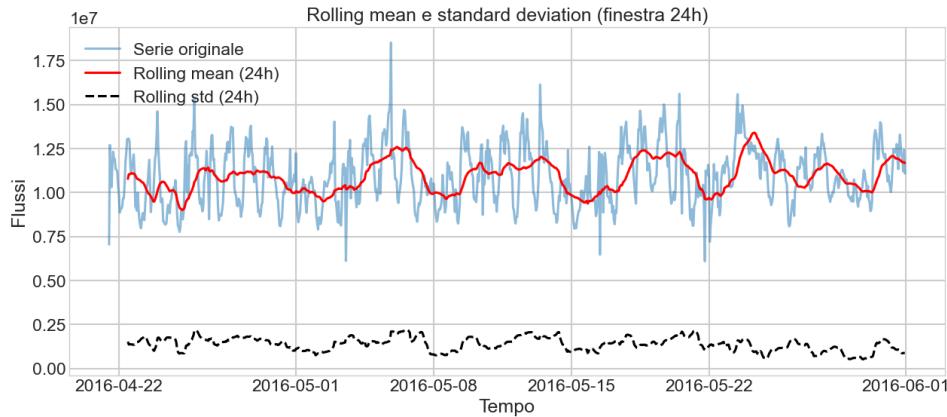


Figura 5.16: Rolling mean e rolling standard deviation

Dal grafico si nota che:

- la media mobile segue un andamento regolare, con oscillazioni coerenti con il ciclo giorno/notte;
- la deviazione standard è relativamente stabile, senza esplosioni improvvise.

Questo suggerisce che la serie è sicuramente più regolare rispetto al dataset iniziale, a conferma della buona riuscita del pre-processing, ma, verosimilmente, non ancora stazionaria, rendendo necessaria una modellazione che includa stagionalità e differenziazione.

5.3 Decomposizione STL

Per comprendere in modo più profondo la struttura del segnale, si applica una decomposizione STL (Seasonal-Trend decomposition using Loess):

```

1 from statsmodels.tsa.seasonal import STL

3 # Configurazione ed esecuzione della scomposizione STL

```

```
# period=24 indica la stagionalità giornaliera (24 ore)
5 stl = STL(ts, period=24, robust=True)
result = stl.fit()
```

Listing 5.8: Scomposizione stagionale STL della serie temporale

STL consente di catturare stagionalità non perfettamente costante, trend non lineare e garantisce anche maggiore robustezza agli outlier (`robust=True`). Per questo è particolarmente adatta a serie reali di traffico di rete. Il periodo stagionale è impostato a 24 per catturare il ciclo giornaliero. Invece, una volta rimosse trend e stagionalità, i residui dovrebbero oscillare intorno a zero e presentare varianza relativamente costante. Per individuare eventi anomali è stata utilizzata una soglia statistica basata sulla regola delle 3 deviazioni standard:

$$soglia = \mu \pm 3\sigma \quad (5.12)$$

In codice:

```
# Calcolo dei limiti superiore e inferiore (regola dei 3-sigma)
2 upper_threshold = mu + 3 * sigma
lower_threshold = mu - 3 * sigma
4
# Estrazione dei punti della serie temporale identificati come
# anomalie
6 anomalies = ts[(resid > upper_threshold) | (resid <
lower_threshold)]
```

Listing 5.9: Identificazione delle anomalie tramite soglie statistiche sui residui

Questo è il risultato:

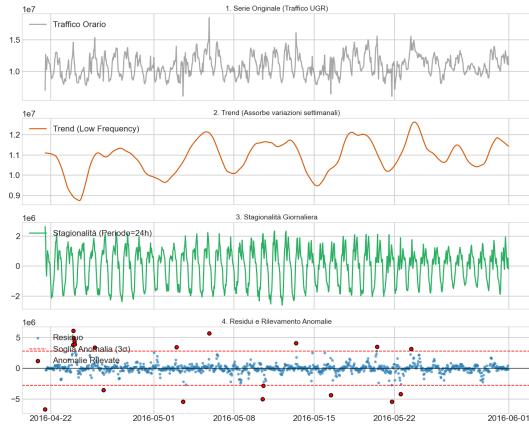


Figura 5.17: Decomposizione STL

- **Componente di trend:** la linea del Trend non è piatta né monotona, ma presenta un andamento ondulatorio "lento" e regolare, con cicli che si ripetono circa ogni 7 giorni. Poiché l'algoritmo STL è stato configurato con un periodo di stagionalità giornaliero ($period = 24$), la componente *Seasonal* (verde) cattura esclusivamente il ciclo circadiano (00:00 - 23:59). Di conseguenza, le variazioni macroscopiche tra giorni feriali e giorni festivi non possono essere assorbite dalla componente stagionale e confluiscono nel Trend. Le "valli" (minimi locali) del Trend corrispondono ai fine settimana (Sabato/Domenica), mentre i "plateau" (massimi locali) rappresentano i giorni lavorativi. Questo conferma che il modello STL ha correttamente separato l'alta frequenza (giorno/notte) dalla bassa frequenza (settimana).
- **Componente Stagionale:** la componente stagionale mostra un pattern oscillatorio estremamente regolare con frequenza giornaliera ($T = 24h$). L'onda presenta un'ampiezza massima costante di circa 2.6 milioni di flussi, riflettendo fedelmente il ciclo di attività antropica (picchi diurni e minimi notturni). Inoltre, la pulizia e la costanza di questa componente confermano che, in assenza di anomalie, il traffico di base (tecnicamente *Background Traffic*) è un processo fortemente stazionario nella sua

ciclicità.

- **Componente residuale:** Il grafico dei residui evidenzia una netta separazione tra il "rumore di fondo" e gli eventi anomali. La maggior parte delle osservazioni si concentra attorno allo zero, indicando che il modello STL ha catturato correttamente la varianza nominale del traffico. Si osservano però anche picchi isolati ed estremi che raggiungono valori fino a 6.1 milioni di flussi, superando di oltre due volte l'ampiezza della normale stagionalità ($6.1M > 2.6M$). Questi picchi spuri nascono molto probabilmente da eventi di traffico strordinari o attacchi e incidenti di rete (essendo UGR un dataset con dati reali contiene anche questi eventi). La presenza di questi picchi confinata interamente nei residui dimostra che gli eventuali attacchi presenti nel dataset sono di natura additiva e impulsiva. Non alterano il trend o la stagionalità sottostante, ma si sommano ad essi violentemente.

5.4 Verifica della stazionarietà

Una volta ottenuta una serie temporale oraria pulita, continua e strutturalmente omogenea, si procede alla suddivisione dei dati in training set e test set.

A differenza di problemi di regressione classica, nelle serie temporali non è possibile effettuare uno split casuale, poiché questo violerebbe la dipendenza temporale tra le osservazioni. Per questo motivo viene adottata una strategia di hold-out temporale, basata sull'ultima porzione della serie.

```
# Definizione della finestra di test (7 giorni)
2 TEST_DAYS = 7

4 # Calcolo della data di separazione basata sull'ultima
   osservazione disponibile
```

```

cutoff_date = traffic_hours_clean.index.max() - pd.Timedelta(
    days=TEST_DAYS)

# Creazione dei subset per l'addestramento e la validazione
train_set = traffic_hours_clean.loc[:cutoff_date]
test_set = traffic_hours_clean.loc[cutoff_date:]

```

Listing 5.10: Calcolo dinamico della data di cutoff per il test set

Questa scelta permette di testare il modello su esattamente una settimana, vedendo il comportamento sia nei giorni feriali che nel weekend e avendo comunque giorni sufficienti a disposizione per valutare l'apprendimento del pattern giornaliero. Inoltre, si ha in questo modo:

- **Training:** 804 ore (33.5 giorni) ovvero circa l’80% del dataset
- **Test:** 168 ore (7 giorni), circa il restante 20%

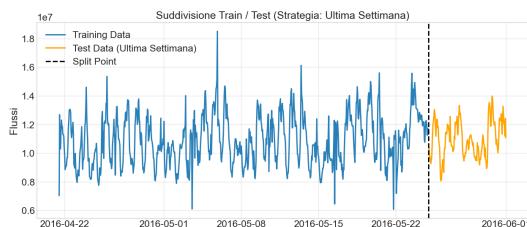


Figura 5.18: Split del dataset

È ora necessario verificare una delle ipotesi fondamentali di questi modelli: la stazionarietà della serie temporale.

Per questo scopo viene applicato il test **Augmented Dickey-Fuller (ADF)** esclusivamente sul training set, in modo da non utilizzare informazioni future.

Il test è interpretabile come:

- Ipotesi nulla (H_0): la serie possiede una radice unitaria e quindi non è stazionaria
- Ipotesi alternativa (H_1): la serie è stazionaria

Il test dà i seguenti risultati:

```
ADF Test Statistic      -2.850431
p-value                 0.051423
Lags Used                21.000000
Number of Observations   782.000000
Critical Value (1%)       -3.438740
Critical Value (5%)       -2.865243
Critical Value (10%)      -2.568742
dtype: float64

⚠ Serie non stazionaria: considerare differenziazione
```

Figura 5.19: Risultati del test

Il risultato è borderline, ma formalmente il **p-value** è leggermente superiore alla soglia 0.05 e la statistica del test non supera il valore critico al 5%. Di conseguenza, non è possibile rifiutare l'ipotesi nulla, e la serie viene considerata non stazionaria. Questo risultato è coerente con quanto osservato in precedenza e cioè la presenza di trend a bassa frequenza, forte stagionalità giornaliera e variazioni sistematiche legate alla settimana.

Poiché la serie non risulta stazionaria, è necessario dunque applicare una o più operazioni di differenziazione, che nei modelli SARIMA sono di due tipi:

- **Differenziazione non stagionale (d)**: serve a rimuovere trend e variazioni lente nel livello medio della serie.
- **differenziazione stagionale (D)**: serve a rimuovere pattern periodici ripetitivi (in questo caso giornalieri, con periodo 24).

Data la struttura del segnale osservata tramite rolling statistics e decomposizione STL, è ragionevole ipotizzare la presenza sia di non stazionarietà non stagionale sia stagionale. Per questo motivo non ci si limita a testare una sola configurazione, ma si valutano sistematicamente tre casi:

- $d = 1, D = 0$: rimozione del solo trend non stagionale
- $d = 0, D = 1$: rimozione della sola componente stagionale
- $d = 1, D = 1$: rimozione congiunta di trend e stagionalità

Per ciascuna configurazione, l'ADF test viene nuovamente calcolato sulla serie trasformata e in tutti e tre i casi il risultato è di serie stazionaria.

5.5 Analisi di ACF e PACF

L'identificazione strutturale del modello SARIMA è stata condotta analizzando il comportamento delle funzioni di autocorrelazione su diverse trasformazioni della serie temporale (differenziazione semplice d e stagionale D). Di seguito si riporta l'interpretazione puntuale dei pattern osservati e la relativa figura:

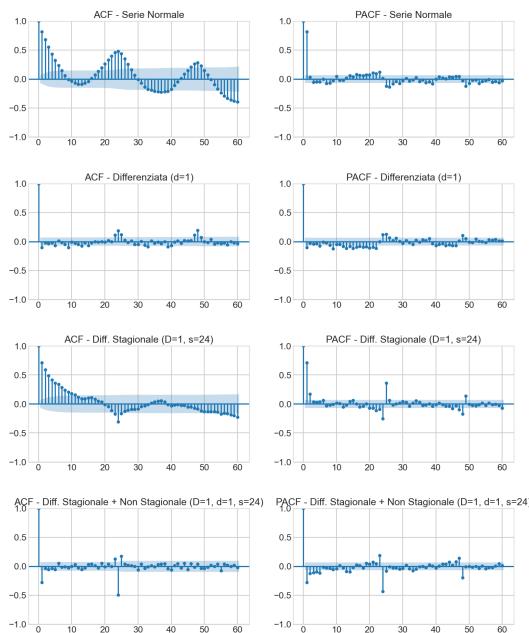


Figura 5.20: ACF e PACF

- **Caso $d = 0$ e $D = 0$:** Introdotto solo per completezza, si nota la difficoltà nell'operare in questo modo che certifica la non stazionarietà della serie e la forte stagionalità.
- **Caso $d = 1$ e $D = 0$:** Per quanto riguarda l'ACF, si osserva una persistenza significativa della correlazione ai lag stagionali (24, 48, 72...). I picchi si ripetono ciclicamente senza decadere rapidamente verso lo zero. La PACF presenta anch'essa picchi molto marcati ai lag multipli di 24.

La differenza semplice $d = 1$ ha rimosso quindi il trend locale, ma ha lasciato intatta la forte struttura stagionale. La serie non è ancora del tutto stazionaria stagionalmente.

Questo scenario suggerisce che, se volessimo evitare la differenziazione stagionale ($D = 0$) per mantenere bassa la varianza, saremmo costretti a introdurre parametri autoregressivi stagionali "forti" ($P = 2$ o più) per catturare quell'onda persistente. Da qui nasce l'idea di testare la configurazione $(0, 1, 1) \times (2, 0, 0, 24)$.

- **Caso $d = 0$ e $D = 1$:** L'ACF mostra un decadimento lento nei primissimi lag (1, 2, 3...), pur avendo rimosso i picchi ai lag 24 e 48. La differenza stagionale ha quindi rimosso correttamente il ciclo giorno/notte, ma la serie residua mostra ancora un comportamento non stazionario nel breve termine. Questo suggerisce che il modello deve concentrarsi principalmente sulla componente stagionale. Si potrebbe dunque testare un modello $(2, 0, 0) \times (0, 1, 1, 24)$ che testa l'ipotesi che la dinamica a breve termine possa essere approssimata da un processo $AR(2)$ senza forzare un trend stocastico ($d = 0$).
- **Caso $d = 1$ e $D = 1$:** Questa è la trasformazione che rende la serie stazionaria e su cui si basano i modelli principali.
 - Analisi della Componente Non Stagionale (Lag 1-5): L'ACF al lag 1 mostra un picco netto e negativo (significativo), seguito da un taglio brusco o valori non significativi. Questa è la firma classica di un processo a Media Mobile ($q = 1$). La PACF al lag 1 mostra un picco che decade poi. Questo conferma la predominanza della componente MA rispetto a quella AR . In conclusione questo pattern giustifica la scelta di $q = 1$ e $p = 0$ come base. Tuttavia, se visto che la PACF al lag 1 è significativa, potrebbe indicare un'inerzia autoregressiva residua, giustificando anche un test di $p = 1$.

- Analisi della Componente Stagionale (Lag 24): L'ACF al lag 24 presenta un picco netto e negativo isolato. Esattamente come per il caso non stagionale, questo indica un processo a Media Mobile Stagionale ($Q = 1$). Il termine $D = 1$ ha rimosso la stagionalità ma ha creato una correlazione negativa artificiale a distanza di 24 ore che va corretta. La PACF al lag 24 mostra un picco negativo seguito da decadimento e conferma il modello MA stagionale. In conclusione la combinazione di ACF negativo a lag 1 e lag 24 fa pensare a un modello $(0, 1, 1)(0, 1, 1)_{24}$. Dato che però la PACF al lag 24 mostra un segnale ancora ambiguo, è lecito testare l'aggiunta di un termine autoregressivo stagionale ($P = 1$) per vedere se cattura meglio la dipendenza diretta dal giorno precedente.

In conclusione, da questa analisi emergono 5 modelli testabili per capire quale sia il migliore:

- **Baseline:** $(0, 1, 1) \times (0, 1, 1, 24)$ è il modello "baseline" che ci aspettiamo essere migliore da ACF e PACF.
- **Test AR:** $(1, 1, 1) \times (0, 1, 1, 24)$ per verificare se l'introduzione di un termine AR non stagionale migliora il modello (valore di un lag dipende più fortemente dal precedente).
- **Test SAR:** $(0, 1, 1) \times (1, 1, 1, 24)$ per verificare se la stagionalità potrebbe anche essere autoregressiva, cioè quanto è forte la dipendenza diretta tra lo stesso orario di giorni consecutivi.
- **Solo $d = 1$:** $(0, 1, 1) \times (2, 0, 0, 24)$ per verificare se, evitando la doppia differenziazione, la stagionalità può comunque essere catturata con AR stagionali.
- **Solo $D = 1$:** $(2, 0, 0) \times (0, 1, 1, 24)$ per verificare se il trend globale è trascurabile e basta la sola stagionalità.

5.6 Scelta del modello migliore

La valutazione dei modelli di previsione è stata condotta secondo una procedura rigorosa di tipo *out-of-sample*, coerente con le buone pratiche della modellazione di serie temporali. In particolare, ciascun modello candidato è stato addestrato esclusivamente sul training set, costituito dall'intera serie storica disponibile fino al punto di cutoff temporale, mentre la valutazione delle prestazioni è stata effettuata sull'ultima settimana di dati, completamente esclusa dalla fase di stima.

Per ogni configurazione SARIMA considerata, il modello è stato stimato utilizzando la classe SARIMAX di `statsmodels`, senza includere termini di trend esplicativi (`trend='n'`) e disabilitando i vincoli di stazionarietà e invertibilità (`enforce_stationarity=False`, `enforce_invertibility=False`), al fine di evitare forzature numeriche e consentire al modello di adattarsi liberamente alla struttura dei dati. Una volta completato il fitting sul training set, ciascun modello ha generato una previsione multi-step pari alla lunghezza del test set mediante il metodo `get_forecast`, simulando una reale previsione operativa priva di aggiornamenti intermedi con dati osservati.

Le prestazioni predittive sono state valutate confrontando le previsioni con i valori reali del test set attraverso metriche complementari: **R²**, **RMSE** e **MAPE**, tutte calcolate in regime out-of-sample. In parallelo, per ciascun modello è stato registrato anche il valore dell'**Akaike Information Criterion** (AIC), utilizzato come indicatore di bontà statistica in-sample e di complessità del modello. Questo approccio ha permesso di distinguere chiaramente tra qualità di adattamento ai dati storici (AIC) e capacità di generalizzazione predittiva (metriche sul test), evitando bias di valutazione e consentendo un confronto oggettivo e sistematico tra modelli alternativi.

Di seguito il codice:

```

# Addestramento
temp_model = SARIMAX(train['total_flows'],
                     order=cand['order'],
                     seasonal_order=cand['seas'],
                     trend='n',
                     enforce_stationarity=False,
                     enforce_invertibility=False)
temp_res = temp_model.fit(disp=False)

# Previsione e Metriche
forecast = temp_res.get_forecast(steps=len(target_test))
pred = forecast.predicted_mean.clip(lower=0)

r2 = r2_score(target_test, pred)
rmse = np.sqrt(mean_squared_error(target_test, pred))
mape = np.mean(np.abs((target_test - pred) / target_test)) * 100

```

Figura 5.21: Codice per la valutazione dei modelli

I risultati sono invece i seguenti:

--- TABELLA DI CONFRONTO FINALE ---					
	Modello	AIC	R2	MAPE	RMSE
0	1. Baseline	22964.31	0.4284	7.05%	942968.0
1	2. Test AR	22956.28	0.4089	7.18%	958893.0
2	3. Test SAR	22965.93	0.4304	7.06%	941272.0
3	4. Solo d=1	22961.29	-0.0090	9.29%	1252846.0
4	5. Solo D=1	23003.95	0.3160	7.68%	1031507.0

Figura 5.22: Risultati dei modelli testati

Commentando:

- **Modello baseline:** Come ci si aspettava è il modello che ha le statistiche migliori complessivamente. Il modello rappresenta un compromesso efficace tra semplicità strutturale e capacità descrittiva della serie. La presenza della doppia differenziazione, ordinaria e stagionale, consente di rimuovere sia il trend di breve periodo sia la stagionalità giornaliera, rendendo la serie stazionaria senza introdurre eccessiva complessità. Questo si riflette in un RMSE relativamente contenuto per quello che è l'ordine di grandezza del problema, poiché l'errore non viene amplificato lungo l'orizzonte di previsione. Il MAPE relativamente basso indica che il modello riesce a mantenere una buona coerenza sul livello medio dei flussi, evitando errori percentuali sistematici. Il valore di R² pari a circa 0.43 evidenzia una capacità moderata di spiegare la variabilità osservata nel periodo di test, coerente con un modello puramente endogeno che

cattura la struttura regolare del traffico ma non le sue fluttuazioni più imprevedibili. L'AIC risulta competitivo, confermando l'efficienza del modello in relazione al numero ridotto di parametri stimati.

- **Test AR:** L'introduzione di un termine autoregressivo non stagionale aumenta la dipendenza del modello dai valori passati della serie. Questo porta a una leggera riduzione dell'AIC, segnale di un miglior adattamento in fase di stima, ma non si traduce in un miglioramento delle prestazioni predittive. Il RMSE più elevato rispetto al modello precedente indica che l'inerzia introdotta dal termine AR tende a propagare l'errore nel forecast multi-step. Analogamente, il MAPE leggermente superiore suggerisce una maggiore difficoltà nel mantenere una stima accurata del livello dei flussi nel periodo di test. Il R^2 inferiore conferma che l'aumento di complessità non migliora la capacità esplicativa out-of-sample, ma introduce una dinamica più rigida che penalizza la generalizzazione.
- **Test SAR:** Questo modello introduce un termine autoregressivo stagionale, che consente di catturare una memoria più lunga nella dinamica giornaliera del traffico. Tale scelta produce un RMSE leggermente inferiore rispetto al baseline, indicando una migliore capacità di seguire l'andamento medio stagionale. Il MAPE rimane sostanzialmente invariato, segno che il modello non altera significativamente la stima del livello complessivo. Il R^2 risulta marginalmente più alto, suggerendo un lieve miglioramento nella spiegazione della variabilità osservata. Tuttavia, l'AIC più elevato indica che il guadagno in termini di accuratezza non compensa completamente l'aumento di complessità del modello, rendendolo meno parsimonioso rispetto alla soluzione baseline.
- **Solo $d = 1$:** L'assenza della differenziazione stagionale rende il modello incapace di rimuovere correttamente la forte periodicità giornaliera della serie. Questo porta a una drastica perdita di capacità predittiva, evi-

dente nel RMSE molto elevato e nel MAPE significativamente superiore rispetto agli altri modelli. Il R^2 negativo indica che il modello risulta meno informativo di una previsione basata sulla media del training set, segnalando una specificazione inadeguata. Sebbene l'AIC non sia estremamente penalizzante, le metriche di errore mostrano chiaramente che il modello non è in grado di descrivere in modo coerente la struttura del traffico orario.

- **Solo $D = 1$:** In questo caso la rimozione della sola stagionalità giornaliera, senza differenziazione ordinaria, lascia residui di non stazionarietà nel livello della serie. Il modello riesce parzialmente a catturare la struttura ciclica, ma mostra una maggiore variabilità dell'errore, come evidenziato da un RMSE superiore rispetto ai modelli migliori. Il MAPE più alto indica una minore precisione relativa, mentre il R^2 ridotto conferma una capacità limitata di spiegare la dinamica del traffico nel periodo di test. L'AIC più elevato riflette un peggior compromesso tra qualità del fit e complessità, rendendo il modello complessivamente meno competitivo.

A questo punto è chiaro che la scelta migliore che si possa fare è il modello baseline, come suggerivano già ACF e PACF.

5.7 Forecast finale e valutazione risultati

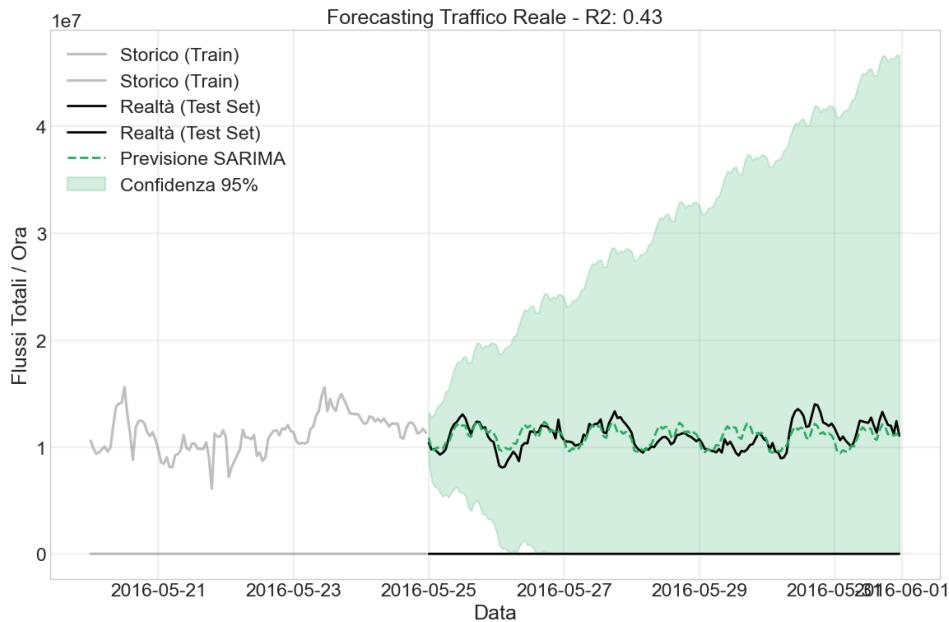


Figura 5.23: Previsione sul test set del modello scelto

Il grafico in figura illustra la sovrapposizione tra la serie temporale reale (linea nera) e la previsione generata dal modello SARIMA (linea verde tratteggiata) sul periodo di Test (ultima settimana di Maggio). L'area ombreggiata verde rappresenta l'intervallo di confidenza al 95%.

Dall'ispezione visiva emergono quattro dinamiche fondamentali che caratterizzano il comportamento del modello:

- La previsione si allinea perfettamente con la ciclicità della serie reale. I picchi diurni e le valli notturne della linea verde coincidono temporalmente con quelli della linea nera.
- Il grafico evidenzia il difetto principale del modello univariato: una sovrastima sistematica del traffico. Il modello proietta sul weekend lo stesso profilo di carico dei giorni feriali. Infatti, essendo il modello cieco al calendario, basa la sua previsione di "domani" su "ieri". Poiché Venerdì era

un giorno ad alto traffico, il modello assume erroneamente che anche Sabato lo sarà. La stagionalità a 24 ore non è sufficiente a catturare il ciclo settimanale (che richiederebbe $s = 168$, computazionalmente oneroso).

- La linea verde appare molto più "liscia" e regolare rispetto alla nera. Mentre la realtà presenta una "rugosità" continua (varianza ad alta frequenza dovuta alla natura stocastica delle connessioni TCP/UDP), il modello traccia il comportamento medio atteso. Gli errori locali si manifestano principalmente nei picchi e nei minimi dei flussi orari. Ad esempio, durante i momenti di traffico particolarmente intenso o scarso, la previsione non raggiunge i valori massimi osservati né scende ai minimi reali, mostrando un ritardo o una sottostima delle variazioni più repentine. Questo fenomeno è tipico dei modelli puramente endogeni come SARIMA, che si basano esclusivamente sulle dinamiche interne della serie storica e non considerano fattori esogeni, come eventi straordinari, condizioni meteorologiche o variazioni nella domanda che possono generare shock improvvisi o anomalie temporanee. Questo spiega la discrepanza tra l'ottimo MAPE (7%, errore sulla media) e il moderato R² (0.43, errore sulla varianza).
- La banda di confidenza al 95% è molto ampia, soprattutto sugli orizzonti più lontani. Questo riflette l'incertezza intrinseca del modello in previsioni multi-step su serie altamente volatili, come i flussi orari, e anche questo giustifica perché il R² non è elevato (0.43).

La figura seguente mostra l'analisi dei residui del modello:

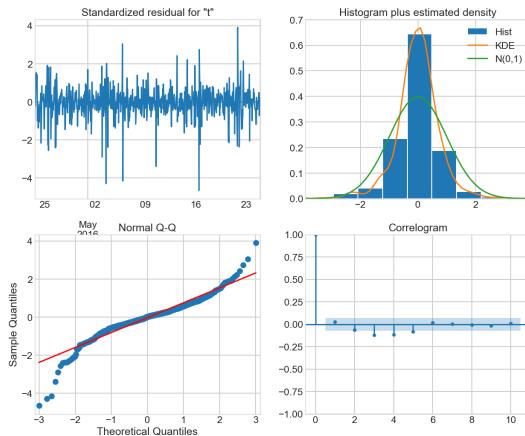


Figura 5.24: Analisi dei residui

L’analisi congiunta di questi quattro pannelli permette di verificare il rispetto delle assunzioni di base del modello (indipendenza, stazionarietà e normalità dei residui). Si nota che i residui oscillano attorno allo zero con una varianza che appare costante per la maggior parte del tempo, senza mostrare trend o pattern ciclici evidenti. Questo indica che il modello ha catturato con successo sia il trend che la stagionalità. Si osservano tuttavia sporadici picchi verticali che superano le soglie standard. In un contesto di cybersecurity, questi non sono difetti del modello, ma rappresentano traffico anomalo (attacchi o altri eventi) che il modello correttamente non è riuscito a prevedere, isolandoli come segnale di allarme.

La distribuzione è poi centrata sullo zero (assenza di bias), ma presenta una forma più appuntita della normale e code pesanti. La deviazione dalla normalità perfetta è attesa e fisiologica nei dati di rete reali. Le code lunghe confermano la presenza di outlier estremi, che allontanano la distribuzione dalla gaussiana pura.

Nel terzo grafico, la maggior parte dei punti giace perfettamente sulla linea rossa a 45 gradi, indicando che per il "traffico normale" gli errori seguono una distribuzione gaussiana. Alle estremità (in basso a sinistra e in alto a destra), i punti divergono nettamente dalla linea rossa. Questa è la conferma visiva delle "code pesanti".

Nell’ultimo grafico, quasi tutte le barre cadono all’interno dell’area ombreggiata blu (intervallo di confidenza al 95%). Questo indica che non c’è autocorrelazione significativa residua. Il modello SARIMA ha estratto tutta la struttura temporale (dipendenze passate e cicli) dai dati. Ciò che rimane è assimilabile a Rumore Bianco.

5.8 Inserimento variabili esogene

Al fine di migliorare la capacità predittiva del modello e incorporare informazioni strutturali non direttamente catturabili dalla dinamica autoregressiva, è stata introdotta una variabile esogena binaria denominata `is_weekend`. Tale variabile codifica esplicitamente la distinzione tra giorni feriali e fine settimana, assumendo valore 1 per le osservazioni corrispondenti a sabato e domenica e 0 negli altri casi. Questa scelta è motivata dall’evidenza empirica, osservata nelle fasi esplorative precedenti, di una riduzione sistematica del traffico durante il weekend, difficilmente modellabile esclusivamente tramite componenti stagionali regolari.

La variabile esogena è stata costruita direttamente a partire dall’indice temporale della serie ed è stata generata sia per il periodo di addestramento sia per l’orizzonte di test, garantendo coerenza temporale tra input esogeni e osservazioni previste. In questo modo, il modello SARIMAX è posto nelle condizioni di sfruttare un’informazione nota anche nel futuro, evitando qualsiasi forma di leakage informativo.

Il modello finale mantiene la stessa struttura del baseline individuato in precedenza, ovvero un SARIMA $(0, 1, 1)(0, 1, 1, 24)$, al quale viene aggiunto il regressore esogeno `is_weekend`. L’obiettivo non è alterare la dinamica interna del modello, già ritenuta adeguata, ma correggere in modo esplicito uno specifico effetto strutturale ricorrente.

--- Analisi Impatto Variabile Esogena ---						
	coef	std err	z	P> z	[0.025	0.975]
is_weekend	-8.924e+04	3.5e-11	-2.55e+15	0.000	-8.92e+04	-8.92e+04
ma_L1	-0.2996	0.032	-9.423	0.000	-0.362	-0.237
ma_S.L24	-0.8865	0.026	-30.477	0.000	-0.858	-0.755
sigma2	1.506e+12	6.34e-15	2.38e+26	0.000	1.51e+12	1.51e+12

Figura 5.25: Impatto statistico variabile esogena

L’analisi dei coefficienti stimati evidenzia un termine associato alla variabile `is_weekend` fortemente significativo dal punto di vista statistico. Il coefficiente negativo indica che, a parità di altre condizioni, le ore appartenenti al fine settimana sono associate a una riduzione sistematica del traffico orario medio. Questo risultato è coerente con l’interpretazione del fenomeno osservato e conferma che il modello riesce a isolare e quantificare l’effetto weekend in modo esplicito, anziché assorbirlo implicitamente nelle componenti di errore o stagionalità.

```
--- RISULTATI SARIMAX (CON ESOGENE) ---
R2 Score: 0.4416 (Prima era: 0.4284)
RMSE:      932029.09 (Prima era: 942967.58)
MAPE:      6.95% (Prima era: 7.05%)
AIC:       22966.51
```

Figura 5.26: Risultati modello SARIMAX con variabile esogena

Dal punto di vista delle prestazioni predittive sul periodo di test, l’introduzione della variabile esogena produce un miglioramento misurabile ma contenuto. Il valore di R^2 **aumenta** da 0.428 a 0.442, indicando una maggiore capacità del modello di spiegare la variabilità osservata nei dati fuori campione. Allo stesso tempo, si osserva una **riduzione** dell’RMSE e del MAPE, segnale di una previsione mediamente più **accurata** sia in termini assoluti sia relativi. Questi miglioramenti suggeriscono che la correzione esplicita dell’effetto weekend consente al modello di ridurre errori sistematici localizzati, in particolare nelle giornate di sabato e domenica.

Tuttavia, l’entità del miglioramento resta complessivamente limitata. Ciò è spiegabile dal fatto che una parte significativa dell’effetto weekend era già

implicitamente catturata dalla stagionalità giornaliera e dalla struttura autoregressiva del modello baseline. In altre parole, il SARIMA senza esogene era già in grado di rappresentare una parte consistente della differenza tra giorni feriali e festivi, sebbene in modo meno diretto. L'introduzione della variabile `is_weekend` agisce quindi come una correzione di fino, migliorando l'allineamento del livello previsto ma senza modificare radicalmente la qualità complessiva delle previsioni.

Questa dinamica è confermata anche dall'analisi dell'AIC, che mostra un valore leggermente più alto rispetto al modello baseline. L'aumento, seppur contenuto, riflette la penalizzazione dovuta all'aggiunta di un parametro extra, a fronte di un guadagno predittivo moderato. Ne consegue che il modello con variabile esogena risulta leggermente meno parsimonioso, pur offrendo una rappresentazione più interpretativa del fenomeno.

In sintesi, l'inserimento della variabile esogena `is_weekend` consente di rendere il modello più aderente alla struttura reale del traffico, migliorando la leggibilità e la coerenza interpretativa delle previsioni. Il miglioramento quantitativo delle metriche è presente ma non drastico, confermando che il limite principale delle prestazioni non risiede esclusivamente nell'assenza di informazioni strutturali, bensì nella natura intrinsecamente variabile e rumorosa del traffico di rete osservato.

Graficamente la previsione è la seguente:

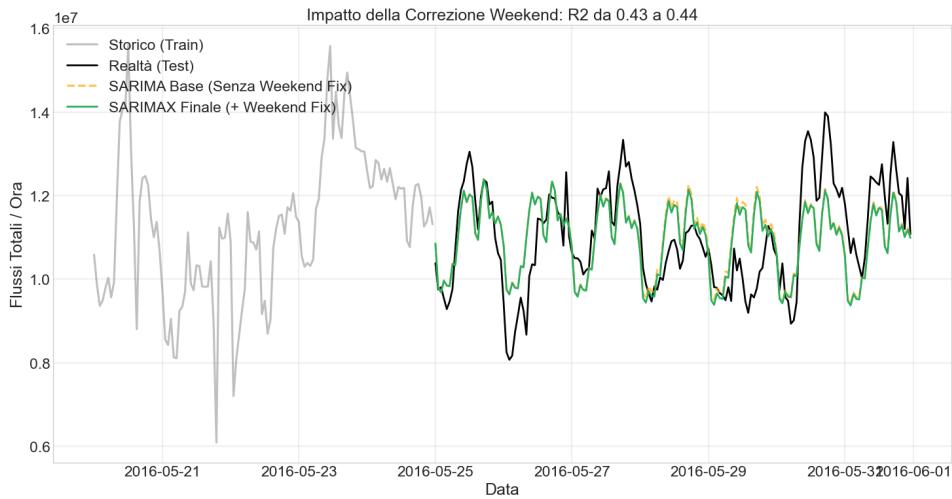


Figura 5.27: Forecast SARIMAX con esogena

La figura mostra un confronto diretto tra il modello SARIMA baseline e il modello SARIMAX arricchito con la variabile esogena `is_weekend`, limitatamente all’orizzonte di test (ultima settimana), includendo anche un breve tratto finale dello storico di training per continuità visiva.

La linea nera rappresenta il traffico reale osservato nel periodo di test, mentre le due curve di previsione evidenziano rispettivamente il comportamento del modello senza correzione esplicita del weekend (linea tratteggiata arancione) e del modello con variabile esogena (linea verde).

Dal confronto emerge che la struttura generale delle due previsioni è molto simile. Entrambi i modelli riproducono correttamente la stagionalità giornaliera dominante, con cicli regolari e ampiezza comparabile. Questo conferma che la componente stagionale a 24 ore è già ben catturata dal modello SARIMA di base e rappresenta il principale contributo alla forma della previsione.

Tuttavia, osservando con maggiore attenzione i periodi corrispondenti al fine settimana, si nota che il modello SARIMAX tende a produrre livelli medi leggermente più bassi rispetto al SARIMA baseline. Questa differenza, sebbene contenuta, è coerente con il segno e il significato del coefficiente stimato per la variabile introdotta e rappresenta l’effetto correttivo introdotto dall’in-

formazione esogena. In altre parole, il modello con esogene riesce a ridurre parzialmente la sovrastima sistematica del traffico nei giorni non lavorativi.

La figura evidenzia anche un aspetto cruciale: gli errori residui più grandi non sono concentrati esclusivamente nei weekend, ma si manifestano anche in corrispondenza di picchi improvvisi e cali rapidi del traffico reale. Queste fluttuazioni di breve termine, probabilmente legate a eventi di rete non osservabili (variazioni di carico applicativo, comportamenti anomali degli utenti, processi automatici), non possono essere spiegate da una semplice variabile binaria come `is_weekend`. Ciò chiarisce visivamente perché il miglioramento delle metriche globali (R^2 , RMSE, MAPE) risulti presente ma non drastico.

In sintesi, il grafico conferma che l'introduzione della variabile esogena non modifica la dinamica fondamentale del modello, ma agisce come una correzione di livello locale, migliorando l'aderenza della previsione nei periodi festivi senza risolvere le discrepanze dovute a variabilità non strutturale. Questo comportamento è coerente con l'aumento moderato di R^2 (da 0.43 a 0.44) osservato quantitativamente e rafforza l'interpretazione secondo cui il limite principale del modello non è l'assenza di informazione calendariale, bensì la complessità intrinseca del traffico di rete.