



UNIVERSITÀ DI TRIESTE

Dipartimento di Matematica, Informatica e Geoscienze

Laurea Triennale in Intelligenza Artificiale e Data Analytics

TESI DI LAUREA

Quando le parole fanno male: riconoscimento tramite Machine Learning dei modelli di misoginia nei social network

Relatore:

Prof. Alejandro Garcia Rodriguez

Laureanda:

Margherita
Necchi

A. A. 2024/2025

Abstract

La crescente proliferazione di contenuti offensivi sui social media rappresenta una sfida significativa, all'interno della quale la misoginia, come forma di linguaggio d'odio rivolto alle donne in quanto tali, è diventata una problematica sempre più rilevante. La necessità di sviluppare sistemi automatici per l'identificazione di tali contenuti è cruciale per promuovere un ambiente online più sicuro e rispettoso.

L'obiettivo di questa tesi è la progettazione, l'implementazione e la valutazione comparativa di diverse metodologie di Machine Learning e Deep Learning per il compito di riconoscimento automatico della misoginia in testi italiani. Il lavoro è stato condotto sul dataset di tweet fornito per il task di Automatic Misogyny Identification (AMI) della conferenza EVALITA 2020.

L'approccio metodologico adottato prevede un confronto sistematico tra due famiglie di modelli. In primo luogo, sono stati implementati e ottimizzati modelli di classificazione classici, tra cui Naive Bayes, Regressione Logistica e Support Vector Machines (SVM). Successivamente, è stata esplorata un'architettura di Deep Learning basata su Reti Neurali Convolutionali (CNN). Per quest'ultima, è stato impiegato un embedding pre-addestrato specifico per il dominio social al fine di catturare efficacemente le relazioni semantiche del testo.

I risultati evidenziano che, sebbene i modelli classici ben ottimizzati costituiscano una baseline robusta e altamente competitiva, l'architettura CNN ottiene performance superiori, dimostrando una maggiore efficacia nell'apprendere le feature gerarchiche e complesse del linguaggio d'odio.

Questo studio confronta l'efficacia di diverse generazioni di modelli di classificazione del testo, fornendo un'analisi pratica delle strategie per l'identificazione della misoginia in italiano e confermando il valore delle architetture neurali per questo compito di grande rilevanza sociale.

Indice

Abstract	i
Indice	iii
Elenco delle figure	v
Elenco delle tabelle	vii
1 Introduzione	1
2 Contesto teorico	3
2.1 Word Embedding	3
2.1.1 GloVe	3
2.2 Modelli Lineari	3
2.2.1 Regressione Logistica	3
2.2.2 Support Vector Machines (SVM) con Kernel lineare	4
2.3 Modelli probabilistici	5
2.3.1 Naive Bayes (Multinomial)	5
2.4 Modelli basati su alberi decisionionali	5
2.4.1 Decision Trees	5
2.4.2 Random Forest	6
2.5 Reti Neurali	6
2.5.1 Reti Neurali Convoluzionali (CNN)	6
2.6 Metriche di valutazione per la classificazione	7
3 Dataset e preprocessing	9
3.1 Descrizione del dataset	9
3.2 Analisi esplorativa	9
3.2.1 Statistiche generali e distribuzione delle classi	9
3.2.2 Analisi della lunghezza dei tweet	10
3.3 Pipeline di preprocessing	11
3.3.1 Pulizia dei dati	11
3.3.2 Normalizzazione	12

3.3.3	Tokenizzazione e lemmatizzazione	12
3.3.4	"Disambiguazione"	12
3.4	Analisi del dataset post-preprocessing	13
3.4.1	Nuove statistiche	13
4	Metodologia sperimentale	17
4.1	Rappresentazione delle feature	17
4.1.1	TF-IDF	17
4.1.2	Word Embedding	17
4.2	Configurazione	18
4.2.1	Modelli tradizionali	18
4.2.2	CNN	20
5	Risultati ottenuti	23
5.1	Risultati dei modelli classici	23
5.2	Risultati della CNN	25
5.3	Analisi comparativa	26
	Bibliografia	31

Elenco delle figure

2.1	Esempio preso da Soni et al. 2022 [15].	7
3.1	Word cloud delle parole più frequenti nel testo suddivise per Classe 0 (sinistra) e Classe 1 (destra).	10
3.2	Distribuzione e proporzione delle classi nel dataset originale.	10
3.3	Distribuzione della lunghezza (caratteri e parole) del testo originale, suddivisa per classe.	11
3.4	Confronto della distribuzione di lunghezza (parole e caratteri) tra il testo originale (blu) e quello processato (arancione).	13
3.5	Confronto delle 20 parole più frequenti prima e dopo il preprocessing.	14
4.1	Distribuzione delle classi nei set di Training, Validation e Test. Si noti come la proporzione tra le classi venga mantenuta in ogni suddivisione.	21
5.1	Curve ROC a confronto per i modelli di classificazione classici.	23
5.2	Andamento della funzione di perdita (Loss) e dell'accuratezza (Accuracy) durante l'addestramento della CNN ottimizzata.	25
5.3	Analisi grafica delle performance del modello CNN ottimizzato sul test set.	26

Elenco delle tabelle

3.1	Esempi di applicazione della pipeline di preprocessing.	15
4.1	Griglia degli iperparametri per la grid search	19
5.1	Performance dei modelli di classificazione classici sul test set. I valori riportati sono F1-score (macro) e AUC (Area Under the ROC Curve).	23
5.2	Configurazione degli iperparametri ottimali per i modelli classici, individuata tramite Grid Search.	24
5.3	Confronto delle performance finali tra il miglior modello classico e la CNN. .	26
5.4	Top 12 feature più importanti per il random forest.	27

Capitolo 1

Introduzione

L'avvento dell'era digitale e la pervasività dei social network hanno trasformato radicalmente le modalità di comunicazione e interazione sociale. Piattaforme come X, Facebook e Instagram sono diventate le nuove piazze globali, offrendo opportunità senza precedenti per la condivisione di informazioni e il dibattito pubblico. Tuttavia, questa rivoluzione digitale presenta anche un lato oscuro: la diffusione esponenziale di discorsi d'odio (hate speech), disinformazione e violenza verbale.

All'interno del vasto panorama dei discorsi d'odio, la misoginia online rappresenta una delle manifestazioni più dannose e pervasive. Non si tratta di semplici insulti, ma di un sistema di aggressioni, molestie e denigrazioni dirette contro le donne, che mira a sminuirle, intimidirle e a escluderle dal dibattito pubblico. Questo fenomeno ha forti conseguenze reali: normalizza la violenza di genere, rafforza stereotipi dannosi e può avere un impatto psicologico devastante sulle vittime. Contrastare la misoginia online non è quindi solo un problema tecnico, ma un'impellente necessità sociale.

In questo contesto si inserisce il presente lavoro di tesi. L'obiettivo principale è quello di esplorare, implementare e confrontare criticamente diverse metodologie di Machine Learning e Deep Learning per il compito di riconoscimento automatico della misoginia in testi italiani estratti dal social network X. Sfruttando le tecniche del Natural Language Processing (NLP), questa tesi si propone di rispondere a una domanda fondamentale: con quale efficacia e con quali limiti i modelli computazionali possono comprendere le complesse sfumature del linguaggio d'odio rivolto alle donne e assistere nella sua moderazione? Per condurre questa analisi, è stato utilizzato il dataset di riferimento del task AMI (Automatic Misogyny Identification) della conferenza EVALITA 2020.

La scelta di questo argomento non nasce solo da un interesse accademico, ma anche da una profonda consapevolezza della sua rilevanza sociale. La lotta contro la misoginia e la violenza di genere è una delle sfide più urgenti della nostra contemporaneità, un percorso che richiede l'impegno di tutte le discipline. Questo lavoro di tesi, pur nei limiti del suo campo d'azione, vuole quindi rappresentare un contributo, seppur piccolo, a questo sforzo collettivo, applicando gli strumenti del machine learning a un problema di fondamentale importanza per una società migliore. L'ambizione e la speranza sono che ogni passo avanti nella tecnologia possa in qualche modo rappresentare anche un passo verso l'estinzione delle disuguaglianze di genere.

Capitolo 2

Contesto teorico

2.1 Word Embedding

Per fornire in input i testi a modelli di deep learning come le Reti Neurali Convoluzionali (CNN), è necessario convertire le parole in una rappresentazione numerica. Un approccio basilare come il bag of words, che rappresenta i testi contando la frequenza di ogni parola, presenta due svantaggi principali: un'enorme dimensionalità e l'incapacità di catturare le relazioni di significato e l'ordine delle parole.

Per superare questi limiti si ricorre al word embedding, una tecnica che rappresenta le parole in uno spazio numerico continuo tramite vettori densi e a bassa dimensionalità. Il vantaggio che se ne ricava è duplice. In primo luogo, la dimensionalità viene ridotta drasticamente: ad esempio, con un vocabolario di 10.000 parole, un vettore bag of words avrebbe 10.000 dimensioni, mentre un word embedding ha una dimensione tipica di 100 o 300, rendendo i calcoli molto più efficienti. In secondo luogo, e più importante, questa tecnica cattura le relazioni semantiche e sintattiche tra le parole: termini con significato simile avranno infatti vettori vicini nello spazio di embedding, permettendo al modello di comprendere il contesto in modo più efficace.

2.1.1 GloVe

Per questo lavoro di tesi, si è scelto di utilizzare word embedding pre-addestrati per sfruttare la conoscenza linguistica già appresa da enormi quantità di testo. In particolare, è stato adottato il modello GloVe (Global Vectors for Word Representation) [13], sviluppato da Pennington, Socher e Manning. È stata selezionata una versione di GloVe addestrata specificatamente su miliardi di tweet: una scelta che garantisce che la rappresentazione vettoriale delle parole sia più fedele al linguaggio informale, ricco di slang e neologismi, tipico del social network analizzato.

2.2 Modelli Lineari

2.2.1 Regressione Logistica

La Regressione Logistica è un modello di classificazione binaria che ha lo scopo di predire una variabile di output discreta. Il modello utilizza la funzione logistica per calcolare la probabilità che una data osservazione appartenga a una delle due classi.

Il meccanismo centrale del modello consiste nel mappare l'output di una funzione lineare, $z = w^T x + b$, in un intervallo di probabilità $[0, 1]$ attraverso una funzione di squashing. La funzione utilizzata a tale scopo è la funzione sigmoide, definita come:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad (2.1)$$

L'output, $p(x) = \sigma(w^T x + b)$, viene interpretato come la probabilità di appartenenza alla classe positiva. La classificazione finale si ottiene applicando una soglia α (tipicamente 0.5) a tale probabilità. I parametri del modello, ovvero i pesi w e il bias b , vengono appresi minimizzando una funzione di costo. Per la regressione logistica, si utilizza la funzione di perdita cross-entropy. La formula è la seguente:

$$\mathcal{L}(w, b) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(p(x_i)) + (1 - y_i) \log(1 - p(x_i))] \quad (2.2)$$

dove y_i è l'etichetta reale (0 o 1) e $p(x_i)$ è la probabilità predetta dal modello per l' i -esima osservazione.

Una proprietà fondamentale di questa funzione di costo, nel contesto della regressione logistica, è la sua convessità rispetto ai parametri w e b . Sebbene la funzione di cross-entropy non sia convessa in generale, la sua applicazione a un modello lineare seguito da una funzione sigmoide (come nella regressione logistica) produce una superficie di costo convessa. Questo garantisce che gli algoritmi di ottimizzazione, come il gradient descent, possano convergere verso un unico minimo globale, trovando così i parametri ottimali del modello in modo affidabile.

2.2.2 Support Vector Machines (SVM) con Kernel lineare

Le Support Vector Machines (SVM) sono una classe di modelli di supervised learning il cui obiettivo è identificare l'iperpiano ottimale che separa i punti appartenenti a due classi distinte. Il principio guida di una SVM è la ricerca dell'iperpiano a margine massimo, ovvero quello che si posiziona alla massima distanza possibile dai punti più vicini di ciascuna classe.

Matematicamente, questo si traduce in un problema di ottimizzazione vincolata, in cui si cerca di minimizzare la norma del vettore dei pesi (equivalente a massimizzare il margine), sotto il vincolo che tutti i punti siano classificati correttamente:

$$\min_{w, b} \|w\|_2 = \min_{w, b} w^T w \quad \text{soggetto a} \quad y_i(w^T x_i + b) \geq 0, \quad \forall i \quad (2.3)$$

dove w e b definiscono l'iperpiano e (x_i, y_i) sono i punti del dataset con le loro etichette $y_i \in \{-1, 1\}$.

Per risolvere questo problema vincolato, si utilizza il metodo dei moltiplicatori di Lagrange. Questa tecnica trasforma il problema originale in una sua forma "duale", introducendo un moltiplicatore α_i per ogni punto del dataset. La soluzione del problema duale ha una proprietà fondamentale: i moltiplicatori α_i risultano essere diversi da zero esclusivamente per i punti che si trovano esattamente sul margine. Questi punti sono i vettori di supporto (support vectors) e sono gli unici a determinare la posizione dell'iperpiano, da cui il nome del modello.

Il modello adottato in questo lavoro di tesi è una Support Vector Machine con Kernel Lineare. Questo approccio è particolarmente indicato per i compiti di classificazione del testo.

L'elevata dimensionalità infatti che caratterizza i dati testuali una volta vettorizzati, implica sovente che le classi siano già linearmente separabili senza la necessità di ricorrere a trasformazioni non lineari garantite invece da altre tipologie di kernel.

2.3 Modelli probabilistici

2.3.1 Naive Bayes (Multinomial)

Il classificatore Naive Bayes è un modello di apprendimento supervisionato di tipo probabilistico, il cui funzionamento si basa sull'applicazione del Teorema di Bayes. L'obiettivo del modello è calcolare la probabilità che un dato campione, descritto da un vettore di feature x , appartenga a una determinata classe c . Matematicamente, questo viene espresso come:

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \quad (2.4)$$

dove $P(c|x)$ è la probabilità a posteriori, $P(x|c)$ è la verosimiglianza (likelihood), $P(c)$ è la probabilità a priori della classe e $P(x)$ è l'evidenza.

La caratteristica fondamentale e distintiva di questo algoritmo è l'assunzione naive che tutte le feature $x_1, x_2, \dots, x_n$ siano tra loro condizionalmente indipendenti, data la classe di appartenenza c . Questa assunzione permette di semplificare notevolmente il calcolo della verosimiglianza $P(x|c)$, che diventa il prodotto delle probabilità delle singole feature:

$$P(x|c) = P(x_1|c)P(x_2|c) \dots P(x_n|c) = \prod_{i=1}^n P(x_i|c) \quad (2.5)$$

Nel contesto della classificazione del testo, ciò significa che il modello assume che la presenza di una parola in una frase sia un evento indipendente dalla presenza di qualsiasi altra. Sebbene questa sia una forte semplificazione della realtà linguistica, permette al modello di essere computazionalmente molto efficiente.

Esistono diverse varianti del modello, ognuna adatta a un diverso tipo di dati. Per i compiti di classificazione del testo, dove le feature rappresentano la frequenza delle parole, la variante più appropriata è la Multinomial Naive Bayes, impiegata in questo studio. Il modello calcola la probabilità a posteriori per ogni possibile classe e sceglie quella che la massimizza. La regola di decisione, nota come stima a massimo a posteriori (MAP), è quindi:

$$\hat{y} = \arg \max_c P(c) \prod_{i=1}^n P(x_i|c) \quad (2.6)$$

Si noti che il denominatore $P(x)$ viene omissso, poiché è costante per tutte le classi e non influenza la scelta della classe con la probabilità più alta.

2.4 Modelli basati su alberi decisionionali

2.4.1 Decision Trees

I Decision Trees sono modelli di apprendimento supervisionato estremamente apprezzati per la loro elevata interpretabilità. La loro struttura gerarchica permette di rappresentare un insieme di decisioni e le loro possibili conseguenze in un formato facilmente comprensibile, simile a un diagramma di flusso. In un albero decisionale, ogni nodo interno rappresenta un

test su un attributo (feature), ogni ramo rappresenta un possibile esito del test, e ogni nodo foglia corrisponde a un'etichetta di classe.

La costruzione di un albero avviene tipicamente attraverso un processo ricorsivo top-down, partendo da un nodo radice che contiene l'intero dataset. Ad ogni passo, l'algoritmo deve selezionare l'attributo migliore su cui effettuare una suddivisione. Il criterio di selezione si basa sulla capacità di un attributo di creare sottoinsiemi di dati il più possibile puri, ovvero contenenti campioni appartenenti prevalentemente a una singola classe.

Per misurare la purezza di un nodo si utilizzano diverse metriche. Una delle più comuni è l'entropia, che misura il grado di disordine o incertezza di un insieme di dati S . Per un problema di classificazione con k classi, l'entropia è definita come:

$$E(S) = \sum_{i=1}^k -p_i \log_2(p_i) \quad (2.7)$$

dove p_i è la proporzione di campioni appartenenti alla classe i nel nodo S . L'algoritmo sceglie quindi l'attributo che massimizza l'information gain, ovvero la riduzione di entropia che si ottiene dopo aver effettuato la suddivisione.

Un albero lasciato crescere senza vincoli tende a diventare eccessivamente complesso e a memorizzare il rumore presente nei dati di addestramento, un fenomeno noto come overfitting. Per evitare ciò, si utilizzano tecniche di pruning (potatura in italiano).

2.4.2 Random Forest

Il Random Forest è un modello di apprendimento supervisionato che rientra nella categoria degli ensemble learning methods. Invece di basarsi su un singolo modello, un Random Forest costruisce una moltitudine di alberi decisionali durante la fase di addestramento e combina le loro predizioni per ottenere un risultato finale più accurato e stabile. L'obiettivo principale di questa tecnica è ridurre la varianza e, di conseguenza, il rischio di overfitting a cui un singolo albero decisionale è soggetto.

La forza e la robustezza del modello derivano dall'introduzione di casualità nel processo di costruzione della foresta.

2.5 Reti Neurali

2.5.1 Reti Neurali Convoluzionali (CNN)

Benché originariamente concepite per l'analisi di immagini, le Reti Neurali Convoluzionali (CNN) sono state adattate con notevole successo anche per compiti di Natural Language Processing, inclusa la classificazione del testo [15]. L'intuizione chiave consiste nel trattare la sequenza di word embedding di un testo come una matrice a una dimensione. Su questa matrice operano dei filtri (o kernel) che scorrono lungo la sequenza per identificare pattern locali, agendo di fatto come rilevatori di n-grammi o frasi significative. Attraverso strati successivi, la CNN è in grado di comporre queste feature di basso livello in rappresentazioni semantiche più complesse.

L'architettura di una CNN per la classificazione del testo si compone tipicamente di diversi strati sequenziali:

1. **Strato di Embedding:** L'input testuale viene prima convertito in una sequenza di word embedding. Questo trasforma il testo in una matrice numerica $X \in \mathbb{R}^{L \times d}$, dove

L è la lunghezza della sequenza e d è la dimensione degli embedding. Questa matrice è l'input effettivo per gli strati successivi.

2. **Strato Convoluzionale ('Conv1D'):** Questo è il cuore del modello. A questo strato vengono applicati dei kernel $w \in \mathbb{R}^{h \times d}$, dove h è la dimensione della finestra (es. 3, 4 o 5 parole). Ogni filtro scorre lungo la sequenza di embedding per individuare pattern locali, agendo come un rilevatore di n-grammi. Il risultato dell'applicazione di un filtro è una feature map c . Ciascun elemento c_i della feature map è calcolato come:

$$c_i = f(w \cdot x_{i:i+h-1} + b) \quad (2.8)$$

dove f è una funzione di attivazione non lineare (come ReLU), $x_{i:i+h-1}$ è la sotto-matrice di embedding su cui il filtro sta operando, e b è un termine di bias.

3. **Strato di Pooling ('MaxPooling'):** Dopo la convoluzione, uno strato di pooling riduce la dimensionalità delle feature map. Una tecnica molto comune nell'NLP è il global max pooling, che estrae il singolo valore massimo da un'intera feature map $c = [c_1, c_2, \dots, c_m]$. L'operazione è definita come:

$$\hat{c} = \max\{c_1, c_2, \dots, c_m\} \quad (2.9)$$

Questa operazione permette di dare robustezza geometrica rispetto a deformazioni relative a una gerarchia di pattern, mantenendo solo le caratteristiche più rilevanti.

4. **Strati Fully-Connected e di Output:** I valori estratti da tutti i filtri, $\hat{c}$, vengono concatenati in un unico vettore. Questo vettore viene infine passato a uno o più strati densi (fully-connected) che effettuano la classificazione finale. Per un problema binario, l'output è un singolo neurone con funzione di attivazione sigmoide, che calcola la probabilità finale p :

$$p = \sigma(W_f \cdot z + b_f) \quad (2.10)$$

dove z è il vettore concatenato proveniente dal pooling e W_f, b_f sono i pesi e il bias dello strato di output.

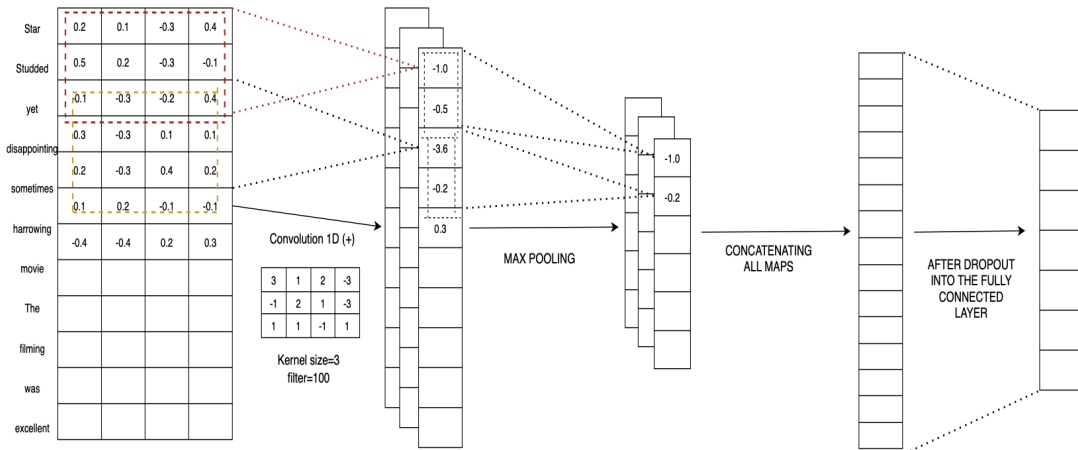


Figura 2.1: Esempio preso da Soni et al. 2022 [15].

2.6 Metriche di valutazione per la classificazione

Per valutare e confrontare in modo oggettivo le performance dei modelli di classificazione sviluppati, è indispensabile utilizzare un insieme di metriche quantitative standard. Ciascuna

metrica offre una prospettiva diversa sulla capacità del classificatore di distinguere tra le classi. Le metriche fondamentali, utilizzate in questo studio, si basano sui quattro possibili esiti di una predizione binaria:

- **Veri Positivi (True Positives, TP):** Campioni di classe positiva classificati correttamente.
- **Veri Negativi (True Negatives, TN):** Campioni di classe negativa classificati correttamente.
- **Falsi Positivi (False Positives, FP):** Campioni di classe negativa classificati erroneamente come positivi (Errore di Tipo I).
- **Falsi Negativi (False Negatives, FN):** Campioni di classe positiva classificati erroneamente come negativi (Errore di Tipo II).

Questi valori sono tipicamente riassunti in una matrice di confusione e costituiscono la base per il calcolo delle seguenti metriche di valutazione:

1. **Accuracy:** Rappresenta la frazione di predizioni corrette sul totale delle predizioni effettuate. È la metrica più intuitiva, ma può essere fuorviante in presenza di dataset sbilanciati, dove un modello potrebbe ottenere un'alta accuratezza semplicemente predicendo sempre la classe maggioritaria. È definita come:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.11)$$

2. **Precision:** Misura la proporzione di predizioni positive che erano effettivamente corrette. Risponde alla domanda: "Di tutti i campioni che il modello ha identificato come misogini, quanti lo erano veramente?". Un'alta precisione indica un basso numero di falsi positivi.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.12)$$

3. **Recall:** Misura la proporzione di campioni positivi reali che sono stati identificati correttamente. Risponde alla domanda: "Di tutti i tweet realmente misogini, quanti ne ha trovati il modello?". Un'alta recall indica un basso numero di falsi negativi.

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.13)$$

4. **F1-Score:** È la media armonica di precision e recall, e fornisce un singolo valore che bilancia entrambe le metriche. È particolarmente utile in contesti con classi sbilanciate, poiché un F1-Score elevato richiede che sia la precisione sia il richiamo siano alti. Per questo motivo, è stata scelta come metrica di riferimento per l'ottimizzazione e la valutazione dei modelli in questa tesi.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \quad (2.14)$$

Capitolo 3

Dataset e preprocessing

3.1 Descrizione del dataset

Il dataset utilizzato in questo lavoro di tesi, come già introdotto, è stato fornito per il task di Automatic Misogyny Identification (AMI) della conferenza EVALITA 2020 [4]. Si tratta di una raccolta di testi in lingua italiana estratti dal social network X (precedentemente Twitter), che sono stati manualmente etichettati per il compito specifico di riconoscimento del linguaggio misogino.

Il dataset è composto da 13115 tweet in lingua italiana, che rappresentano un campione eterogeneo di linguaggio informale, spesso caratterizzato da espressioni gergali, errori di battitura e neologismi tipici della piattaforma.

Come specificato nelle linee guida del task, ogni istanza del dataset è costituita dal testo del tweet e da un'etichettatura manuale che ne definisce la natura, misogino, aggressivo, entrambi o nessuno.

Per concentrare l'analisi sul compito primario di riconoscimento della misoginia, il problema di classificazione è stato semplificato rispetto alla formulazione originale del task. Nello specifico, si è scelto di considerare unicamente l'etichetta binaria relativa alla presenza o assenza di contenuto misogino, escludendo dall'analisi la distinzione secondaria sul livello di aggressività. Questo ha permesso di focalizzare la valutazione dei modelli sulla loro capacità fondamentale di discriminare il linguaggio d'odio di genere.

3.2 Analisi esplorativa

Prima di applicare qualsiasi tecnica di preprocessing, è stata condotta un'analisi esplorativa dei dati sul dataset originale per comprenderne le caratteristiche fondamentali.

3.2.1 Statistiche generali e distribuzione delle classi

L'analisi preliminare del dataset rivela un leggero sbilanciamento nella distribuzione delle classi, come illustrato in Figura 3.2. Su un totale di 13,115 tweet, 7,653 (pari al 58.4%) sono etichettati come non misogini (Classe 0), mentre i restanti 5,462 (41.6%) sono classificati come misogini (Classe 1). Sebbene lo sbilanciamento non sia estremo, suggerisce l'importanza di utilizzare metriche di valutazione robuste, come l'F1-score, che sono meno sensibili alla distribuzione delle classi rispetto alla semplice accuracy.



Figura 3.1: Word cloud delle parole più frequenti nel testo suddivise per Classe 0 (sinistra) e Classe 1 (destra).

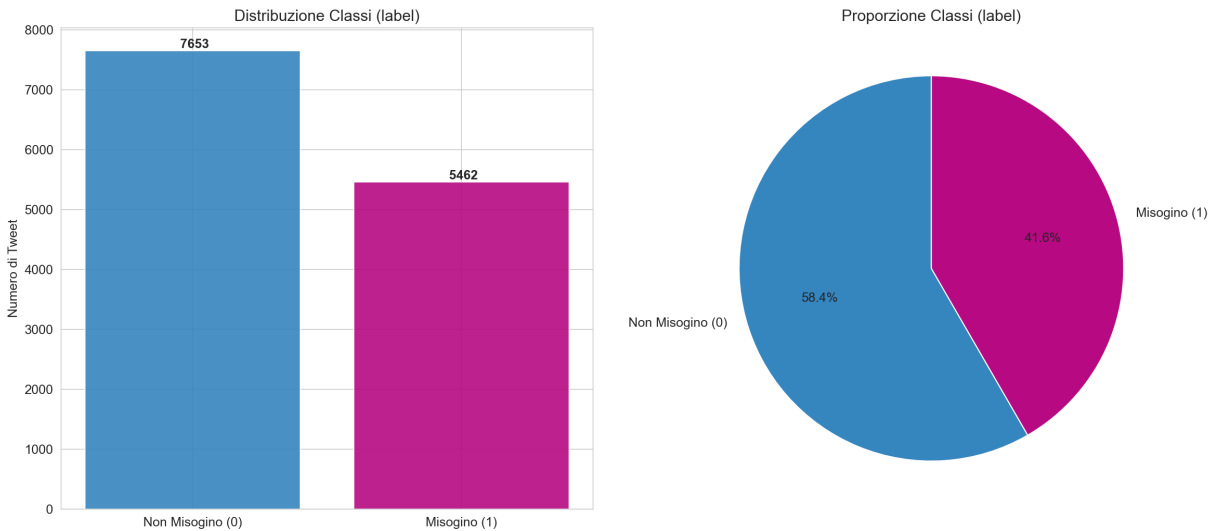


Figura 3.2: Distribuzione e proporzione delle classi nel dataset originale.

3.2.2 Analisi della lunghezza dei tweet

La distribuzione della lunghezza dei caratteri mostra una tendenza bimodale, con un picco per i tweet brevi (sotto i 100 caratteri) e un secondo picco più moderato in prossimità del limite massimo di caratteri allora consentito da X (280 caratteri). La distribuzione del numero di parole è fortemente asimmetrica, con la maggior parte dei tweet che si concentra tra le 5 e le 25 parole. In entrambi i grafici, si può notare come i tweet misogini (Classe 1) tendano ad essere mediamente più brevi, suggerendo probabilmente un linguaggio più conciso e diretto.

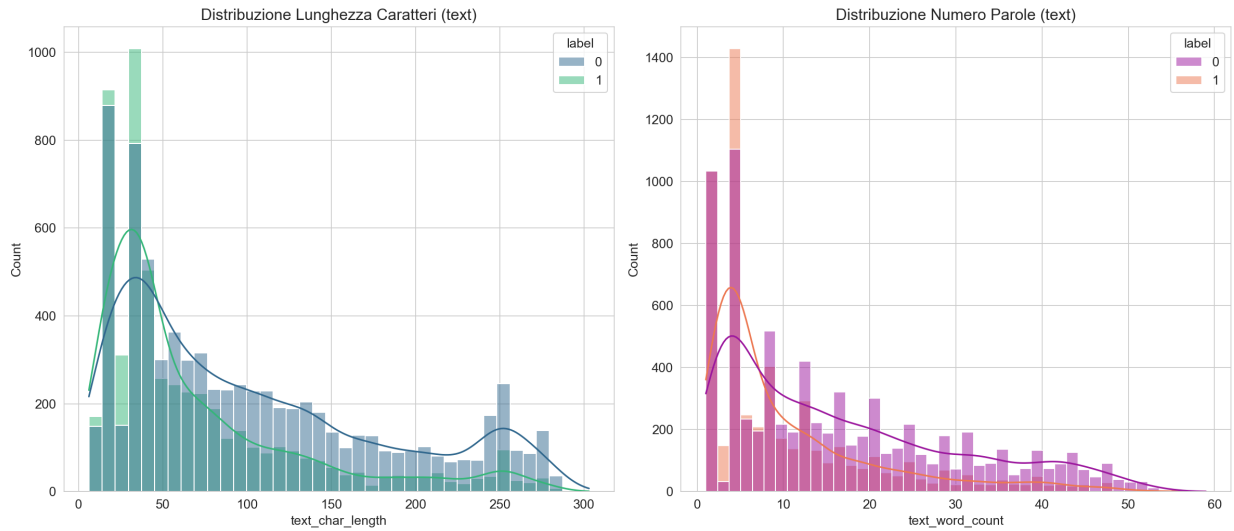


Figura 3.3: Distribuzione della lunghezza (caratteri e parole) del testo originale, suddivisa per classe.

3.3 Pipeline di preprocessing

Il testo proveniente dai social network, e da X in particolare, è generalmente caratterizzato da un'elevata "rumorosità": presenta slang, abbreviazioni, errori di battitura, emoji e elementi strutturali specifici della piattaforma come URL e menzioni. Per rendere questi dati utilizzabili dai modelli di machine learning e per ridurre la complessità del vocabolario, è stata applicata una pipeline di preprocessing articolata in più fasi. L'obiettivo è quello di normalizzare e pulire il testo, preservando al contempo le informazioni semantiche rilevanti.

3.3.1 Pulizia dei dati

La prima fase si è concentrata sulla rimozione o sulla sostituzione degli elementi che non portano un significato lessicale diretto, ma che possono introdurre rumore.

- **Rimozione di URL, menzioni e hashtag:** gli URL, i riferimenti ad altri utenti (menzioni, es. @utente) e gli hashtag (#) sono stati gestiti per semplificare il testo. Gli URL sono stati completamente rimossi, mentre le menzioni sono state sostituite con un segnaposto generico, `mention`. Questo permette di eliminare la variabilità dei nomi utente, non direttamente portatori di significato, mantenendo l'informazione che il tweet era un'interazione diretta. Gli hashtag sono stati privati del simbolo # ma la parola che li compone è stata mantenuta nel testo, poiché spesso parte integrante del messaggio.
- **Gestione delle emoji:** le emoji e le emoticon, pur veicolando un forte segnale emotivo e sentimentale, aumentano fortemente la complessità del vocabolario. Per questo motivo, sono state sostituite con un segnaposto unico, `emoji`, in modo da segnare la loro presenza senza dover gestire centinaia di simboli diversi.
- **Rimozione di caratteri speciali e punteggiatura:** la punteggiatura e i caratteri speciali (es. *, &, \$) sono stati rimossi. Una gestione particolare è stata riservata ai segni di punteggiatura che possono indicare un'enfasi emotiva, come i punti esclamativi e interrogativi, la cui presenza è stata codificata con un apposito segnaposto.

3.3.2 Normalizzazione

La seconda fase si è focalizzata sulla standardizzazione del testo rimanente per raggruppare forme diverse della stessa parola o dello stesso concetto.

- **Conversione in minuscolo:** tutto il testo è stato convertito in minuscolo. Questo passaggio è fondamentale per garantire che parole come "Donna", "donna" e "DONNA" vengano trattate come un unico token, evitando di gonfiare inutilmente il vocabolario.
- **Abbreviazioni e slang:** il linguaggio dei social media è ricco di abbreviazioni e slang (es. `xkè` per "perché"). È stato a tal fine definito un dizionario di abbreviazioni comuni e diffuse, utilizzato poi per espandere queste forme nella loro versione standard, migliorando la coerenza del testo.
- **Parole allungate:** spesso per veicolare meglio emozioni vengono utilizzate dagli utenti parole allungate (es. `ciao` per "ciao"). Queste forme colloquiali sono state normalizzate riducendo le sequenze di lettere ripetute a un massimo di due, riconducendo il termine alla sua forma corretta.

3.3.3 Tokenizzazione e lemmatizzazione

Una volta pulito e normalizzato, il testo è stato suddiviso in unità atomiche, o token, che corrispondono tipicamente a singole parole. Infine, è stato applicato un processo di lemmatizzazione. A differenza dello stemming, che si limita a tagliare la fine di una parola, la lemmatizzazione riconduce ogni parola alla sua forma base del dizionario (il "lemma"), ad esempio trasformando "andando", "andai" e "andrei" in "andare". Questo passaggio, eseguito tramite la libreria spaCy per la lingua italiana, è cruciale per ridurre drasticamente la dimensionalità del vocabolario e per raggruppare tutte le flessioni di un verbo o le declinazioni di un nome sotto un unico concetto.

3.3.4 "Disambiguazione"

Una delle sfide più complesse nel riconoscimento automatico della misoginia risiede nell'ambiguità semantica del linguaggio. L'intento misogino infatti spesso non è veicolato da parole intrinsecamente offensive, ma dall'uso di termini neutri in un contesto peggiorativo. Ad esempio, una parola come "cagna" può riferirsi alla femmina del cane in un contesto neutro, oppure essere usata come un grave insulto in un contesto d'odio. I modelli tradizionali, che si basano sulla frequenza delle parole o su embedding statici, faticano a cogliere questa differenza contestuale.

Per affrontare questo problema, è stata implementata un'ulteriore fase di preprocessing volta a disambiguare il significato di un insieme predefinito di termini ambigui, ispirandosi a recenti lavori di ricerca nel campo [muti-et al-2024-pejorativity]. L'obiettivo di questo passaggio è di arricchire il testo, sostituendo le parole ambigue con un sostituto (o "ancora") che ne espliciti il significato contestuale, al fine di fornire un input più chiaro e meno ambiguo ai successivi modelli di classificazione.

Il processo si basa su tre componenti chiave:

1. **Lessico:** è stato come prima cosa definito un lessico personalizzato che mappa una serie di parole ambigue a una coppia di "ancore" testuali. Per ogni parola, è stata definita un'ancora che ne rappresenta il senso peggiorativo e una per il senso neutro. Ad esempio, per la parola "cagna", le ancore definite sono state "insulto donna" (peggiorativo) e "cane femmina" (neutro).

2. **Modello di classificazione:** per determinare quale dei due sensi fosse appropriato in un dato tweet, è stato impiegato un modello di classificazione pre-addestrato (MoritzLaurer/mDeBERTa-v3-base-mnli-xnli), accessibile tramite la libreria Transformers. Per ogni parola ambigua trovata in un tweet, il modello riceve in input l'intero testo del tweet e le due ancore (peggiorativa e neutra) come etichette candidate. Il classificatore determina quindi quale delle due etichette/ancore è più appropriata per descrivere il contesto del tweet, restituendo il senso più probabile.
3. **Sostituzione:** una volta determinato il senso contestuale della parola tra i due forniti, la parola originale all'interno del tweet viene sostituita con la corrispondente ancora testuale. Per ragioni di efficienza computazionale, il processo di disambiguazione e sostituzione è stato applicato solo alla prima parola ambigua trovata in ogni tweet.

3.4 Analisi del dataset post-preprocessing

Dopo l'applicazione dell'intera pipeline di pulizia e arricchimento, è stata condotta una seconda fase di analisi per valutare l'impatto di tali trasformazioni sul dataset e avere una visione di insieme sui dati che verranno utilizzati.

3.4.1 Nuove statistiche

L'effetto più evidente della pipeline di preprocessing è la significativa riduzione e standardizzazione della lunghezza dei testi. La Figura 3.4 confronta la distribuzione della lunghezza dei tweet (in parole e caratteri) prima e dopo il processo di pulizia.

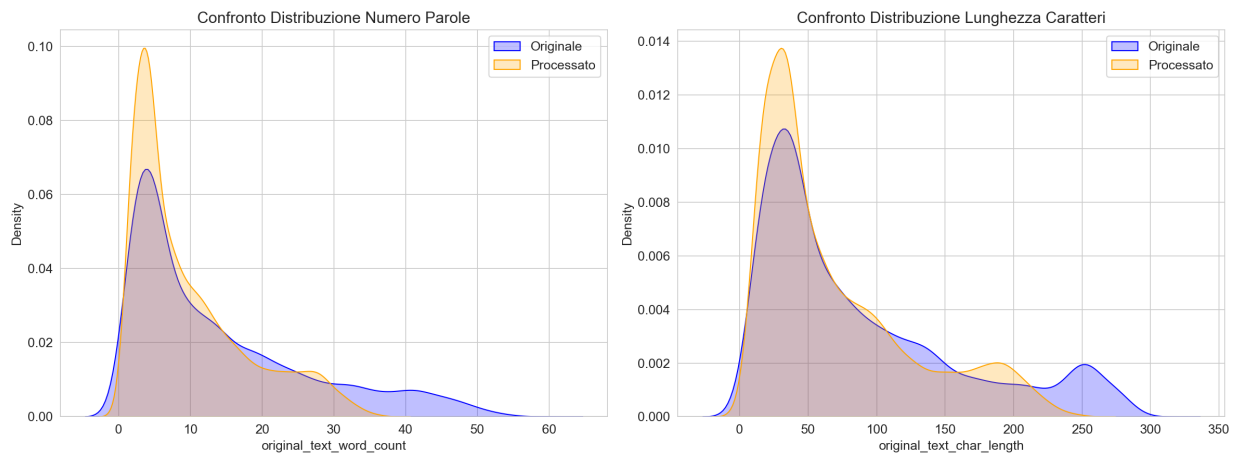


Figura 3.4: Confronto della distribuzione di lunghezza (parole e caratteri) tra il testo originale (blu) e quello processato (arancione).

Come si può osservare, la distribuzione del testo processato (in arancione) è nettamente spostata verso sinistra, con un picco più accentuato su valori di lunghezza inferiori. Questo indica che la rimozione di elementi "rumorosi" come URL, menzioni, punteggiatura e stop-words ha avuto successo nel ridurre la lunghezza dei testi, eliminando le code di tweet molto lunghi e uniformando il formato dell'input. La distribuzione, originariamente bimodale, assume ora una forma più unimodale.

Per quantificare in modo oggettivo le caratteristiche del corpus e l'impatto della pipeline di preprocessing, è stata condotta un'analisi statistica basata sulla frequenza dei termini. Per questa analisi, sono stati aggregati tutti i token provenienti dalla partizione di training del dataset, prima e dopo il preprocessing, per costruire due mappe di frequenza distinte.

Una parte fondamentale dell'analisi ha riguardato le parole fuori vocabolario (Out-of-Vocabulary, OOV), ovvero quei termini non presenti in un dizionario di riferimento. Come riferimento è stato utilizzato il vocabolario del modello linguistico **AIBERTO**, specifico per la lingua italiana [14]. L'insieme delle parole uniche di ogni corpus (originale e processato) è stato confrontato con questo vocabolario per identificare i termini OOV. Per misurarne l'impatto, sono state calcolate due metriche percentuali: la percentuale sui *tipi* (parole uniche) e quella sui *token* (occorrenze totali).

L'applicazione di questa metodologia ha prodotto i seguenti risultati. Il vocabolario del testo originale, composto da 20,007 parole uniche, si riduce a **13,342** dopo il preprocessing, con una diminuzione di circa il 33%, a testimonianza dell'efficacia della lemmatizzazione e della normalizzazione.

L'analisi OOV mostra un miglioramento ancora più netto: la percentuale di parole uniche fuori vocabolario scende dal 52.2% nel testo originale al 33.4% in quello processato. L'impatto di queste parole sul testo complessivo è ancora più significativo: la loro frequenza totale (token) passa dal 18.3% a solo il 10.8%. Questo conferma che la pipeline di pulizia ha rimosso o corretto con successo una grande quantità di termini rari, probabilmente errori di battitura o slang, rendendo il vocabolario finale più pulito e standard.

La Figura 3.5 mostra infine un confronto visivo delle 20 parole più frequenti prima e dopo il preprocessing, evidenziando come dal testo processato emergano termini con un significato più forte e potenzialmente più discriminante per il compito di classificazione.

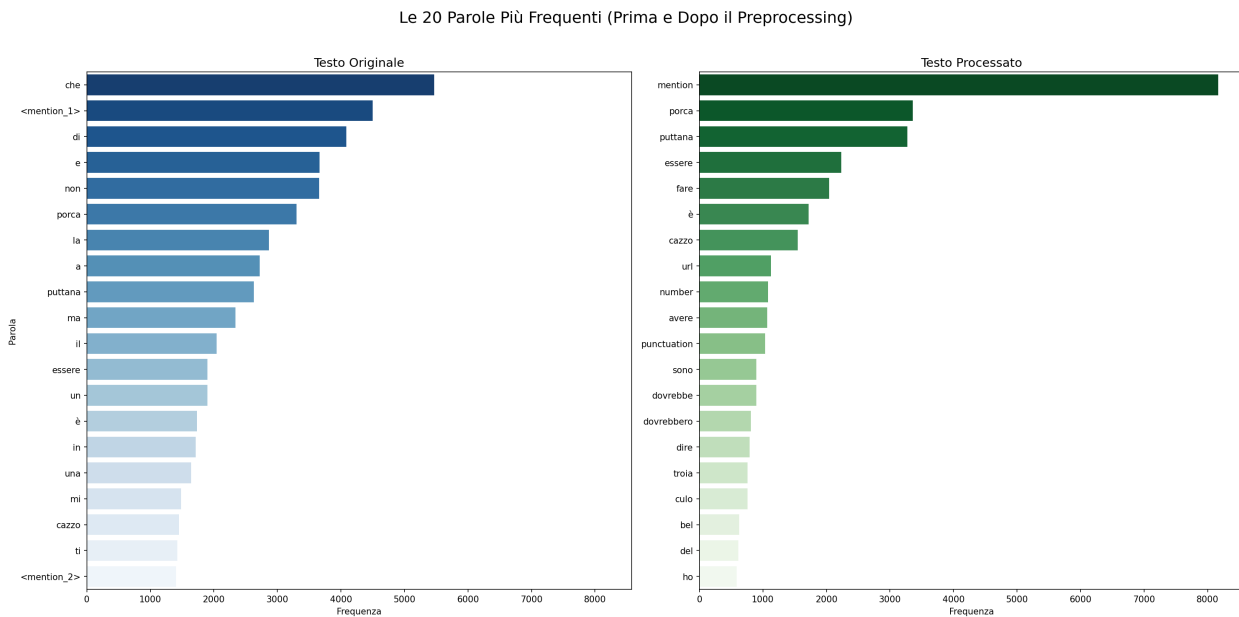


Figura 3.5: Confronto delle 20 parole più frequenti prima e dopo il preprocessing.

Per illustrare l'impatto complessivo della pipeline di preprocessing, di seguito vengono infine mostrati due esempi di tweet, uno misogino e uno non misogino, prima e dopo le trasformazioni applicate. La Tabella 3.1 riassume i passaggi chiave.

Tabella 3.1: Esempi di applicazione della pipeline di preprocessing.

Tweet Originale	Tweet Processato	Principali Trasformazioni Applicate
@Ejramruj ma statte zitta ke non capisci nnt! sei solo una cagna da sottomettere duramente	<mention_d> stare zitto capire niente essere solo insulto donna sottomettere duramente	• Normalizzazione (minuscolo, slang "ke", "nnt"). • Sostituzione di mention con tag. • Lemmatizzazione ("statte" → "stare"). • Rimozione di stopwords (es. "ma", "le", "come", ...).
CCronache dall'ascensore.Porca puttana sono bloccata aaaahh https://t.co/k74o9RHYC5	cronaca dall ascensore porca puttana essere bloccare ahh <url>	• Conversione in minuscolo. • Sostituzione url con tag. • Lemmatizzazione ("bloccata" → "bloccare").

Capitolo 4

Metodologia sperimentale

4.1 Rappresentazione delle feature

Per poter essere elaborato da un algoritmo di apprendimento automatico, un testo deve essere trasformato in una rappresentazione vettoriale numerica. In questo studio sono state adottate due strategie distinte: un approccio basato sulla frequenza dei termini per i modelli classici e uno basato su embedding semantici per l'architettura neurale.

4.1.1 TF-IDF

Per i modelli di machine learning classici (Naive Bayes, Regressione Logistica, SVM, etc.), è stata utilizzata la tecnica **TF-IDF (Term Frequency-Inverse Document Frequency)**. Questo metodo rappresenta ogni tweet come un vettore numerico in cui ogni dimensione corrisponde a una parola del vocabolario. Il valore di ogni dimensione non è un semplice conteggio, ma un peso calcolato combinando due fattori:

- **Term Frequency (TF)**: Misura la frequenza con cui un termine appare all'interno di un singolo tweet. Un termine più frequente in un documento avrà un valore TF più alto.
- **Inverse Document Frequency (IDF)**: Misura l'importanza di un termine su scala globale (l'intero dataset). Ai termini molto comuni in tutti i documenti (es. articoli, preposizioni) viene assegnato un peso IDF basso, mentre ai termini più rari e potenzialmente più informativi viene assegnato un peso più alto.

Il punteggio TF-IDF finale per una parola in un documento è il prodotto di questi due valori. Questo approccio ha il vantaggio di dare maggiore risalto alle parole che sono frequenti in un testo specifico ma rare nel resto del dataset, rendendole feature molto discriminanti per la classificazione.

4.1.2 Word Embedding

Per l'architettura di deep learning (CNN), è stata scelta una rappresentazione più semanticamente ricca basata su **word embedding** pre-addestrati. Come descritto nel capitolo precedente, questi modelli rappresentano le parole come vettori densi in uno spazio multidimensionale dove la distanza e la direzione catturano relazioni semantiche.

Come già introdotto, per questo lavoro di tesi, è stato scelto di utilizzare i vettori **GloVe** pre-addestrati.

La configurazione specifica utilizzata è la seguente:

- **Modello:** GloVe (glove.twitter.27B).
- **Dimensione dei Vettori:** 100 dimensioni.

Durante la fase di preprocessing, ogni tweet è stato convertito in una sequenza di indici numerici, dove ogni indice corrisponde a una parola nel vocabolario. Successivamente, questi indici sono stati utilizzati per recuperare i corrispondenti vettori a 100 dimensioni dalla matrice GloVe, generando così la matrice di input per la Rete Neurale Convoluzionale.

4.2 Configurazione

Vengono ora descritte le architetture e le configurazioni specifiche di tutti i modelli implementati. Per ogni modello, si è cercata la configurazione ottimale attraverso un processo di tuning degli iperparametri.

4.2.1 Modelli tradizionali

Per la fase di classificazione con approcci classici, sono stati implementati e valutati diversi modelli di machine learning, tra cui **Multinomial Naive Bayes**, **Regressione Logistica**, **Support Vector Machine (SVM) con Kernel Lineare**, **Decision Tree** e **Random Forest**.

Tutti questi modelli sono stati addestrati utilizzando come input i vettori di feature TF-IDF descritti nella sezione precedente. Per identificare la configurazione ottimale degli iperparametri per ciascun modello, è stata utilizzata una strategia di ricerca tramite Grid Search con cross validation a 5-fold. Questo processo, come già accennato in precedenza, è stato guidato dalla massimizzazione dell’F1-score.

Si trova di seguito riportata la griglia con gli iperparametri testati per ciascun modello.

Tabella 4.1: Griglia degli iperparametri per la grid search

Modello	Parametro della Pipeline	Valori Testati
MultinomialNB	<code>sel__k</code> (N. feature)	[500, 600, 700]
	<code>tfidf__use_idf</code> (Uso IDF)	True, False
	<code>clf__alpha</code> (Smoothing)	1.0, 0.1, 0.01
LogisticRegression	<code>sel__k</code> (N. feature)	[600, 700, 800]
	<code>tfidf__norm</code> (Normalizzazione)	'l1', 'l2'
	<code>clf__C</code> (Inv. regolarizzazione)	<code>logspace(-2, 2, 5)</code>
	<code>clf__penalty</code> (Tipo penalità)	['l1', 'l2']
LinearSVC	<code>sel__k</code> (N. feature)	[600, 700, 800]
	<code>tfidf__use_idf</code> (Uso IDF)	True, False
	<code>clf__C</code> (Inv. regolarizzazione)	<code>logspace(-2, 2, 5)</code>
DecisionTree	<code>sel__k</code> (N. feature)	[400, 500, 600, 700]
	<code>clf__criterion</code> (Criterio split)	['gini', 'entropy']
	<code>clf__max_depth</code> (Profondità max)	[None, 10, 20, 30]
	<code>clf__min_samples_split</code>	[2, 10, 20]
	<code>clf__min_samples_leaf</code>	[1, 5, 10]
RandomForest	<code>sel__k</code> (N. feature)	[400, 500, 600]
	<code>clf__n_estimators</code> (N. alberi)	[10, 20, 25]
	<code>clf__criterion</code> (Criterio split)	['gini', 'entropy']
	<code>clf__max_depth</code> (Profondità max)	[None, 10, 25]
	<code>clf__min_samples_split</code>	[2, 10, 20]
	<code>clf__min_samples_leaf</code>	[1, 5, 10]

C (Inverso regolarizzazione) Utilizzato in regressione logistica e SVM, questo parametro controlla la penalità associata agli errori di classificazione. Un valore di C più basso porta a un modello più semplice con un margine di separazione più ampio, mentre un valore alto spinge il modello a classificare correttamente più esempi di training, rischiando però l'overfitting.

penalty Specifico per la regressione logistica, determina il tipo di norma utilizzata per la regolarizzazione (L1 o L2). L1 implica una regolarizzazione Lasso, L2 Ridge.

k (SelectKBest) Si riferisce al numero di feature (in questo caso, parole con i punteggi TF-IDF più alti) da selezionare prima di fornirle in input al classificatore. Serve a ridurre la dimensionalità e il rumore, mantenendo solo i termini più informativi. I valori diversi dipendono dalla robustezza del modello all'overfitting e al rumore.

alpha (Smoothing) Utilizzato nel Multinomial Naive Bayes, è il parametro di smoothing. Serve a gestire parole non presenti nel set di addestramento, evitando che il modello assegni loro una probabilità pari a zero.

criterion Utilizzato in Decision Tree e Random Forest, è la funzione per misurare la qualità di una suddivisione (split). Le opzioni testate sono l'indice di impurità di Gini e l'entropia, entrambi metodi per valutare quanto un nodo sia "puro" (cioè contenga campioni di una sola classe).

max_depth La massima profondità consentita per un albero. È un iperparametro cruciale per controllare la complessità di Decision Tree e Random Forest e prevenire l'overfitting.

min_samples_split Il numero minimo di campioni richiesti per poter suddividere un nodo interno in un albero decisionale. Anche questo parametro serve a controllare la crescita dell'albero.

n_estimators Specifico per il Random Forest, rappresenta il numero totale di alberi decisionali da costruire all'interno della "foresta". Un numero maggiore di alberi generalmente porta a un modello più robusto, a fronte di un costo computazionale più elevato.

4.2.2 CNN

Per l'approccio basato sul deep learning, è stata progettata e implementata una **Rete Neuronale Convolutionale (CNN)**. È stato scelto di mantenere l'architettura del modello il più semplice possibile per evitare il rischio di overfitting. Ogni tipologia di strato è quindi presente una sola volta. L'architettura finale definita è la seguente:

1. **Strato di embedding:** Lo strato di input riceve sequenze di lunghezza fissa (pari a 35 token). Questo strato utilizza la matrice di embedding pre-addestrata **GloVe** (vettori a 100 dimensioni) per convertire i token in vettori densi. Questo strato è stato reso addestrabile (trainable), permettendo al modello di fare un fine-tuning dei vettori GloVe per adattarli specificamente al contesto della misoginia nel linguaggio italiano dei social media.
2. **Strato convoluzionale ('Conv1D'):** Uno strato convoluzionale a una dimensione con 64 filtri e una dimensione del kernel pari a 3. Questo strato è responsabile dell'estrazione di pattern locali (n-grammi) dalla sequenza di embedding, utilizzando la funzione di attivazione ReLU.
3. **Strato di pooling ('GlobalMaxPooling1D'):** Uno strato di Global Max Pooling viene applicato all'output dello strato convoluzionale. Si è deciso di optare in questo caso per un pooling globale, utile in generale per scrivere una rappresentazione invariante.
4. **Strato di regolarizzazione ('Dropout'):** Per contrastare l'overfitting, è stato inserito uno strato di dropout con un rate del 50%, che disattiva casualmente metà dei neuroni durante ogni ciclo di addestramento.
5. **Strato di output:** Lo strato finale è un singolo neurone denso con funzione di attivazione sigmoide, che produce in output una probabilità compresa tra 0 e 1, adatta per la classificazione binaria.

Per l'addestramento della rete, è stato utilizzato l'ottimizzatore Adam con un learning rate di 0.001 e la funzione di loss binary cross-entropy, specifica per problemi di classificazione binaria.

Per garantire una valutazione robusta e imparziale dei modelli, il dataset completo è stato suddiviso in tre set disgiunti:

- **Training Set (60% dei dati):** Utilizzato esclusivamente per l'addestramento dei modelli.
- **Validation Set (20% dei dati):** Utilizzato per il tuning degli iperparametri e per le decisioni intermedie durante l'addestramento (come l'early stopping per la CNN).
- **Test Set (20% dei dati):** Un set assolutamente incontaminato, usato una sola volta alla fine del processo per la valutazione finale delle performance del modello ottimizzato.

La suddivisione è stata eseguita in modo stratificato per assicurare che la proporzione tra le classi fosse mantenuta in tutti e tre i sottoinsiemi, come visualizzato in Figura 4.1.

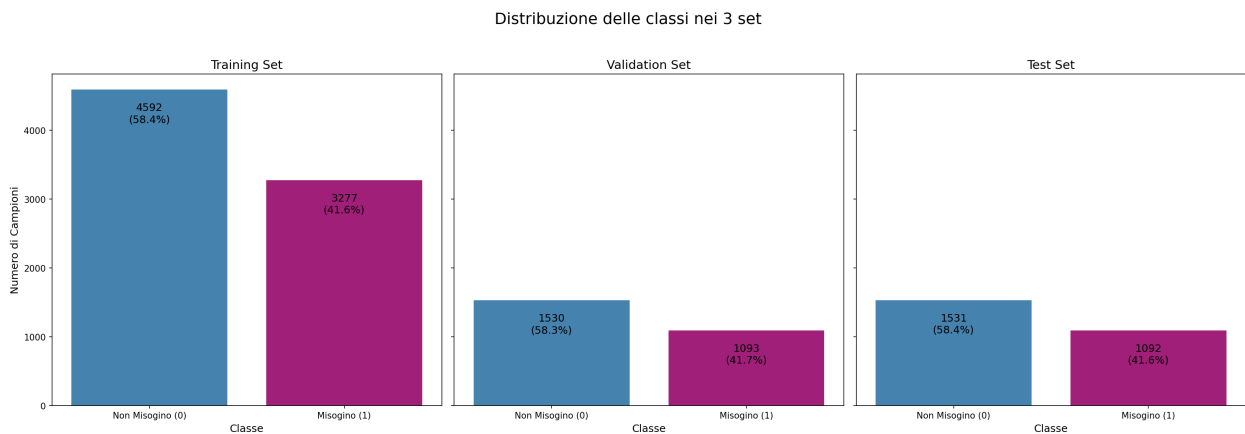


Figura 4.1: Distribuzione delle classi nei set di Training, Validation e Test. Si noti come la proporzione tra le classi venga mantenuta in ogni suddivisione.

Le strategie di validazione sono state poi adattate per le due famiglie di modelli:

- Per i modelli classici: la ricerca degli iperparametri (grid search) è stata eseguita applicando una strategia di cross validation a 5 fold sul solo training set. Il validation set non è stato usato in questa fase. Una volta identificato il modello con la configurazione migliore, la sua performance finale è stata valutata sul test set.
- Per la CNN: il modello è stato addestrato sul training set, mentre le sue performance sono state monitorate epoca per epoca sul validation set per guidare la tecnica di Early Stopping. Infine, il modello finale con i pesi migliori è stato valutato sul test set per ottenere le metriche di performance definitive.

Questo approccio a tre vie garantisce che la valutazione finale avvenga su dati che il modello non ha mai incontrato in nessuna fase del suo addestramento o della sua ottimizzazione, fornendo una stima onesta della sua capacità di generalizzazione.

Inoltre, per l'addestramento della CNN, è stata impiegata la tecnica dell'early stopping per prevenire l'overfitting e per identificare il momento ottimale in cui terminare l'addestramento. Il processo è stato monitorato tenendo traccia della funzione di perdita calcolata sul validation set.

È stata impostata una patience di 5 epoche: se la validation loss non mostrava un miglioramento (ovvero una diminuzione) per cinque epoche consecutive, l'addestramento veniva interrotto automaticamente. Inoltre, è stata attivata l'opzione per ripristinare i pesi del modello allo stato della migliore epoca (quella con la validation loss più bassa), garantendo che il modello finale salvato fosse quello con la migliore capacità di generalizzazione sui dati non visti.

Capitolo 5

Risultati ottenuti

In questo capitolo finale vengono presentati e analizzati i risultati ottenuti dai diversi modelli di classificazione implementati. La metrica principale per il confronto è l’F1-score (macro), affiancata dal punteggio AUC per valutare la capacità discriminatoria complessiva.

5.1 Risultati dei modelli classici

I modelli di machine learning tradizionali presentati in precedenza sono stati addestrati utilizzando feature TF-IDF e ottimizzati tramite Grid Search. La Tabella 5.1 riassume le performance finali ottenute da ciascun classificatore sul test set.

Tabella 5.1: Performance dei modelli di classificazione classici sul test set. I valori riportati sono F1-score (macro) e AUC (Area Under the ROC Curve).

Modello	F1-Score (macro)	AUC
Random Forest	0.8291	0.9328
Regressione Logistica	0.8046	0.9087
Support Vector Machine	0.8041	0.9065
Multinomial Naive Bayes	0.7917	0.8912
Decision Tree	0.8242	0.8946

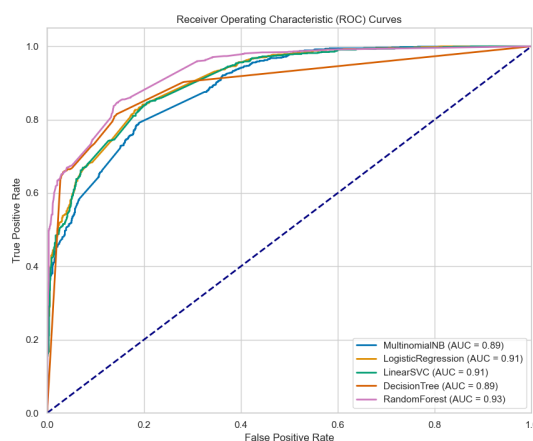


Figura 5.1: Curve ROC a confronto per i modelli di classificazione classici.

Il confronto visivo delle performance è illustrato in Figura 5.1, che mostra le curve ROC di tutti i modelli. La curva del random forest si posiziona in maniera evidente al di sopra delle altre, confermando la superiorità delinata dalle metriche precedenti.

I parametri ottimali individuati per ciascun modello tramite i quali si sono ottenuti i risultati sopra riportati sono elencati nella Tabella 5.2.

Tabella 5.2: Configurazione degli iperparametri ottimali per i modelli classici, individuata tramite Grid Search.

Modello	Iperparametro	Valore Ottimale
Random Forest	n_estimators (nuovo)	20
	max_depth (nuovo)	25
	n_estimators (vecchio)	50
	k (N. features)	600
Decision Tree	criterion	entropy
	max_depth	None
	min_samples_split	2
	min_samples_leaf	1
	k (N. features)	700
Regressione Logistica	C (Regolarizzazione)	10.0
	penalty	l2
	k (N. features)	800
	TfIdf (norma)	l2
Support Vector Machine	C (Regolarizzazione)	1.0
	k (N. features)	800
	TfIdf (uso IDF)	True
Multinomial Naive Bayes	alpha (Smoothing)	0.01
	k (N. features)	700
	TfIdf (uso IDF)	False

5.2 Risultati della CNN

La configurazione finale del modello è stata raggiunta a seguito di un processo iterativo di ottimizzazione. Sono state sperimentate diverse tecniche, tra cui l'adozione di un learning rate dinamico, ma si è riscontrato che una configurazione con un learning rate fisso più basso ($1e-4$) e un'architettura più semplice (32 filtri) ha prodotto i risultati più stabili.

L'andamento del training del modello finale è visualizzato in Figura 5.2. Le curve di apprendimento rivelano un modello che generalizza bene, sebbene emerga un leggero ma evidente segnale di overfitting nelle fasi finali. Come si osserva, dopo le prime 20 epoche, le curve di training e di validazione iniziano a divergere gradualmente: la perdita sul training set continua a diminuire, mentre quella sul validation set si appiattisce. Questo indica che, pur avendo controllato il fenomeno con varie tecniche, il modello inizia a memorizzare alcune specificità dei dati di addestramento.

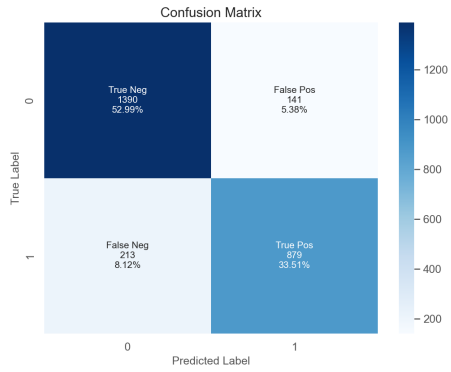
Tale comportamento, per quanto lieve, conferma l'importanza cruciale della tecnica di early stopping. Grazie a essa, infatti, per la valutazione finale sono stati ripristinati i pesi del modello corrispondenti al punto di minima validation loss, garantendo così di utilizzare la versione del modello con la massima capacità di generalizzazione, prima che l'overfitting diventasse più marcato.



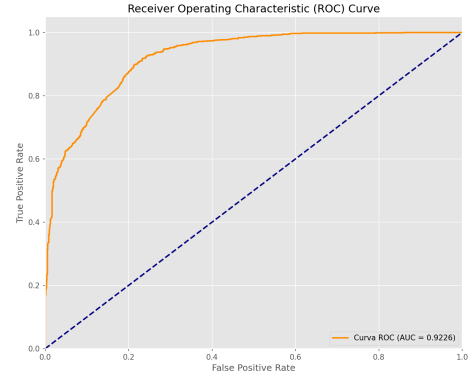
Figura 5.2: Andamento della funzione di perdita (Loss) e dell'accuratezza (Accuracy) durante l'addestramento della CNN ottimizzata.

La valutazione finale del modello ha riportato un F1-score (macro) di **0.83**, un punteggio AUC di **0.919** e un'accuratezza complessiva dell'**83.6%**.

Le performance sono analizzate in dettaglio nelle figure sottostanti. La matrice di confusione (Figura 5.3a) mostra una buona capacità di classificazione per entrambe le classi. In particolare, si nota un'alta Recall (circa 80%) sulla classe misogina, indicando che il modello è molto efficace nell'individuare i tweet che contengono linguaggio d'odio di genere. La curva ROC (Figura 5.3b) conferma visivamente la forte capacità discriminatoria del modello, con un'area sotto la curva (AUC) che sfiora il valore di 0.92.



(a) Matrice di confusione per il modello CNN.



(b) Curva ROC per il modello CNN.

Figura 5.3: Analisi grafica delle performance del modello CNN ottimizzato sul test set.

5.3 Analisi comparativa

Per concludere l'analisi, in questa sezione vengono messe a confronto diretto le performance del miglior modello di machine learning classico identificato, il **random forest**, e il modello di deep learning ottimizzato, la **rete neurale convoluzionale (CNN)**.

Tabella 5.3: Confronto delle performance finali tra il miglior modello classico e la CNN.

Modello	Precision (macro)	Recall (macro)	F1-score (macro)	AUC
Random Forest	0.841	0.823	0.829	0.933
Rete Neurale Conv. (CNN)	0.830	0.830	0.830	0.927

Come emerge dalla tabella, i due modelli raggiungono un livello di performance estremamente competitivo, con risultati quasi equivalenti. La CNN ottiene un F1-score macro leggermente superiore (0.830 vs 0.829), indicando un equilibrio marginalmente migliore tra precision e recall. Sorprendentemente, il random forest registra un valore di AUC lievemente più elevato (0.933 vs 0.927), suggerendo una maggiore affidabilità nella separazione delle classi a livello probabilistico.

Questa vicinanza nelle performance dimostra come un modello classico, se supportato da una solida pipeline di preprocessing e da un'attenta selezione delle feature, possa competere ad armi pari con architetture di deep learning più complesse. Tuttavia, l'analisi qualitativa delle loro strategie e dei loro errori di classificazioni rivela differenze fondamentali nel loro funzionamento.

La forza del random forest risiede infatti nella sua capacità di identificare e pesare termini chiave altamente predittivi. La Tabella 5.4 elenca le 12 feature più rilevanti identificate dal modello.

Tabella 5.4: Top 12 feature più importanti per il random forest.

Feature (termine)	Importanza
porca	0.096085
porca puttana	0.091817
mention	0.048370
puttana	0.042934
tettona	0.025447
cazzo	0.024505
culona	0.016195
schiaffi	0.014924
bel	0.013745
sono	0.012837
url	0.010959
dire	0.010116

Come si può osservare, il modello si basa fortemente su un lessico di insulti espliciti ('porca', 'puttana', 'cazzo'), termini strettamente legati all'oggettivazione del corpo della donna ('tettona', 'culona') e metadati come le menzioni e gli URL ('mention', 'url'). Questa strategia lo rende un eccellente rilevatore di insulti e di pattern offensivi diretti.

Il suo limite, tuttavia, emerge chiaramente dall'analisi degli errori. Il modello si rivela fragile di fronte a sfumature linguistiche come la **negazione** e il **sarcasmo**. Ad esempio, classifica erroneamente come misogini tweet non offensivi che in realtà condannano la violenza, poiché si concentra su una singola parola trigger ignorandone il contesto.

Esempio di falso positivo:

*"@giornalisteTV @Leone09748724 @Fiero70600786 @Priscil79855548
Beh, non mi sento sottomessa o malmenata"*

Analisi: Il modello si concentra sulla parola "malmenata", associata a un'alta importanza, ma non è in grado di interpretare la negazione "non" che ne inverte completamente il significato.

Allo stesso tempo, non riconosce la misoginia in frasi la cui aggressività è espressa tramite sarcasmo o con parole che, singolarmente, non sono volgari, violente o esplicitamente misogine.

Esempio di falso negativo:

@daechwita00 Ti apro in due

Analisi: La struttura sintattica violenta non viene colta. Il modello non trova le parole o i pattern espliciti che si aspetterebbe (come 'puttana' o 'cazzo') e classifica erroneamente il tweet come non misogino.

È qui che si manifesta il vantaggio, seppur minimo, della CNN. La sua natura gerarchica, capace di analizzare sequenze di parole tramite le convoluzioni, le permette di superare parzialmente i limiti del modello classico. Sebbene non immune agli errori, la CNN ha una maggiore capacità di:

1. **Adattarsi al contesto:** Il fine-tuning dello strato di embedding permette ai vettori pre-addestrati di specializzarsi sul gergo del dataset, cogliendo sfumature che un approccio statico come il TF-IDF potrebbe perdere.
2. **Comprendere la sintassi locale:** Analizzando n-grammi in sequenza, la CNN può apprendere che la presenza di “non” prima di un verbo ne inverte il significato, riducendo gli errori sui falsi positivi legati alla negazione.

Un approccio alternativo e promettente per superare i limiti osservati consisterebbe nell'utilizzo di Large Language Models (LLM) come estrattori di rappresentazioni contestuali. Invece di addestrare un modello da zero, si potrebbero sfruttare le rappresentazioni intermedie (hidden states) di modelli pre-addestrati su miliardi di token per catturare sfumature semantiche e pragmatiche ancora più sottili. Il processo funzionerebbe alimentando ogni tweet a un LLM pre-addestrato e estraendo i token di output degli strati intermedi come feature dense e altamente contestualizzate. Queste rappresentazioni, arricchite dalla conoscenza linguistica del modello pre-addestrato, potrebbero poi essere utilizzate come input per classificatori più semplici (es. regressione logistica o SVM) o per architetture neurali leggere. I vantaggi di questo approccio sarebbero infatti molteplici.

In prima istanza si potrebbe raggiungere una comprensione contestuale profonda, capacità chiave in questo contesto: gli LLM, addestrati su corpus testuali enormi, hanno sviluppato una comprensione implicita di fenomeni linguistici complessi come sarcasmo, ironia, negazione e implicature conversazionali.

Si avrebbero poi rappresentazioni dense e ricche: a differenza del TF-IDF che si basa sulla frequenza lessicale, le rappresentazioni degli LLM catturano relazioni semantiche, sintattiche e pragmatiche in spazi vettoriali ad alta dimensionalità.

Per i casi problematici identificati in questa analisi, ci si aspetterebbe che un LLM riesca a interpretare correttamente la negazione in frasi come "non mi sento... malmenata", generando rappresentazioni che riflettano l'inversione semantica e a riconoscere minacce implicite in costruzioni come quella vista in esempio, "Ti apro in due", avendo probabilmente incontrato pattern simili durante il pre-addestramento. Riuscirebbe poi a distinguere in ultima istanza contesti sarcastici da quelli letterali, grazie alla sua capacità di modellare le dipendenze a lungo raggio nel testo.

Questa direzione rappresenterebbe un'evoluzione naturale del presente lavoro, combinando la potenza computazionale degli LLM con la semplicità e l'interpretabilità dei classificatori tradizionali, offrendo potenzialmente il meglio di entrambi gli approcci.

Ringraziamenti

A tutte le meravigliose donne della mia vita.

Bibliografia

- [1] F. Bianchi et al. «PejorativITy: A Novel Resource for the Unsupervised Detection of Pejorative Language and Misogyny». In: *Proceedings of the Language Resources and Evaluation Conference (LREC)*. Marseille, France, giu. 2022, pp. 6780–6789.
- [2] L. Breiman. «Random forests». In: *Machine learning* 45.1 (2001), pp. 5–32.
- [3] C. Cortes e V. Vapnik. «Support-vector networks». In: *Machine learning* 20.3 (1995), pp. 273–297.
- [4] E. Fersini, D. Nozza e P. Rosso. «AMI@EVALITA2020: Automatic Misogyny Identification». In: *Proceedings of the 7th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA 2020)*. CEUR Workshop Proceedings. 2020.
- [5] E. Fersini, D. Nozza e P. Rosso. *Automatic Misogyny Identification (AMI) Shared Task at EVALITA 2020*. Sito Web Ufficiale. Accesso in data: 19 giugno 2025. 2020. URL: <https://amievalita2020.github.io/>.
- [6] GeeksforGeeks. *Bag of words (BoW) model in NLP*. <https://www.geeksforgeeks.org/bag-of-words-bow-model-in-nlp/>. Accesso: 26-06-2025. 2019.
- [7] M. Honnibal et al. «spaCy: Industrial-strength Natural Language Processing in Python». In: *Proceedings of the 17th Python in Science Conference*. A cura di A. Meurer, A. Smith e K. Class. 2020, pp. 425–434. DOI: 10.25080/Majora-342d178e-035.
- [8] D. Jurafsky e J. H. Martin. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall, 2009.
- [9] Y. Kim. «Convolutional Neural Networks for Sentence Classification». In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, ott. 2014, pp. 1746–1751. URL: <https://www.aclweb.org/anthology/D14-1181>.
- [10] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. URL: <http://tensorflow.org/>.
- [11] E. W. Pamungkas, V. Basile e V. Patti. «Misogyny detection in Twitter: a multilingual and cross-domain study». In: *Information Processing & Management* 57.6 (2020), p. 102360.
- [12] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

- [13] J. Pennington, R. Socher e C. Manning. «GloVe: Global Vectors for Word Representation». In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. A cura di A. Moschitti, B. Pang e W. Daelemans. Doha, Qatar: Association for Computational Linguistics, ott. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: <https://aclanthology.org/D14-1162/>.
- [14] M. Polignano et al. «AlBERTo: Italian BERT-based models for a wide range of NLP tasks». In: *Proceedings of the Sixth Italian Conference on Computational Linguistics (CLiC-it 2019)*. Bari, Italy: Accademia University Press, nov. 2019, pp. 1–7. URL: <https://www.aclweb.org/anthology/2019.clic-it-1.21>.
- [15] S. Soni, S. S. Chouhan e S. S. Rathore. «TextConvoNet: a convolutional neural network based architecture for text classification». In: *Applied Intelligence* 53.11 (ott. 2022), pp. 14249–14268. ISSN: 0924-669X. DOI: 10.1007/s10489-022-04221-9. URL: <https://doi.org/10.1007/s10489-022-04221-9>.
- [16] T. Wolf et al. «Transformers: State-of-the-Art Natural Language Processing». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, ott. 2020, pp. 38–45. URL: <https://www.aclweb.org/anthology/2020.emnlp-demos.6>.