REALIZZATO CON IL SOSTEGNO DI

PERCORSO ITS DENOMINATO:	BIG DATA ENGINEER & SOLUTIONS ARCHITECT - Tecnico Superiore per la progettazione, lo sviluppo e il test di soluzioni per la gestione del ciclo di vita dei Big Data
BIENNIO DI PERTINENZA:	2023-2025
GRUPPO DI LAVORO:	Davide Pedretti, Francesco Bombardieri, Riccardo Moltrasio, Andrea Cirillo

TITOLO DELLA TESINA:

Rilevamento di Fake News Un Approccio Machine Learning e Deep Learning

TECH TALENT FACTORY ITS ACADEMY

Rilevamento di Fake News con Word Embedding: Un Approccio di Machine Learning e Deep Learning

Davide Pedretti, Francesco Bombardieri, Riccardo Moltrasio, Andrea Cirillo

Luglio - Agosto 2025

Contents

1 Introduzione

La diffusione capillare delle fake news sui social media ha reso necessaria l'elaborazione di metodi di rilevamento sempre più sofisticati per preservare l'integrità dell'informazione. Questa ricerca analizza in modo sistematico l'efficacia di diverse tecniche di *word embedding* — TF-IDF, Word2Vec e Transformers — applicate a una varietà di modelli di *machine learning* (ML) e *deep learning* (DL) per il rilevamento delle fake news. Sfruttando il dataset **fake-and-real-news-dataset**, che include un insieme eterogeneo di articoli di notizie e post sui social

media etichettati e raccolti nell'arco di oltre un decennio, abbiamo valutato le prestazioni di classificatori come le Support Vector Machines (SVM) e i Multilayer Perceptrons (MLP).

1.1 Fonte dati

Fake and real news e un dataset, disponibile sulla piattaforma Kaggle, che raccoglie notizie e tweet, separandoli in fake e true news. Piu precisamente sono presenti 23481 osservazioni per la parte di notizie false e 21417 per quelle annotate come vere, formando, quindi, un dataset approssimativamente bilanciato. Le osservazioni sono composte da informazioni inerenti al titolo della notizia, al corpo, allargomento in essa contenuto e alla data di pubblicazione. La colonna relativa al titolo, pur essendo potenzialmente rilevante, non è stata considerata ai fini dell'addestramento. La motivazione riguarda principalmente la presenza di tweet: in tali casi è infatti difficile parlare propriamente di un "titolo", il quale risulta inoltre tendenzialmente poco informativo.

	title	text	subject	date
0	As U.S. budget fight looms, Republicans flip t...	WASHINGTON (Reuters) - The head of a conservat...	politicsNews	December 31, 2017
1	U.S. military to accept transgender recruits o...	WASHINGTON (Reuters) - Transgender people will...	politicsNews	December 29, 2017
2	Senior U.S. Republican senator: 'Let Mr. Muell...	WASHINGTON (Reuters) - The special counsel inv...	politicsNews	December 31, 2017
3	FBI Russia probe helped by Australian diplomat...	WASHINGTON (Reuters) - Trump campaign adviser ...	politicsNews	December 30, 2017
4	Trump wants Postal Service to charge 'much mor...	SEATTLE/WASHINGTON (Reuters) - President Donal...	politicsNews	December 29, 2017
5	White House, Congress prepare for talks on spe...	WEST PALM BEACH, Fla./WASHINGTON (Reuters) - T...	politicsNews	December 29, 2017
6	Trump says Russia probe will be fair, but time...	WEST PALM BEACH, Fla (Reuters) - President Don...	politicsNews	December 29, 2017
7	Factbox: Trump on Twitter (Dec 29) - Approval ...	The following statements were posted to the ve...	politicsNews	December 29, 2017
8	Trump on Twitter (Dec 28) - Global Warming	The following statements were posted to the ve...	politicsNews	December 29, 2017
9	Alabama official to certify Senator-elect Jone...	WASHINGTON (Reuters) - Alabama Secretary of St...	politicsNews	December 28, 2017

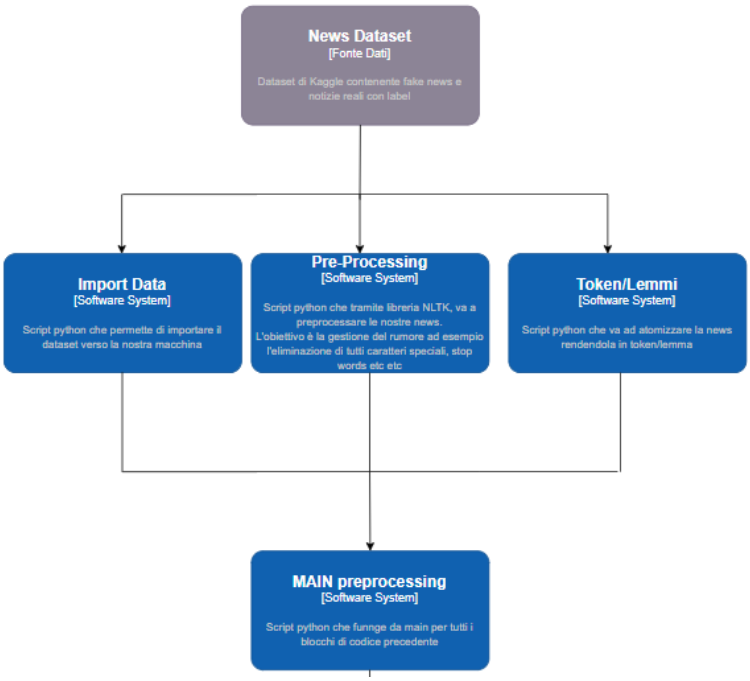
Figure 1: Esempio di notizie etichettate come *Real News*.

	title	text	subject	date
0	Donald Trump Sends Out Embarrassing New Year'...	Donald Trump just couldn t wish all Americans ...	News	December 31, 2017
1	Drunk Bragging Trump Staffer Started Russian ...	House Intelligence Committee Chairman Devin Nu...	News	December 31, 2017
2	Sheriff David Clarke Becomes An Internet Joke...	On Friday, it was revealed that former Milwauk...	News	December 30, 2017
3	Trump Is So Obsessed He Even Has Obama's Name...	On Christmas day, Donald Trump announced that ...	News	December 29, 2017
4	Pope Francis Just Called Out Donald Trump Dur...	Pope Francis used his annual Christmas Day mes...	News	December 25, 2017
5	Racist Alabama Cops Brutalize Black Boy While...	The number of cases of cops brutalizing and ki...	News	December 25, 2017
6	Fresh Off The Golf Course, Trump Lashes Out A...	Donald Trump spent a good portion of his day a...	News	December 23, 2017
7	Trump Said Some INSANELY Racist Stuff Inside ...	In the wake of yet another court decision that...	News	December 23, 2017
8	Former CIA Director Slams Trump Over UN Bully...	Many people have raised the alarm regarding th...	News	December 22, 2017
9	WATCH: Brand-New Pro-Trump Ad Features So Muc...	Just when you might have thought we d get a br...	News	December 21, 2017

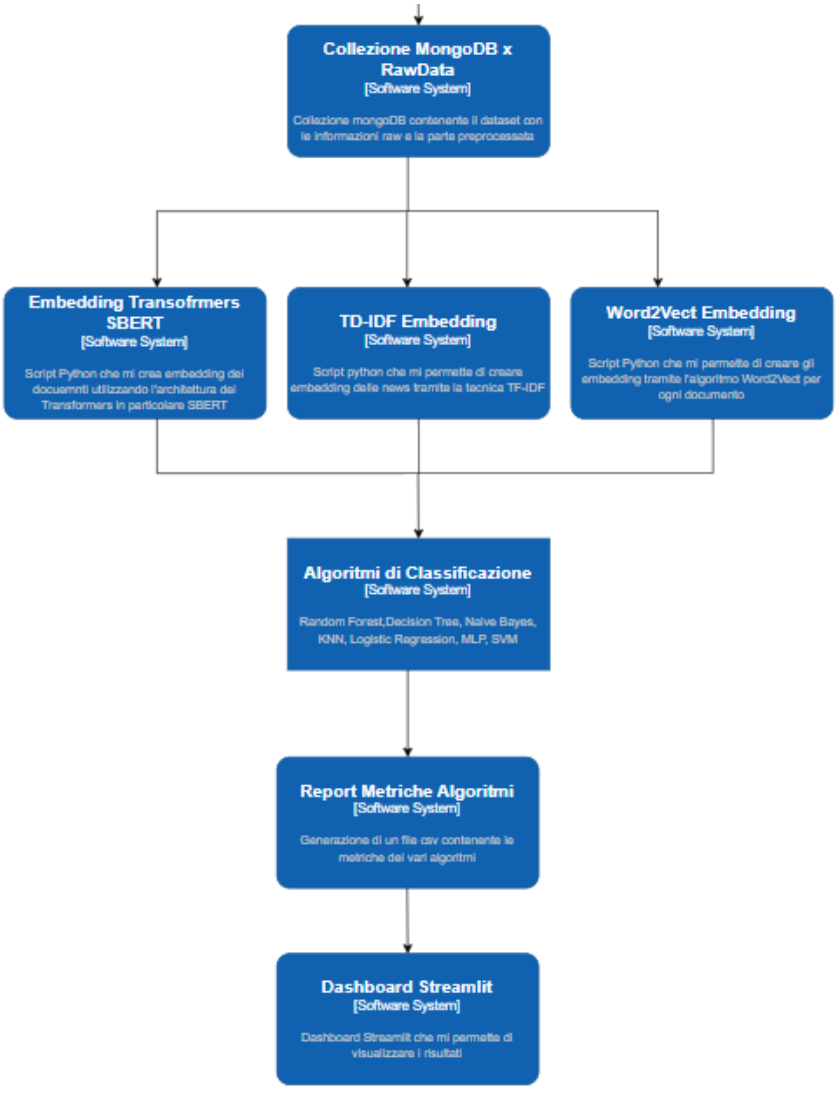
Figure 2: Esempio di notizie etichettate come *Fake News*.

2 Architettura

L'architettura implementata per la rilevazione di fake news è strutturata come una pipeline modulare che integra fasi di acquisizione, pre-elaborazione, rappresentazione e classificazione dei dati testuali. Il processo ha inizio con l'importazione di un dataset etichettato di notizie, che viene sottoposto a operazioni di pre-processing linguistico tramite script Python e librerie NLP (tokenizzazione, lemmatizzazione, rimozione di stopwords e caratteri speciali). La gestione centralizzata delle fasi di preprocessing è demandata a uno script principale, mentre i dati — sia in forma grezza che preprocessata — vengono archiviati all'interno di una collezione MongoDB per garantirne tracciabilità e riusabilità. Successivamente, i testi sono trasformati in rappresentazioni numeriche attraverso tre approcci differenti di embedding: TF-IDF, Word2Vec e SBERT (basato su Transformers). Tali rappresentazioni costituiscono l'input per una serie di algoritmi di classificazione supervisionata (Random Forest, Decision Tree, Naive Bayes, KNN, Logistic Regression, MLP e SVM), al fine di valutare le performance predittive in base alla tecnica di embedding adottata. Le metriche di ciascun modello vengono sintetizzate in un file di report e successivamente rese disponibili tramite una dashboard interattiva sviluppata in Streamlit, che consente l'analisi comparativa dei risultati e la valutazione critica delle diverse configurazioni sperimentali.



(a) Pipeline: Preprocessing del dataset di News



(b) Pipeline: Embedding, Classificazione e Dashboard

Figure 3: Architettura del sistema di analisi delle News

3 Richiami di teoria

3.1 Word emmbedding

Nel contesto della classificazione testuale e dell'elaborazione del linguaggio naturale, le tecniche di *word embedding* rivestono un ruolo fondamentale in quanto consentono di trasformare i dati testuali in vettori numerici interpretabili dai modelli di *machine learning* e *deep learning*. In questa sezione vengono discusse tre delle metodologie di embedding più utilizzate: Term Frequency–Inverse Document Frequency (TF-IDF), Word2Vec e Tranformers, che sono state adottate in questo lavoro.

3.1.1 TF-IDF

La *Term Frequency–Inverse Document Frequency* (TF-IDF) è una misura statistica utilizzata per valutare l'importanza di una parola all'interno di un documento rispetto a una collezione di documenti, o *corpus*. Combina due metriche: la *Term Frequency* (TF), che misura la frequenza di una parola in un documento, e la *Inverse Document Frequency* (IDF), che quantifica quanto unica o rara sia una parola all'interno del corpus.

Matematicamente, TF-IDF è definita come segue:

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{1}$$

dove

$$\text{TF}(t, d) = \frac{f(t, d)}{\sum_{t' \in d} f(t', d)} \tag{2}$$

$$\text{IDF}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{3}$$

Qui $f(t, d)$ rappresenta la frequenza del termine t nel documento d, $|D|$ è il numero totale di documenti, e $|\{d \in D : t \in d\}|$ è il numero di documenti che contengono il termine t.

Il punteggio TF-IDF assegna valori più alti ai termini che compaiono frequentemente in un documento ma raramente nell'intero corpus, evidenziando così i termini che risultano più rilevanti per il documento specifico.

3.1.2 Word2Vec

Word2Vec è una tecnica di *word embedding* basata su reti neurali che consente di apprendere rappresentazioni vettoriali distribuite delle parole a partire da grandi corpora testuali. L'obiettivo è generare vettori densi che catturino le relazioni semantiche tra termini, superando i limiti dei modelli basati esclusivamente su frequenze.

Il modello si articola in due principali architetture:

- **Continuous Bag of Words (CBOW)**: predice una parola target a partire dalle parole di contesto.
- **Skip-gram**: predice le parole di contesto a partire da una parola target.

Le rispettive funzioni obiettivo sono definite come segue:

$$J_{CBOW} = -\log p(w_t \mid w_{t-m}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+m}) \tag{4}$$

$$J_{Skip\text{-}gram} = -\log p(w_{t-m}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+m} \mid w_t) \tag{5}$$

dove w_t rappresenta la parola target, mentre $w_{t-m}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+m}$ rappresentano le parole di contesto.

Il risultato è uno spazio vettoriale continuo e denso, in cui la prossimità tra vettori riflette somiglianze semantiche e sintattiche.

3.1.3 SBERT e modelli basati su Transformers

I modelli basati su architettura *Transformers* hanno introdotto un approccio più avanzato al *word embedding*, consentendo di catturare non solo informazioni semantiche statiche, ma anche dipendenze contestuali tra le parole in una sequenza. A differenza di tecniche come TF-IDF, Word2Vec e FastText, che producono rappresentazioni indipendenti dal contesto, i *Transformers* generano embedding dinamici che variano a seconda della frase in cui un termine compare.

In particolare, *Sentence-BERT* (SBERT) estende l'architettura di BERT per produrre rappresentazioni vettoriali dense di intere frasi o documenti, ottimizzate per il calcolo della similarità semantica. SBERT utilizza una rete di tipo *Siamese* o *triplet* per addestrare i modelli a collocare frasi semanticamente simili in regioni vicine dello spazio vettoriale.

Nel contesto di questo progetto è stato adottato il modello pre-addestrato `sentence-transformers/all` una variante leggera e ottimizzata che bilancia efficienza computazionale e qualità della rappresentazione. Tale modello fornisce embedding a 384 dimensioni, mantenendo buone prestazioni in compiti di classificazione e similarità testuale, con tempi di calcolo ridotti rispetto a versioni più complesse di BERT

4 Tecniche di Machine Learning

In questo lavoro sono stati utilizzati diversi algoritmi di machine learning per affrontare il compito di classificazione delle fake news. Ogni tecnica presenta caratteristiche specifiche che ne giustificano l'applicazione in scenari differenti. Di seguito una sintesi dei modelli adottati:

- **Naive Bayes (NB):** Algoritmo probabilistico basato sul teorema di Bayes. Calcola la probabilità di appartenenza a una classe assumendo indipendenza tra le feature. È efficiente e adatto alla classificazione testuale, ma sensibile a correlazioni tra variabili e distribuzioni non gaussiane.

- **Logistic Regression (LR):** Modello statistico per la classificazione binaria. Applica la funzione logistica a una combinazione lineare delle feature, fornendo semplicità, interpretabilità ed efficienza, ma limitato nella cattura di relazioni non lineari.

- **K-Nearest Neighbors (KNN):** Algoritmo non parametrico che assegna la classe in base ai K vicini più prossimi secondo una metrica di distanza. Semplice ed efficace, ma poco scalabile e sensibile alla dimensione e alla scelta di K.

- **Support Vector Machines (SVM):** Ricercano l'iperpiano che massimizza il margine tra classi. L'uso di kernel consente di gestire relazioni lineari e non lineari. Il parametro C bilancia ampiezza del margine ed errori di classificazione.

- **Multilayer Perceptron (MLP):** Reti neurali feedforward con più strati di neuroni. Catturano relazioni complesse grazie a funzioni di attivazione non lineari. Possono soffrire di overfitting e richiedono risorse computazionali elevate.

- **Decision Trees (DT):** Strutture gerarchiche che suddividono ricorsivamente lo spazio delle feature. Interpretabili e capaci di gestire dati numerici e categorici, ma soggetti a overfitting se non adeguatamente potati.

- **Random Forest (RF):** Metodo di ensemble che combina più alberi di decisione addestrati su sottoinsiemi casuali di dati e feature. Riduce l'overfitting, migliora robustezza e accuratezza, e fornisce misure di importanza delle feature, sebbene con costi computazionali superiori.

4.1 Metriche di Valutazione

La valutazione dell'efficacia dei modelli di *machine learning* e *deep learning* nel rilevamento delle fake news richiede l'utilizzo di diverse metriche di performance. Questi indicatori consentono di analizzare la capacità dei modelli nel distinguere in modo accurato tra notizie false e autentiche, garantendo che il sistema di rilevazione sia affidabile ed efficace. Le metriche considerate sono le seguenti:

- **Accuracy:** misura la proporzione complessiva di classificazioni corrette, includendo sia i veri positivi (TP) sia i veri negativi (TN), rispetto al numero totale di istanze. Fornisce una panoramica generale delle prestazioni del modello, ma può risultare fuorviante in presenza di dataset sbilanciati.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision:** valuta la capacità del modello di identificare correttamente le notizie false, riducendo il numero di falsi positivi (FP). È particolarmente rilevante in contesti in cui i falsi allarmi hanno un costo elevato, poiché misura la proporzione di veri positivi tra tutte le istanze classificate come fake news.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** indica la capacità del modello di individuare tutte le notizie false realmente presenti. Si calcola come il rapporto tra i veri positivi e il totale delle fake news, inclusi i falsi negativi (FN). Un alto valore di recall è cruciale per ridurre il rischio di notizie false non intercettate.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score:** rappresenta la media armonica di precision e recall, fornendo una misura bilanciata che tiene conto sia dei falsi positivi sia dei falsi negativi. È particolarmente utile nel rilevamento delle fake news, dove occorre garantire un compromesso equilibrato tra sensibilità e precisione del modello.

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Dove:

- TP = veri positivi (notizie false correttamente individuate),
- TN = veri negativi (notizie autentiche correttamente classificate),
- FP = falsi positivi (notizie autentiche erroneamente classificate come false),
- FN = falsi negativi (notizie false non rilevate dal modello).

5 Implementazione

Dopo aver definito l'architettura complessiva e le scelte metodologiche alla base del processo di rilevazione delle fake news, si passa ora alla fase di implementazione pratica. In questa sezione vengono descritti gli strumenti, gli script e le librerie utilizzate per realizzare ciascun blocco della pipeline, evidenziando le modalità di integrazione tra le diverse componenti (preprocessing, embedding, classificazione e visualizzazione dei risultati) al fine di ottenere un sistema funzionale e replicabile. Dopo aver descritto l'architettura generale della pipeline per la rilevazione delle fake news, è possibile formalizzare il flusso delle operazioni tramite un algoritmo. L'obiettivo è tradurre in forma procedurale i blocchi principali già introdotti (preprocessing, rappresentazione tramite embedding e classificazione), evidenziando la sequenzialità dei passi e le interazioni tra le diverse componenti.

L'Algoritmo sintetizza dunque l'intero processo, dalla pulizia del testo grezzo fino alla valutazione comparativa dei modelli.

Algorithm 1: News Pre-processing e Valutazione dei modelli di Machine Learning

Input: Raw news text T, labels y
Output: Valutazione risultati R
Step 1: Text Cleaning ;
foreach *news* $t \in T$ **do**
    Rimozione URLs, caratteri speciali, valori numerici, emoticons, emojis, menzioni, hashtags;
    Rimozione caratteri non alfanumerici;
    Rimozione extra spazi e la formattazioni del testo;
    Conversione a lower-case;
Step 2: Tokenization ;
foreach *cleaned news* $t \in T$ **do**
    Suddivisione di t in token elementari;
Step 3: Rimozione Stop-Word ;
foreach *token* $w \in T$ **do**
    if w *is a stop word* **then**
        Rimuovi w from T;
Step 4: Train-Test Split ;
Suddivisione del dataset in training set (T_{train}, y_{train}) e testing set (T_{test}, y_{test});
Step 5: TF-IDF Vectorization ;
Vettorizzare T_{train} e T_{test} usando TF-IDF;
Step 6: Word2Vec Embedding ;
Allena Word2Vec on T_{train} e vettorizza T_{train}, T_{test};
Step 7: Transoformers Embedding ;
Allena Transofrmer on T_{train} e vettorizza T_{train}, T_{test};
Step 8: Definizione dei modelli ;
Modelli: Logistic Regression, Naive Bayes, KNN, SVM, MLP, Decision Tree, Random Forest;
Step 9: Valutazione e salvataggio dei risultati ;
foreach *embedding* $e \in \{$*TF-IDF*, *Word2Vec*, *Transformers*$\}$ **do**
    foreach *model* m *in models* **do**
        Allena m on T_{train} con embedding e;
        Predizione $\hat{y}_{test}$ usando m;
        Calcolo accuracy, precision, recall, F1-score;
        Salvo i risultati in R;
return R;

6 Importazione e Pre-Processing dei Dati

All'interno dell'intero progetto, i file `ImportData.py`, `PreProcessing.py` e `main.py` sono dedicati in modo specifico alla fase di importazione ed elaborazione preliminare dei dati, costituendo il blocco di partenza della pipeline.

6.1 ImportData.py

Il modulo `ImportData.py` gestisce la connessione a MongoDB e fornisce funzioni per l'importazione di dataset in formato CSV all'interno di una collezione: in questa fase i valori mancanti vengono sostituiti in modo coerente, eventuali colonne spurie vengono eliminate e vengono creati indici per ottimizzare sia le query di ricerca full-text che le operazioni di filtraggio.

Oltre all'inserimento, il file include strumenti di esplorazione e valutazione della qualità, come il conteggio delle occorrenze per soggetto, la distribuzione temporale delle notizie, statistiche sulla lunghezza dei testi, ricerche per parola chiave e controlli su campi mancanti o non validi.

6.2 PreProcessing.py

Il cuore della pipeline è però rappresentato dal modulo `PreProcessing.py`, che implementa in maniera modulare e robusta le principali operazioni di pulizia e normalizzazione del linguaggio naturale.

Il processo inizia con la rimozione di URL, menzioni, hashtag, caratteri speciali, numerici ed emoticon, con lo scopo di eliminare elementi privi di significato semantico e riportare il testo in una forma omogenea; a questo si aggiungono la standardizzazione in minuscolo e la rimozione di spazi superflui.

Segue la tokenizzazione, che suddivide il testo in unità elementari (token) e permette di trattare ogni parola come entità indipendente per le fasi successive di analisi. Una volta ottenuti i token, viene applicato il filtro delle stopwords, ovvero quelle parole molto frequenti che non portano informazione rilevante e che rischierebbero di introdurre rumore; in questa fase vengono però mantenute le negazioni, considerate importanti per preservare il senso delle frasi.

Infine, il processo si completa con la lemmatizzazione supportata da POS tagging, che consente di ricondurre ogni termine alla sua forma canonica, riducendo varianti morfologiche e ambiguità.

L'output di queste operazioni viene salvato in nuove colonne del dataset (`clean`, `tokens`, `lemmas`), rendendo i dati già strutturati per le successive fasi di embedding e classificazione.

6.3 Main.py

A coordinare il flusso interviene lo script `main.py`, che a partire dal dataset grezzo richiama le funzioni di pre-processing, gestisce l'inserimento dei dati trasformati in MongoDB e produce un report sintetico con campioni di notizie, distribuzioni, statistiche e indicatori di qualità.

In questo modo la fase di importazione e preparazione del testo viene incapsulata in un processo chiaro, riproducibile e facilmente estendibile, che costituisce la base solida su cui innestare i moduli successivi della pipeline dedicati alla rappresentazione tramite embedding e alla classificazione supervisionata.

7 Machine Learning

La fase di machine learning rappresenta il momento centrale della pipeline, in cui le rappresentazioni testuali ottenute attraverso i diversi metodi di embedding (TF-IDF, Word2Vec e SBERT) vengono utilizzate per addestrare e valutare vari algoritmi di classificazione. I notebook sviluppati a supporto di questa fase contengono la logica di training e testing dei modelli, seguendo un approccio sperimentale e comparativo. Sono stati implementati sia classificatori tradizionali, come Logistic Regression, Support Vector Machines, Naive Bayes, K-Nearest Neighbors e Random Forest, sia modelli più complessi come le reti neurali multilayer (MLP). Per ciascun algoritmo è stato definito un flusso che prevede l'addestramento sul training set e la successiva validazione sul test set, con la raccolta sistematica delle metriche principali (accuracy, precision, recall e F1-score). In questo modo è stato possibile osservare come la scelta dell'embedding influenzi direttamente le prestazioni dei diversi modelli e individuare i punti di forza e i limiti di ciascun approccio. La documentazione dei test all'interno dei notebook ha garantito trasparenza e riproducibilità, fornendo un quadro chiaro dei risultati ottenuti e delle differenze emerse tra le varie configurazioni.

8 Risultati Sperimentali

L'analisi comparativa dei modelli evidenzia come la scelta della tecnica di *word embedding* incida in maniera significativa sulle performance complessive. Con TF-IDF si osservano valori molto elevati in termini di accuratezza e F1-score, in particolare per Linear SVM (Accuracy = 0.9967, F1 = 0.9968) e Logistic Regression (Accuracy = 0.9888, F1 = 0.9892). Questi risultati confermano l'efficacia delle rappresentazioni sparse basate su frequenza, soprattutto quando i dati testuali presentano pattern ricorrenti facilmente separabili.

Passando agli embedding neurali, i modelli basati su Word2Vec mostrano prestazioni leggermente inferiori rispetto a TF-IDF, con Linear SVM, Logistic Regression e Random Forest che si attestano attorno a valori di accuratezza compresi tra 0.957 e 0.961. Questo suggerisce che, sebbene le rappresentazioni distribuite siano in grado di catturare meglio la semantica, l'assenza di un fine-tuning mirato possa limitarne l'efficacia in compiti di classificazione binaria come il rilevamento delle fake news.

Gli embedding contestuali ottenuti tramite SBERT (`all-MiniLM-L6-v2`) raggiungono performance competitive, con Logistic Regression e Linear SVM che mantengono valori di accuratezza prossimi a 0.97 e F1-score superiori a 0.97. Questi risultati evidenziano la capacità dei modelli Transformer di fornire rappresentazioni dense e contestuali più informative rispetto a Word2Vec, pur non raggiungendo in questo caso i valori estremi osservati con TF-IDF.

In sintesi, il confronto incrociato mostra come TF-IDF, nonostante la sua semplicità, resti estremamente efficace in scenari supervisionati con dataset etichettati di grandi dimensioni. Word2Vec appare meno performante in assenza di un addestramento specifico, mentre SBERT si pone come soluzione intermedia, garantendo buone prestazioni e maggiore generalizzazione grazie alla natura contestuale delle sue rappresentazioni. I risultati sono inoltre resi disponibili attraverso una dashboard interattiva sviluppata con *Streamlit*, che permette di esplorare in modo dinamico le metriche e confrontare in maniera più immediata le configurazioni sperimentali.

Table 1: Risultati con embedding TF-IDF

Modello	**Accuracy**	**Precision**	**Recall**	**F1**
Linear SVM	0.9967	0.9960	0.9977	0.9968
Logistic Regression	0.9888	0.9919	0.9866	0.9892
Random Forest	0.9718	0.9758	0.9702	0.9730
Decision Tree	0.9462	0.9386	0.9600	0.9492
KNN	0.8579	0.8022	0.9666	0.8768
Gaussian NB	0.9406	0.9412	0.9400	0.9406
Neural Network (MLP)	0.9891	0.9888	0.9894	0.9891

Table 2: Risultati con embedding Word2Vec

Modello	**Accuracy**	**Precision**	**Recall**	**F1**
Logistic Regression	0.9571	0.9570	0.9573	0.9571
Linear SVM	0.9578	0.9577	0.9578	0.9578
Random Forest	0.9614	0.9614	0.9614	0.9614
Gaussian NB	0.9002	0.9001	0.9002	0.9001
Decision Tree	0.9165	0.9174	0.9158	0.9163
KNN	0.9254	0.9280	0.9225	0.9252
Neural Network (MLP)	0.9408	0.9401	0.9412	0.9406

Table 3: Risultati con embedding SBERT (all-MiniLM-L6-v2)

Modello	**Accuracy**	**Precision**	**Recall**	**F1**
Logistic Regression	0.9700	0.9723	0.9704	0.9713
Linear SVM	0.9698	0.9723	0.9700	0.9711
Random Forest	0.9266	0.9218	0.9393	0.9305
KNN	0.8952	0.9321	0.8624	0.8959
Gaussian NB	0.8692	0.8769	0.8722	0.8746
Decision Tree	0.9025	0.9052	0.9000	0.9026
Neural Network (MLP)	0.9498	0.9510	0.9490	0.9499

Table 4: Confronto diretto dei modelli con diversi embedding (solo Accuracy).

Modello	**TF-IDF Acc**	**Word2Vec Acc**	**SBERT Acc**
Linear SVM	0.9967	0.9578	0.9698
Logistic Regression	0.9888	0.9571	0.9700
Random Forest	0.9718	0.9614	0.9266
Decision Tree	0.9462	0.9165	0.8304
KNN	0.8579	0.9381	0.8952
Gaussian NB	0.9406	0.9002	0.8692
Neural Network (MLP)	0.9891	0.9408	0.9498

Table 5: Confronto delle performance medie per embedding.

Embedding	**Accuracy media**	**F1 media**
TF-IDF	0.929	0.936
Word2Vec	0.944	0.944
SBERT (all-MiniLM-L6-v2)	0.920	0.923

9 Conclusioni

Il progetto ha mostrato come diverse tecniche di *word embedding* influenzino le prestazioni dei modelli di machine learning applicati al rilevamento delle fake news. TF-IDF si è confermato molto efficace in scenari supervisionati, mentre Word2Vec e SBERT hanno evidenziato una maggiore capacità di generalizzazione semantica. La pipeline realizzata, insieme alla dashboard interattiva in Streamlit, ha dimostrato la possibilità di costruire un sistema modulare, riproducibile e facilmente estendibile per l'analisi della disinformazione digitale.

9.1 Sviluppi futuri

Per estendere il lavoro svolto sono previsti diversi possibili sviluppi. In primo luogo, l'integrazione di modelli più complessi come le Convolutional Neural Networks (CNN) potrebbe migliorare ulteriormente la capacità di rilevare pattern non lineari e caratteristiche latenti del testo. Parallelamente, un potenziamento dell'infrastruttura permetterebbe di gestire un flusso più ampio e continuo di notizie da importare, processare e classificare in tempo reale. Un ulteriore fronte di espansione riguarda l'apertura ad altre tipologie di fake news: immagini manipolate, contenuti multimediali o dati provenienti da scraping di social network, con l'obiettivo di sviluppare un sistema multimodale. Infine, un obiettivo ambizioso ma auspicabile è quello di rendere il progetto *open source*, favorendo la collaborazione della comunità scientifica e degli sviluppatori per arricchire, scalare e migliorare costantemente la piattaforma.

References

[1] Wikipedia, *Word Embedding*. Disponibile su: https://en.wikipedia.org/wiki/Word_embedding.

[2] Autori Vari, *Enhancing Fake News Detection with Word Embedding: A Machine Learning and Deep Learning Approach*. Articolo disponibile online.

[3] Wikipedia, *Machine Learning*. Disponibile su: https://en.wikipedia.org/wiki/Machine_learning.

[4] Reimers, N. e Gurevych, I., *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. Disponibile su: https://www.sbert.net/.

[5] TensorFlow, *Word2Vec Tutorial*. Disponibile su: https://www.tensorflow.org/text/tutorials/word2vec.

[6] Scikit-learn, *sklearn.feature_extraction.text.TfidfVectorizer*. Disponibile su: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html.

[7] Scikit-learn, *Scikit-learn User Guide*. Disponibile su: https://scikit-learn.org/stable/index.html.

[8] PyTorch, *PyTorch Documentation*. Disponibile su: https://pytorch.org/.

[9] NLTK Project, *Natural Language Toolkit (NLTK)*. Disponibile su: https://www.nltk.org/.