Coronavirus tweets NLP - Text Classification

Relazione tecnica per il progetto di infrastrutture di calcolo e algoritmi efficienti per la biologia e medicina

Francesco Vissicchio - *matr.* 250331
Giusy Giovinazzo - *matr.* 250318
Maria Antonietta Iaria - *matr.* 250340

Febbraio 2024

Indice

1 Introduzione

In questo progetto è stato effettuato un tipico processo di Text Analysis. In particolare, abbiamo effettuato uno studio di Sentiment Analysis, che è definita come il processo di analisi di un testo per determinare se il tono emotivo del messaggio ha polarità positiva, negativa o neutrale. Abbiamo utilizzato tecniche statistiche e algoritmi di Natural Language Processing (NLP) e Machine Learning per compiere un compito di classificazione di dataset testuali contenenti tweet sul Coronavirus, i quali sono annotati in base al Sentiment espresso nel testo.

Questa relazione tecnica è suddivisa in tre sezioni principali e si completa con una conclusione in cui si riassume brevemente il lavoro svolto nel progetto e i principali risultati ottenuti. L'elaborato si articola quindi cominciando con una descrizione dell'analisi dei dataset di Train e di Test del nostro caso di studio, prosegue successivamente con una sezione dedicata al Machine Learning per la classificazione binaria del Sentiment dei nosti dati testuali. Infine, si conclude lo studio con un analisi più approfondita con due modelli diversi di Deep Learning.

2 Analisi dei dataset e preprocessing

La procedura che abbiamo eseguito per l'analisi di entrambi i dataset e per il preprocessing degli stessi è identica per entrambi. Perciò, tutto quello che è stato applicato per l'analisi di un dataset è stato adottato anche per l'altro. Di conseguenza, quando andremo a delineare la logica generale e i passaggi della nostra analisi faremo riferimento a un solo dataset, sapendo comunque che la procedura è la stessa anche per l'altro.

Per cominciare, abbiamo caricato i dataset di Train e di Test in un repository Github e abbiamo successivamente salvato ciascuno di essi in una variabile del nostro codice. Abbiamo effettuato un'analisi preliminare esplorativa del dataset tramite la libreria Pandas, che è una libreria per la manipolazione di dati in formato sequenziale o tabellare, come ad esempio serie temporali o dati di microarray. In questa analisi preliminare - dopo aver caricato il dataset - abbiamo visualizzato le prime 10 righe dello stesso, stampato le sue informazioni generali e visualizzato le statistiche descrittive delle sue colonne numeriche. Infine, calcoliamo il numero di valori nulli e visualizziamo il numero di valori unici in ciascuna colonna.

A questo punto ci occupiamo dell'analisi della distribuzione delle classi del *Sentiment* all'interno del dataset. In entrambi i dataset il Sentiment è etichettato sempre con 5 labels: Extremely positive, Positive, Neutral, Negative, e Extremely negative. Nel nostro progetto ci siamo limitati ad una classificazione binaria, escludendo dunque la classe Neutral dai modelli di apprendimento e associando le classi con la stessa polarità in un unica classe. Nella figura 1 viene mostrata la distribuzione delle classi nel dataset di addestramento.

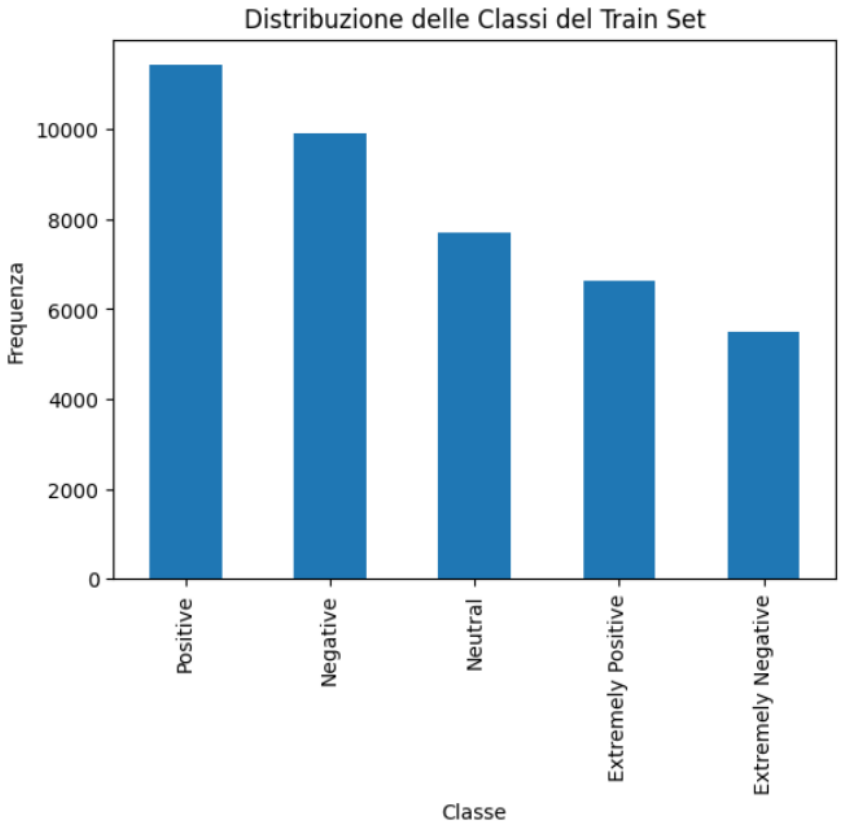

Figura 1: Distribuzione delle classi nel Train Set

Come ulteriore analisi esplorativa dei dataset ci occupiamo di effettuare la ricerca delle 10 parole più comuni del dataset e la stampa della WordCloud delle parole con frequenza maggiore dell'intero dataset. Scriviamo dunque un frammento di codice che analizza i dati testuali nel dataset di addestramento, identifica le parole più comuni e fornisce una rappresentazione visiva delle stesse tramite una word cloud. Ricorriamo alla libreria NLTK (Natural Language Toolkit) di Python, che è una libreria molto popolare utilizzata per il lavoro nel campo del linguaggio naturale (NLP). Serve come un insieme di strumenti e risorse per analizzare testi. Di seguito, in figura 2, si ha la WordCloud risultante per il dataset di addestramento.

Figura 2: WordCloud del Train Set

A questo punto eseguiamo una serie di operazioni per la pulizia del testo e l'estrazione delle caratteristiche che ci serviranno in seguito per l'applicazione dei modelli di apprendimento. Nel nostro progetto, abbiamo utilizzato la libreria scikit-learn di Python, che è una delle librerie più utilizzate per il machine learning e fornisce un insieme di strumenti per l'apprendimento automatico, l'analisi dei dati e l'elaborazione dei dati in Python. Lo scopo di questa prima parte di codice, però, è quello di eseguire la pulizia del testo e l'estrazione delle feature utilizzando CountVectorizer, fornendo una rappresentazione numerica dei dati testuali che può essere utilizzata per l'addestramento di modelli di machine learning. Successivamente facciamo una analisi esplorativa del dataset con il testo pulito e le features estratte. Infatti, per la pulizia del testo, abbiamo definito una funzione apposita, la quale rimuove le URL e le emoji, i numeri, i caratteri speciali e gli spazi extra. Inoltre, questa funzione rimuove la punteggiatura e converte tutti i caratteri in minuscolo.

Per quanto riguarda la feature extraction, applichiamo la funzione che abbiamo definito precedentemente ad ogni elemento della colonna contenente il tweet nel DataFrame. Utilizziamo dunque il CountVectorizer, che è una tecnica di feature extraction che converte una collezione di documenti di testo in una matrice di conteggi di token. Limitiamo il numero di parole da considerare alle 1000 parole più frequenti nel corpus. Successivamente viene addestrato il CountVectorizer sui dati di testo puliti e si trasformano i dati in una matrice sparsa di conteggi di parole. In altre parole, si impara il vocabolario dei dati di testo e si crea una matrice dove ciascuna colonna rappresenta una parola nel vocabolario e ciascuna riga rappresenta un istanza del set di dati con i valori corrispondenti al conteggio di quella parola nel documento. Infine, si convertono queste variabili in un DataFrame per poter fare un'analisi più semplice in

seguito. Questa procedura è comune nella costruzione di modelli di machine learning basati su dati testuali, soprattutto per compiti come la classificazione di testi.

In seguito alla pulizia del dataset, si esegue un'analisi esplorativa del dataset pulito per comprendere meglio la composizione del testo pulito, la distribuzione delle classi (Sentiment positivo o negativo), e per identificare le parole più comuni nel dataset basandosi sulle frequenze delle parole calcolate dal modello Bag of Words (BoW). Quest'ultimo è un modello semplificato utilizzato nel processamento del linguaggio naturale e nella classificazione di documenti di testo. Il modello BoW trasforma il testo in un insieme, o "borsa", di parole senza tenere conto dell'ordine o della struttura del testo, ma mantenendo la frequenza con cui le parole appaiono nel documento. Il modello BoW è ampiamente utilizzato perché è semplice da capire e implementare, e spesso fornisce una buona base di partenza per compiti di classificazione di testo, come il rilevamento dello spam, il Sentiment Analysis, e la categorizzazione dei documenti. Tuttavia, il modello ha dei limiti, come l'incapacità di catturare il contesto in cui le parole vengono utilizzate, poiché l'ordine delle parole e la semantica non vengono preservati. Come ulteriore analisi, si mostra anche quali parole sono le più comuni in tutto il dataset in ordine di frequenza decrescente, cioè dalla più comune alla meno comune.

Infine, eseguiamo un'analisi statistica dei tweet nei dataset preprocessati a livello delle parole (word level), per fornire un'analisi approfondita dei tweet nei dataset, come le statistiche descrittive, le visualizzazioni delle distribuzioni e l'identificazione delle combinazioni di parole più comuni, e quindi i bigrammi e i trigrammi. I bigrammi e i trigrammi sono delle sequenze che rientrano nell'ambito della linguistica computazionale e del processamento del linguaggio naturale (NLP) e si riferiscono a combinazioni di parole adiacenti in un testo. Un bigramma è una sequenza di due parole consecutive in un testo; un trigramma invece è una sequenza di tre parole consecutive in un testo. In generale, l'analisi di questi n-grammi è fondamentale per costruire modelli statistici del linguaggio che cercano di prevedere la probabilità di una parola data. Infine, vengono visualizzati gli istogrammi della distribuzione della lunghezza dei tweets e del conteggio delle parole in ciascun tweet, che possiamo osservare in figura 3. In pratica, questa ulteriore analisi esplorativa del testo dei tweet serve per capire meglio la loro lunghezza, il numero di parole e le loro combinazioni.

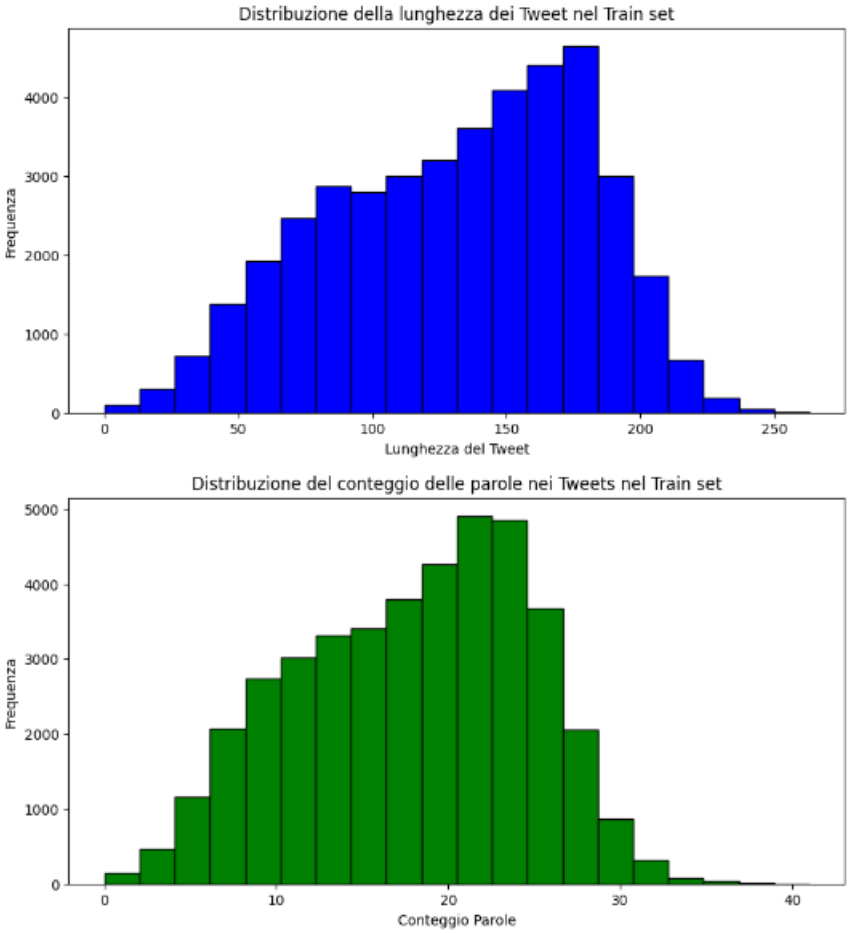

Figura 3: Istogrammi dell'analisi del Train Set (Word Level)

Per quanto riguarda la pulizia e il preprocessing del Test Set, bisogna considerare che deve essere utilizzato il vectorizer già adattato dai dati di training anche per il Test Set, e quindi ci si assicura che il vectorizer che applichiamo al Test Set sia quello già adattato con i dati di training. Questa considerazione va effuettuata poiché se si adattano i dati separatamente le colonne dei due dataset preprocessati non coincidono e non sarà possibile farli combaciare nei modelli di apprendimento.

3 Modelli di Machine Learning per la classificazione binaria del Sentiment

Procedendo con l'analisi del nostro codice dedicata al Machine Learning, verifichiamo innanzitutto che le colonne dei dataset di Train e di Test hanno lo stesso ordine, in modo tale che sarà possibile applicare i modelli di apprendimento ai dataset. Nel contesto del machine learning, per assicurare che i modelli addestrati funzionino correttamente sia durante l'addestramento che durante la valutazione e la previsione su dati nuovi o non visti questa verifica è una buona pratica, perché è necessario che i dataset di Train e di Test abbiano le stesse colonne nell'ordine esatto per poter prevenire molti problemi comuni e garantire che il modello di machine learning possa essere addestrato e testato efficacemente. I motivi principali per fare questa verifica sono la consistenza dei dati e la corrispondenza delle caratteristiche. Infatti, i modelli di machine learning si aspettano di ricevere lo stesso numero di caratteristiche (features) nello stesso ordine sia durante l'addestramento che durante il test. Se l'ordine o il numero delle colonne differisce, il modello potrebbe interpretare erroneamente i dati, portando a prestazioni scarse o errori. Inoltre, ogni colonna rappresenta una caratteristica specifica dei dati. Assicurarsi che le colonne corrispondano sia nel Train Set che nel Test Set garantisce che le stesse caratteristiche siano presenti in entrambi i set di dati, evitando così che il modello tenti di utilizzare una caratteristica durante il test che non è stata considerata durante l'addestramento.

Dopo aver verificato che i dataset di Train e Test che abbiamo analizzato e preprocessato siano compatibili, addestriamo e valutiamo quattro modelli di machine learning per la classificazione binaria del sentiment: Logistic Regression, Decision Tree, Random Forest e Naive Bayes Model. Questo è il processo per determinare se il Sentiment associato a un testo - come un Tweet, nel nostro caso - sia positivo o negativo. Nel codice scartiamo i Tweet con Sentiment Neutral e binarizziamo le classi associando 1 alle classi con polarità Positive e 0 alle classi con polarità Negative. Successivamente applichiamo i quattro modelli diversi per vedere quale è il più performante.

I quattro modelli che abbiamo utilizzato rappresentano un buon campionario delle tipologie principali di algoritmi usati in problemi di classificazione in machine learning, ma non coprono in modo esaustivo ogni possibile approccio. Abbiamo scartato modelli basati su Support Vector Machines (SVM) perché avevano un costo computazionale molto elevato. I modelli utilizzati si collocano nella famiglia dei modelli lineari, rappresentati dal modello Logistic Regression nel nostro codice; dei modelli basati sugli alberi, in cui rientrano Decision Tree e Random Forest; e dei modelli probabilistici, di cui fa parte il modello Naive Bayes. Il modello Logistic Regression è un modello lineare relativamente semplice e veloce da addestrare, e spesso è il primo modello che si prova nei problemi di classificazione binaria, e può servire come un buon punto di partenza per confrontare modelli più complessi. I modelli Decision Tree sono utili per comprendere come le decisioni sono prese dal modello di apprendimento au-

tomatico, perché sono facilmente interpretabili, e a differenza della regressione logistica, possono catturare relazioni non lineari tra le features e il target. Il modello Random Forest è un insieme di alberi decisionali e tende a essere più robusto e accurato rispetto a un singolo Decision Tree, e inoltre riduce il rischio di overfitting, che è comune negli alberi decisionali.

Per ciascuno dei quattro modelli utilizzati, viene eseguito l'addestramento utilizzando i dati di Train e valutata l'accuratezza sul dataset di Test. Il codice itera attraverso questi modelli addestrando ciascuno di essi con il set di addestramento e poi effettuando previsioni sul set di test. In seguito, viene stampato a video un resoconto di classificazione che include precision, recall e F1-score per entrambe le classi, quindi sia per la classe con polarità positiva che quella negativa. Viene definita una funzione per visualizzare la matrice di confusione, che è una tabella utilizzata per descrivere le prestazioni di un modello di classificazione. Infatti, la matrice di confusione mostra le classi corrette rispetto alle previsioni del modello, perché ogni colonna della matrice di confusione rappresenta i valori predetti, mentre ogni riga rappresenta i valori reali. Infine, il codice permette di visualizzare le curve ROC (Receiver Operating Characteristic) per ciascun modello. Le curve ROC sono grafici utili per valutare la performance complessiva di un modello di classificazione binaria. Inoltre, viene calcolata per ciascun modello l'Area Under the Curve (AUC), fornendo una misura singola di efficacia indipendente dalla soglia di decisione. Nella figura 4 si mostrano i grafici delle curve ROC di ciascun modello con le rispettive AUC.

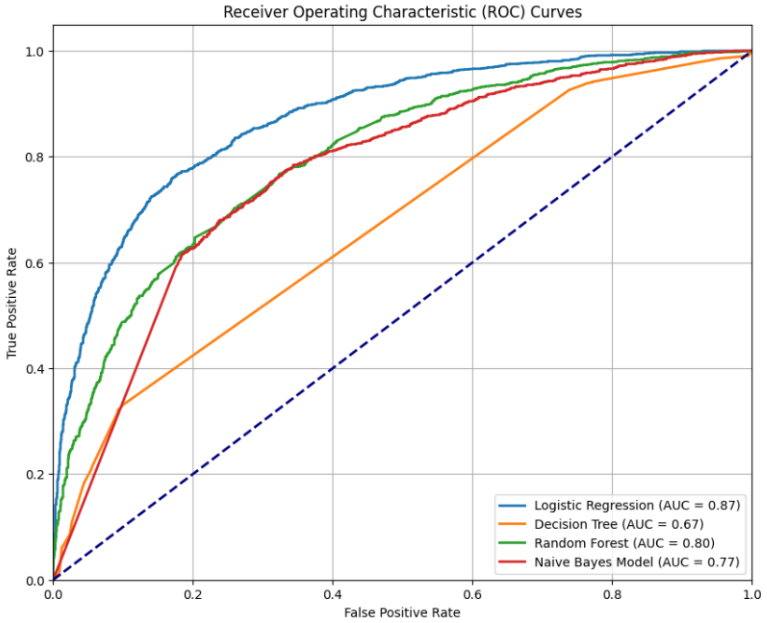

Figura 4: Curve ROC e valori AUC dei modelli utilizzati

Le curve ROC tracciano due parametri: il tasso dei veri positivi (TPR) e il tasso dei falsi positivi (FPR). Questi tassi sono definiti nel modo seguente:

$$TPR = \frac{TP}{TP + FN}$$
$$FPR = \frac{FP}{FP + TN}$$

In queste equazioni TP sono i veri positivi, FN sono i falsi negativi, FP sono i falsi positivi e TN sono i veri negativi. Il tasso dei veri positivi è anche noto come sensibilità, ed è l'asse delle ordinate che indica la proporzione di positivi reali che sono stati correttamente identificati dal modello; il tasso dei falsi positivi, invece, è l'asse delle ascisse che indica la proporzione di negativi reali che sono stati erroneamente identificati come positivi dal modello. L'Area Under the Curve (AUC) è una misura singola che riassume la performance dell'intera curva ROC ed è una misura che varia tra 0 e 1. Possiamo avere quattro casi:

- AUC = 1: Il modello ha una perfetta capacità di distinguere tra le classi positive e negative.
- $0.5 <$ AUC < 1: Maggiore è il valore dell'AUC, migliore è il modello nel distinguere tra le classi positive e negative. Un modello con un AUC di 0.9 è considerato molto buono.
- AUC = 0.5: Il modello non ha capacità di discriminazione tra classi positive e negative e si dice che faccia delle previsioni casuali. In pratica, il modello è equivalente ad una scelta basata sull'esito del lancio di una moneta.
- AUC < 0.5: Il modello performa peggio di una previsione casuale. Tuttavia, nella pratica, un valore di AUC inferiore a 0.5 può indicare che le classi sono state scambiate nel modello.

In conclusione, l'interpretazione delle curve ROC e dei valori AUC forniscono una valutazione complessiva di come i modelli si comportano nella classificazione binaria del Sentiment, e aiutano nella scelta del modello che ha la migliore capacità di distinguere tra le risposte positive e negative. Nel nostro caso, poiché ogni curva ROC mostra quanto bene il modello può distinguere tra le due classi del Sentiment (positivo o negativo), e che la curva più vicina all'angolo in alto a sinistra e più lontana dalla linea tratteggiata (che rappresenta una capacità di previsione casuale) è il modello migliore, allora possiamo affermare che il modello Logistic Regression è il modello più performante e il modello Decision Tree è quello meno performante. Inoltre, il modello con un AUC più alto - che in questo caso è il Logistic Regression - è considerato il migliore.

In seguito all'addestramento dei modelli di machine learning, utilizziamo la libreria Scikit-learn per visualizzare le curve di apprendimento dei quattro modelli presi in considerazione. Le curve di apprendimento indicano il rapporto tra la quantità di informazioni correttamente apprese e il tempo necessario per l'apprendimento, quindi sono grafici che mostrano le prestazioni di un modello di machine learning sia sul training set che sul validation set all'aumentare del numero di esempi di training. Per questo motivo, le curve di apprendimento sono utili per valutare se un modello potrebbe beneficiare dall'aggiunta di più dati di addestramento o se sta soffrendo di underfitting o overfitting. Di seguito vengono riportate i grafici delle curve di apprendimento dei quattro modelli.

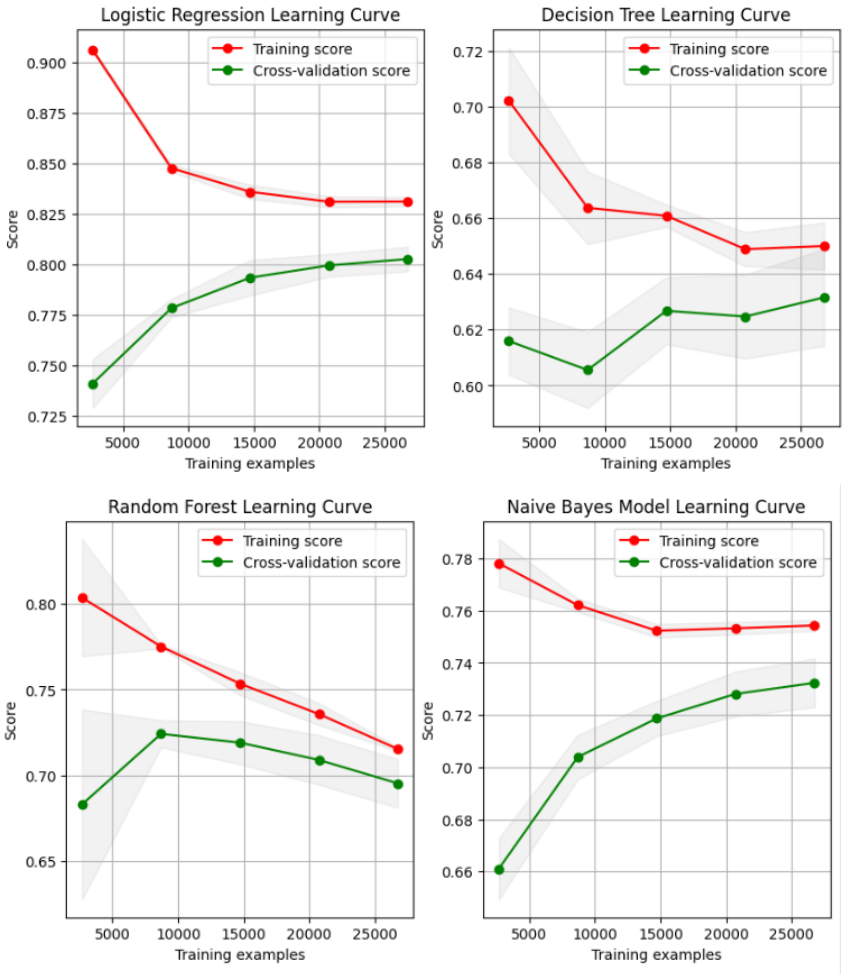

Figura 5: Curve di apprendimenti dei modelli di machine learning utilizzati

L'asse delle ascisse dei grafici rappresenta il Training examples, e mostra il

numero di esempi di addestramento utilizzati per addestrare il modello. L'asse delle ascisse parte da una frazione del set totale di addestramento e aumenta fino all'intero set. L'asse delle ordinate dei grafici, invece, rappresenta lo Score, cioè mostra il punteggio - come per esempio l'accuratezza - ottenuto dal modello sul set di addestramento e il set di validazione durante il processo di addestramento. Il punteggio può variare da 0 a 1, dove 1 indica la prestazione perfetta. La curva del Training Score, raffigurata in rosso, rappresenta la media dei punteggi di addestramento ottenuti per diverse dimensioni del set di addestramento. Se questa curva inizia alta e diminuisce man mano che si aggiungono più esempi, indica che il modello si adatta molto bene ai dati di addestramento quando la quantità di dati è piccola, ma la sua capacità di generalizzazione diminuisce con l'aumento dei dati. La curva del Cross-validation Score raffigurata in verde, invece, rappresenta la media dei punteggi ottenuti durante la cross-validation per diverse dimensioni del set di addestramento. Normalmente, questa curva inizia con un punteggio più basso e aumenta man mano che vengono aggiunti più dati di addestramento, indicando che il modello sta generalizzando meglio ai dati non visti man mano che apprende da più esempi. Infine, la bandiera di variazione, raffigurata dall'area grigia che contiene le curve, indica la deviazione standard dei punteggi per le diverse dimensioni del set di addestramento. Un'area grigia piccola implica che i punteggi sono consistenti tra le diverse ripetizioni della cross-validation, mentre un'area più grande indica una maggiore varianza e quindi meno affidabilità nei punteggi. Alla luce di quanto detto finora, osservando i quattro grafici riportati nella figura 5 possiamo constatare che il modello più performante tra i quattro presentati è la Logistic Regression, perché ha lo Score complessivamente più alto, e l'area grigia attorno alle curve più piccola e quindi i punteggi sono più affidabili.

Una volta effettuato l'addestramento dei modelli e visualizzato le curve di apprendimento, vengono mostrati i grafici a barre delle metriche di classificazione per ciascun modello. Queste visualizzazioni forniscono una panoramica completa delle prestazioni dei modelli in termini di classificazione binaria del Sentiment. Prima di tutto calcolate le metriche di precision, recall e F1-score utilizzando una funzione della libreria scikit-learn. Queste nuove metriche che vengono calcolate forniscono una valutazione più dettagliata delle prestazioni del modello rispetto alla semplice accuratezza. Dopodiché si mostra il grafico delle metriche. Nella tabella 1 vengono riportati i risultati.

Modello	Precision	Recall	F1 score
Logistic Regression	0.77	0.80	0.78
Decision Tree	0.54	0.93	0.68
Random Forest	0.60	0.92	0.73
Naive bayes	0.72	0.70	0.71

Tabella 1: Metriche di classificazione per ciascun modello di Machine Learning

I risultati mostrati nella tabella sono metriche comunemente utilizzate per valutare la performance di modelli di classificazione nel machine learning. In particolare, la precision indica la proporzione di identificazioni positive che sono effettivamente corrette. È calcolata come:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{1}$$

Possiamo osservare dalla tabella 1 che la Logistic Regression ha la precision più alta (0.77). Questo implica che quando questo modello predice un'etichetta positiva, quest'ultima è corretta il 77% delle volte.

Il recall (chiamato anche *sensibilità*) misura la proporzione di casi positivi reali che il modello è stato in grado di identificare correttamente. È definita come:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2}$$

Entrambi i modelli basati su alberi hanno un recall molto alto: il recall vale 0.93 per il Decision Tree e 0.92 per il Random Forest. Ciò significa che sono questi modelli capaci di identificare la maggior parte dei casi positivi.

Infine, la F1 Score è la media armonica di precision e recall e viene definita in formule come:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

La F1 Score tiene conto sia della precision che della recall, e perciò un punteggio F1 alto significa che il modello ha un buon equilibrio tra precision e recall. Nella tabella 1, la Logistic Regression ha il punteggio F1 più alto (0.78), indicando un miglior bilanciamento tra precision e recall rispetto agli altri modelli utilizzati.

In conclusione, considerando complessivamente i risultati delle metriche, il modello Logistic Regression sembra essere il modello più bilanciato per quanto riguarda la F1 score. Ciò potrebbe suggerire che questo modello potrebbe essere la migliore scelta se si desidera un compromesso tra precision e recall. Il modello Decision Tree, invece, ha un recall molto alto ma la precision più bassa. Questo

potrebbe indicare che classifica troppi esempi come positivi, aumentando anche il numero dei falsi positivi. Per quanto riguarda Random Forest, questo modello mostra un miglioramento rispetto al modello di Decision Tree in termini di precision mantenendo comunque un recall alto. Questo dato rende il modello Random Forest un'opzione migliore rispetto al Decision Tree nel caso in cui si vuole ridurre il numero di falsi positivi pur mantenendo un alto tasso di rilevamento dei veri positivi. Infine, il modello Naive Bayes ha valori moderati per precision e recall, rendendolo una scelta ragionevole se si cerca un equilibrio tra queste due metriche.

La scelta del modello dipende dal contesto in cui si deve utilizzare l'applicazione. Infatti, a seconda del costo relativo dei falsi positivi rispetto ai falsi negativi, si potrebbe preferire un modello con una precision più alta per ridurre il numero di falsi positivi oppure un modello con un recall più alto per identificare il maggior numero possibile di positivi veri.

4 Analisi Deep Learning

In questa sezione continueremo il nostro studio con un'analisi Deep Learning. Sono stati utilizzati due diversi approcci: l'approccio TF-IDF e l'approccio Word2Vec. TF-IDF (Term Frequency-Inverse Document Frequency) e Word2Vec sono due metodi differenti per la rappresentazione di parole e documenti utilizzati nel Natural Language Processing (NLP).

TF-IDF è un approccio statistico che valuta l'importanza di una parola in un documento rispetto a una collezione di documenti e si basa sulla frequenza delle parole. Tuttavia, TF-IDF non tiene conto dell'ordine delle parole perché non cattura la posizione delle parole all'interno dei documenti, il contesto o la semantica delle parole. Word2Vec, invece, è un approccio basato su reti neurali e utilizza una rete neurale a due strati per elaborare il testo, e produce un set di vettori di parole (word embeddings) dove parole con un contesto simile sono rappresentate da vettori simili nello spazio vettoriale. Word2Vec tiene conto del contesto in cui una parola appare e ne cattura la semantica. Poiché Word2Vec cattura anche la semantica delle parole, quelle semanticamente simili sono spesso vicine nello spazio vettoriale. In sintesi, nel Deep Learning, Word2Vec è generalmente preferito per la sua capacità di catturare relazioni semantiche e sintattiche complesse, mentre TF-IDF può essere utilizzato per compiti più semplici o come baseline per comparare l'efficacia dei modelli più avanzati.

4.1 Approccio TF-IDF

Cominciamo utilizzando le librerie TensorFlow e Keras per costruire, addestrare e valutare un modello di Deep Learning per la classificazione binaria del sentiment con l'approccio TF-IDF. TensorFlow fornisce un ambiente ricco e flessibile per il calcolo numerico e la rappresentazione dei grafi di dati, che consente agli sviluppatori di creare modelli complessi di apprendimento profondo; Keras, in-

vece, è un'interfaccia di alto livello per la costruzione e l'allenamento di modelli di deep learning. Una volta importate queste librerie, si convertono i dati di allenamento e test da Pandas DataFrame/Series in Numpy arrays, perché è quest'ultimo il formato richiesto da TensorFlow/Keras per addestrare il modello. Successivamente si definisce un modello sequenziale con la seguente architettura: un layer di input densamente connesso (Dense) con 64 neuroni e funzione di attivazione ReLU; un layer di Dropout con un rate del 50% per ridurre l'overfitting; un secondo layer Dense con 32 neuroni e funzione di attivazione ReLU; un secondo layer di Dropout con un rate del 50%; e infine un layer di output con un singolo neurone e funzione di attivazione sigmoid, adatto per la classificazione binaria. A questo punto si compila il modello con *binary_crossentropy* come funzione di Loss, e *adam* come ottimizzatore. Come metrica si tiene traccia dell'accuratezza. A questo punto addestriamo il modello sull'80% del Train Set per 100 epoche e con una dimensione del batch di 64. In seguito si valuta il modello sul Test Set e si stampa l'accuratezza, e poi si visualizza la perdita del modello (funzione di Loss) durante le epoche di allenamento e si visualizza tramite un grafico come quello della figura 6.

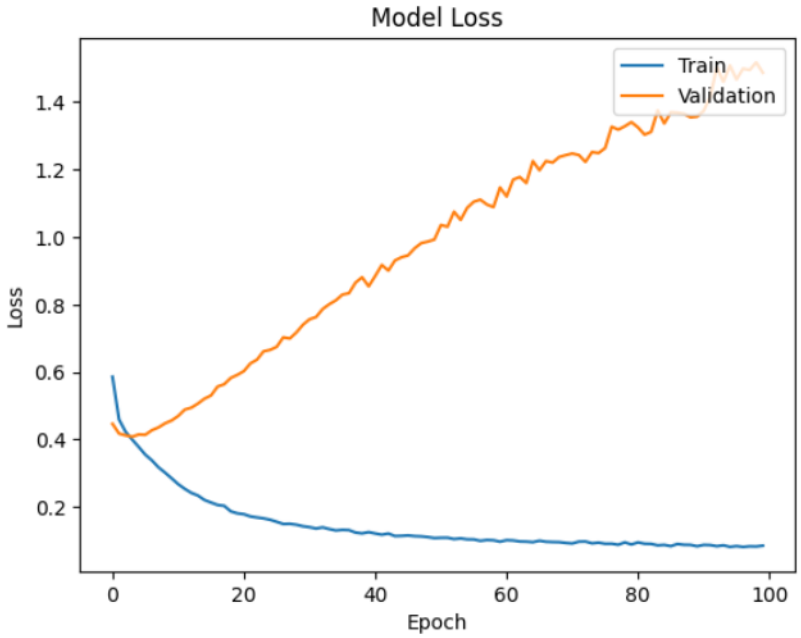

Figura 6: Funzione di Loss per l'approccio TF-IDF

Di conseguenza si utilizza il modello per fare previsioni sui dati di test e si calcola il tasso di veri positivi e il tasso di falsi positivi, che sono utilizzati per tracciare la curva ROC. A questo punto si calcola l'area sotto la curva ROC (AUC), e si visualizza la curva ROC in un grafico riportato in figura 7.

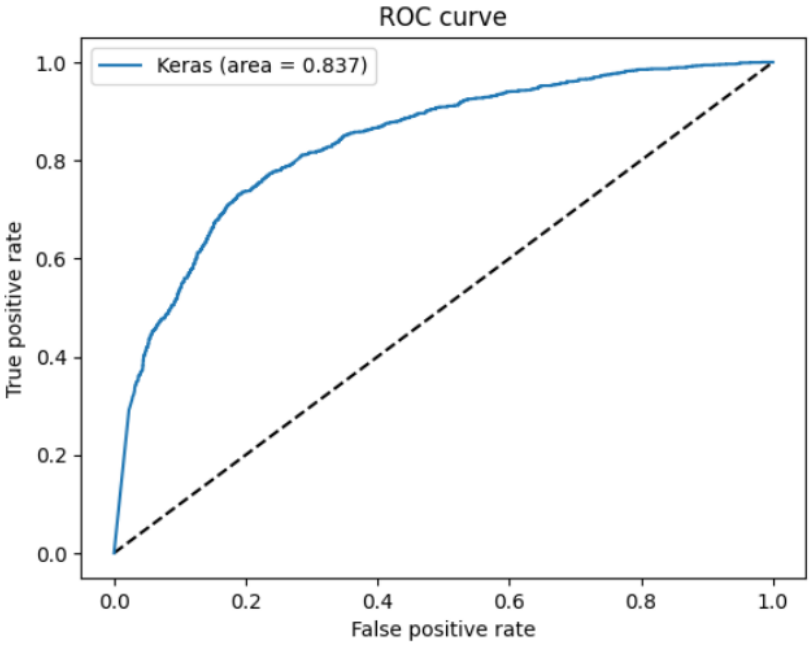

Figura 7: Curva ROC per l'approccio TF-IDF

Infine, si crea una matrice di confusione per valutare le prestazioni del modello, mostrando il numero di veri positivi, veri negativi, falsi positivi e falsi negativi, e si visualizza la matrice di confusione come una heatmap con l'ausilio della libreria seaborn.

4.2 Approccio Word2Vec

Prima di cominciare a costruire il nostro modello Deep Learning con l'approccio Word2Vec bisogna preparare i dati di input in modo più adeguato per questo approccio, che è diverso dal modo usato per l'approccio TF-IDF. Quindi si definisce una funzione Python che ha lo scopo di effettuare una serie di operazioni di preprocessing su un testo passato come parametro. Questo preprocessing è una fase tipica nella preparazione dei dati per tecniche di Natural Language Processing (NLP) come Word2Vec. Questa funzione restituisce quindi una versione pulita del testo originale, pronta per essere utilizzata in modelli di Deep Learning come Word2Vec, che beneficeranno di un input più standardizzato e rilevante. Infatti, la funzione è costruita per beneficiare del contesto fornito dalla punteggiatura e di altri elementi testuali che possono servire per l'approccio Word2Vec.

Per l'approccio Word2Vec dobbiamo importare, oltre la sopracitata libreria Keras, anche la libreria gensim. A questo punto addestriamo il modello Word2Vec sui testi tokenizzati del training set che abbiamo preparato con la

funzione creata appositamente. Questo modello mappa le parole in vettori numerici in uno spazio continuo in modo che le parole con contesti simili siano rappresentate con vettori simili. Successivamente, si utilizza la classe Tokenizer di Keras per convertire i testi in sequenze di numeri interi, dove ogni intero rappresenta un token unico, cioè una parola. Si esegue il padding delle sequenze per assicurarsi che abbiano tutte la stessa lunghezza, che è definita come *MAX_SEQUENCE_LENGTH*. Infine, si costruisce una matrice di embedding iniziale che utilizza i vettori ottenuti dal modello Word2Vec.

L'architettura del modello comprende una rete neurale sequenziale che include un layer di embedding, che è inizializzato con la matrice di embedding di Word2Vec, un layer LSTM (Long Short-Term Memory) per catturare le dipendenze temporali nel testo, e un layer di output con un'attivazione sigmoid per la classificazione binaria. Il modello con questo approccio è compilato in maniera analoga al modello con l'approccio TF-IDF che abbiamo delineato precedentemente. Una volta effettuati questi passaggi Il modello viene addestrato usando il Train Set con l'opportuno Padding e le corrispondenti etichette per un numero fissato di epoche pari a 10, in cui si ha anche una validazione sul Test Set. Esso viene successivamente valutato sul Test Set per ottenere la funzione di Loss e l'accuratezza. Quindi si visualizza la funzione di Loss e la validazione del modello per ogni epoca come riportato in figura 8.

Figura 8: Funzione di Loss per l'approccio Word2Vec

Inoltre, si calcolano le probabilità di appartenere alla classe positiva per

ciascun esempio del Test Set, e poi si calcolano e si visualizzano la curva ROC (Receiver Operating Characteristic) e l'AUC (Area Under the Curve), che sono le misure di performance per la classificazione binaria del Sentiment. Di seguito si riporta la curva ROC del modello con la rispettiva AUC:

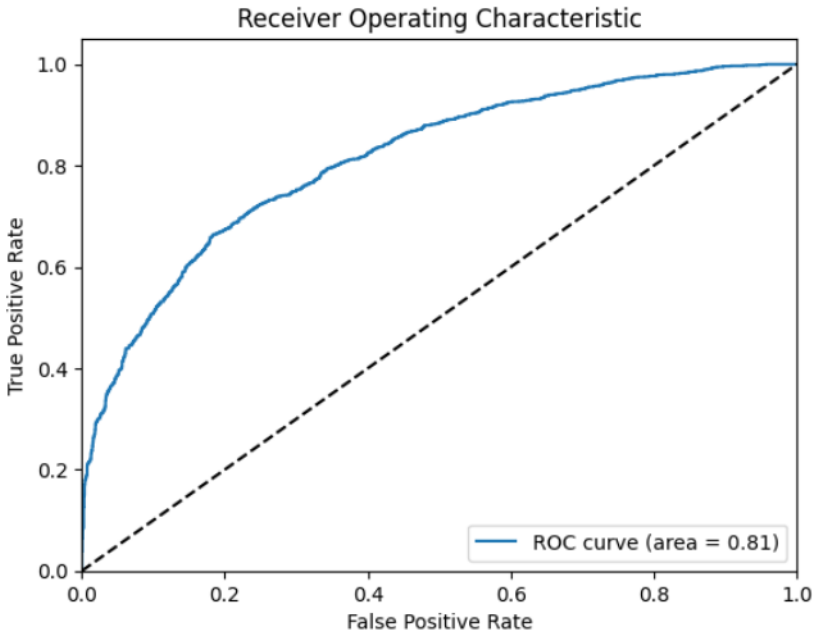

Figura 9: Curva ROC per l'approccio TF-IDF

In conclusione, analogamente a come abbiamo fatto nell'approccio TF-IDF, si calcola e visualizza la matrice di confusione, che mostra il numero di vere positività, false positività, vere negatività e false negatività. Si utilizza una heatmap sfruttando la libreria seaborn per visualizzare la matrice di confusione in modo più intuitivo.

4.3 Confronto dei risultati

Una volta addestrati ed eseguiti i due modelli di Deep Learning, possiamo confrontare i risultati ottenuti nei due diversi modelli dalle metriche di precisione, recall e F1 Score. Tuttavia, prima di valutare le metriche, possiamo confrontare i risultati ottenuti dalle rispettive funzioni di Loss e curve ROC di entrambi i modelli, e successivamente confrontiamo le metriche.

Nei grafici mostrati nelle figure 6 e 8 viene mostrato come la funzione di Loss di ciascun modello cambia nel corso delle epoche durante l'addestramento per il set di dati di allenamento, cioè la *Training Loss*, e per il set di dati di validazione, cioè la *Validation Loss*. Training Loss e Validation Loss sono

due metriche importanti per monitorare le prestazioni di un modello di Deep Learning durante l'addestramento. La curva di Training Loss si riferisce alla funzione di Loss calcolata sul set di dati di training ad ogni epoca durante l'addestramento e misura quanto accuratamente il modello è in grado di predire i dati di training; la Validation Loss, invece, si riferisce alla funzione di Loss calcolata sul set di dati di validazione ad ogni epoca durante l'addestramento, e misura quanto accuratamente il modello è in grado di generalizzare e predire i dati mai visti prima nel set di validazione. In genere la curva di Training Loss dovrebbe diminuire ad ogni epoca poiché il modello impara dai dati di training, mentre la Validation Loss può inizialmente diminuire mentre il modello impara, ma poi inizierà ad aumentare se il modello inizia a sovraddattarsi ai dati di training. Perciò, monitorando Training Loss e Validation Loss durante l'addestramento, possiamo capire se il modello sta imparando correttamente e generalizzando bene. Una Validation Loss che inizia ad aumentare indica overfitting e il bisogno di regolarizzazione o raccolta di più dati. In breve, Training Loss e Validation Loss aiutano a monitorare l'apprendimento e rilevare l'overfitting durante l'addestramento di un modello di Deep Learning.

In questi grafici in cui si mostra la funzione di Loss, l'asse delle ascisse rappresenta il numero di epoche attraverso il quale il modello è stato addestrato. Un'epoca corrisponde a un passaggio completo del set di dati di addestramento attraverso la rete neurale; l'asse delle ordinate, invece, mostra il valore della funzione di Loss, che è un indicatore di quanto bene il modello sta facendo previsioni rispetto alle etichette reali. Un valore basso indica che il modello sta facendo previsioni accurate, mentre un valore alto indica previsioni inesatte. La curva di Training Loss (in blu, in entrambe le figure) indica la perdita del modello sul set di allenamento dopo ogni epoca. Idealmente, questa dovrebbe diminuire man mano che il modello impara dai dati durante l'addestramento. Al contrario, la curva di Validation Loss (in arancione, in entrambe le figure) mostra la perdita del modello sul set di validazione dopo ogni epoca. La Validation Loss fornisce un'indicazione di come il modello si esibisce su dati che non ha mai visto durante l'addestramento, che è un test per valutare se il modello sta generalizzando bene o sta soffrendo di overfitting.

In generale, esistono vari scenari per quanto riguarda l'interpretazione dell'andamento della funzione di Loss nei modelli di Deep Learning. Nel caso in cui si ha una diminuzione di entrambe le curve di Loss, cioè se Training Loss e Validation Loss diminuiscono entrambe nel corso delle epoche e convergono verso un valore basso, allora questo andamento indica che il modello sta imparando bene e generalizzando bene sui dati non visti. Se si ha una diminuzione della Training Loss, ma un aumento della Validation Loss, cioè la curva di Training Loss continua a diminuire mentre la curva di Validation Loss continua ad aumentare, allora questo andamento potrebbe essere un segno di overfitting. Ciò significa che il modello sta imparando a memorizzare i dati di allenamento, ma fallisce nel generalizzare sui dati di validazione. Nel caso in cui si ha un'alta varianza tra le perdite, cioè si ha una grande differenza tra la curva di Training Loss e la curva di Validation Loss, allora potrebbe essere un sintomo di overfitting del modello. Infine, se si hanno curve di Loss di valore elevato e piuttosto

costante, cioè se entrambe le curve di Loss rimangono alte e non decrescono significativamente durante il loro andamento, allora il modello potrebbe non stare imparando affatto, indicando che potrebbero esserci problemi con l'architettura del modello, il processo di addestramento, o che i dati non sono adeguati o non sufficientemente informativi per il compito.

Nei casi dei nostri due modelli di Deep Learning presi in esame abbiamo che il modello che sfrutta l'approccio TF-IDF ha una Training Loss decrescente e una Validation Loss crescente e alla fine delle epoche si ottengono curve molto differenti; il modello che sfrutta l'approccio Word2Vec, invece, ha una Training Loss decrescente e una Validation Loss decrescente che però si assesta ad un valore costante più alto della Training Loss finale alla fine delle epoche. Per quanto riguarda il primo modello (approccio TF-IDF), si evidenzia un classico caso di overfitting. L'overfitting si verifica quando un modello di apprendimento si adatta troppo bene ai dati di addestramento e non generalizza bene su nuovi dati mai visti prima, cioè sui dati di validazione o di test. Questo è evidenziato dalla curva di Training Loss che continua a diminuire - indicando che il modello sta diventando sempre più preciso sui dati di addestramento - mentre la Loss di validazione continua a crescere, suggerendo che le prestazioni del modello stanno peggiorando sui dati di validazione. La crescente discrepanza tra queste due misure suggerisce che il modello sta iniziando a memorizzare i dati di addestramento, compreso il rumore e le anomalie, piuttosto che apprendere rappresentazioni generali. Al contrario, il secondo modello (approccio Word2Vec) sembra evitare l'overfitting, ma potrebbe soffrire di underfitting. L'interpretazione più ottimista potrebbe essere che il modello ha raggiunto tutto sommato un buon compromesso. L'underfitting si verifica quando un modello non è in grado di catturare la struttura sottostante dei dati di addestramento. Tuttavia, se la Validation Loss diminuisce e poi si assesta, potrebbe anche indicare che il modello ha imparato quanto gli è stato possibile dai dati di addestramento e adesso generalizza bene sui dati di validazione, ma potrebbe non essere comunque il modello più performante possibile, perché potrebbe esserci ancora spazio per migliorare la performance riducendo la Training Loss senza incrementare la Validation Loss.

Infine, non ci resta altro che interpretare i risultati delle metriche di precision, recall e F1 Score. Nella tabella 2 si mostrano i valori numerici che sono stati calcolati per ciascuno di questi due modelli (sono stati riportati i valori arrotondati alla seconda cifra decimale).

Modello	Precision	Recall	F1 score
TF-IDF	0.75	0.77	0.76
Word2Vec	0.78	0.62	0.69

Tabella 2: Metriche di classificazione per ciascun modello di Deep Learning

Ricordiamo che un valore alto di precision indica che il modello ha un basso numero di falsi positivi, un valore alto di recall indica che il modello è capace di identificare una grande quantità di esempi positivi reali, e un alto F1 Score indica un buon equilibrio tra precision e recall, che è particolarmente utile se la distribuzione delle classi è sbilanciata. Dunque possiamo osservare che il modello di Deep Learning addestrato con l'approccio TF-IDF ha una precision leggermente più bassa rispetto al modello addestrato con l'approccio Word2Vec, mentre quest'ultimo ha recall e F1 Score più basse del primo modello citato. Pertanto, possiamo concludere dicendo che il modello TF-IDF sembra essere più equilibrato in termini di precision e recall, con uno F1 Score che riflette questo equilibrio. Questo potrebbe essere preferibile in un contesto in cui sia importante identificare correttamente le istanze positive, mantenendo un livello ragionevole di precisione. Il modello Word2Vec, invece, ha una precisione migliore ma un recall inferiore, il che significa che potrebbe essere più selettivo ma tende a perdere più istanze positive reali. Questo potrebbe essere preferibile in situazioni in cui i falsi positivi sono molto costosi.

In ogni caso, per scegliere il modello migliore, abbiamo già accennato che è importante considerare il contesto specifico e l'importanza relativa di evitare falsi positivi rispetto a falsi negativi. Ad esempio, in un'applicazione medica in cui non si vuole perdere alcun possibile caso positivo (come per esempio una malattia), un alta recall potrebbe essere più importante, mentre in un contesto in cui i falsi allarmi (e quindi i falsi positivi) sono più costosi, si potrebbe preferire una precisione più alta.

5 Conclusioni

Nel presente lavoro abbiamo effettuato una Sentiment Analysis su due dataset che raccoglievano dei tweet sul Coronavirus. Abbiamo cominciato con un'analisi esplorativa dei due dataset per ottenere una comprensione preliminare del Train Set e del Test Set, in cui si sono esaminate le loro statistiche descrittive. In generale, un'analisi esplorativa aiuta a identificare anomalie nei dataset, i potenziali outlier e i problemi nei dati che potrebbero influenzare i modelli. Questa analisi consente di generare ipotesi sui dati prima di applicare tecniche più avanzate. Successivamente abbiamo eseguito il preprocessing in modo tale da ottenere i dati pronti per essere analizzati in modo efficace dagli algoritmi di apprendimento automatico che abbiamo sviluppato in seguito.

Una volta che abbiamo esplorato e preprocessato i dataset, ci siamo occupati di sviluppare degli algoritmi di Machine Learning per la classificazione binaria del Sentiment dei tweet contenuti nei dataset. Abbiamo visto le performance di quattro modelli di apprendimento supervisionato tramite le loro rispettive curve ROC, l'andamento delle curve di apprendimento e le metriche di precision, recall e F1 score. Infine, abbiamo messo a confronto due diversi approcci per l'analisi Deep Learning dei nostri dataset. In quest'ultima parte del lavoro abbiamo valutato e confrontato la funzione di Loss di entrambi i due modelli presi in considerazione: TF-IDF e Word2Vec. Inoltre, abbiamo visualizzato le

loro rispettive curve ROC e le metriche che abbiamo visto anche per i modelli di Machine Learning. Così facendo ci è stato possibile stabilire le performance di ciascuno dei due.