UNIVERSITÀ DEGLI STUDI DI SALERNO

Corso di Fondamenti di Intelligenza Artificiale

Professore: Fabio Palomba
Alunno: Simone Cava
(https://github.com/SimoneC24/text-spam-checker)

Anno Accademico 2022-2023

Indice

1 Introduzione

1.1 Cos'è Text-Spam-Checker

Text-Spam-Checker è un progetto di controllo anti spam di testo. Come capita, a quasi tutti noi, spesso ci troviamo a ricevere mail, chiamate o messaggi che consideriamo "spam".

In generale lo spam è l'invio di messaggi pubblicitari e/o truffaldini che l'utente non ha richiesto. E' un fenomeno mondiale, complesso, che colpisce tanti utenti. Purtroppo non è facile difendersi o bloccarlo. I provider lo combattono ogni giorno lavorando, in sintesi, su tre filoni che operano insieme: i servizi internazionali antispam, i propri sistemi antispam, le segnalazioni degli utenti, che possono accelerare il lavoro.

L'obiettivo degli spammer è la pubblicità: comuni offerte commerciali, proposte di vendita di materiale pornografico o illegale, farmaci senza prescrizione medica. Il loro scopo è carpire dati personali, indirizzi email e password di utenti, numeri di carte di credito e di conto corrente ecc.

Gli spammer sono, a tutti gli effetti, dei criminali. Lo spam è soprattutto strumento di truffa, che propone improbabili progetti finanziari, o vincite false fatte dall'utente in questione, chiede le credenziali di accesso al tuo conto corrente online. Altri messaggi truffa imitano quelli del tuo provider, di tuoi fornitori o della tua banca, avvisandoti della "scadenza" di un servizio o del "mancato pagamento" di una fattura e, con la scusa dell'urgenza, chiedono di fare versamenti o di inviare IBAN, carte di credito o dati personali.

Gli spammers inviano le mail pubblicitarie o truffaldine a migliaia di indirizzi, utilizzando server in località remote o caselle mittenti create appositamente di volta in volta. Oppure servendosi di caselle reali di ignari utenti, che sono riusciti a violare grazie al fatto che il titolare utilizzava una password non sicura o che le credenziali sono finite in rete a causa dell'hackeraggio di qualche piccolo sito cui l'utente si era iscritto.

1.2 Le motivazioni

Va detto che l'attività di contrasto allo spam operata dai provider avviene in tempo reale, ma purtroppo non è possibile "bloccare" a priori lo spam per una serie di motivi. Gli spammers utilizzano server e email che cambiano di continuo; anche il formato dei messaggi cambia: mittente, oggetto, contenuto, formattazione e allo stesso modo vengono cambiate le proprietà dei messaggi. Inoltre le campagne possono partire all'improvviso con un numero non elevato di messaggi, spediti però da una molteplicità di indirizzi diversi.

Il progetto Text-Spam-Checker nasce dall'idea di evitare queste truffe. Il concetto di spam è un qualcosa con cui tutti noi abbiamo a che fare ogni giorno anche in ambienti dove questo concetto non sembra esistere. Abbiamo pensato quindi di usare l'Intelligenza Artificiale per ammortizzare questo problema.

2 Specifiche P.E.A.S.

- **P** erformance: sono le misure di prestazione adottate per valutare l'operato di un agente, in questo caso valutiamo se l'agente restituisce il valore giusto tra spam e non-spam. Per valutare la bontà delle predizioni ho usato una matrice di confusione, anche detta tabella di errata classificazione, la quale restituisce una rappresentazione dell'accuratezza del classificatore.
- **E** nvironment: Descrizione degli elementi che formano l'ambiente. L'agente in questione opera nel campo dei messaggi spam. In seguito abbiamo descritto tutte le caratteristiche inerenti all'ambiente in questione.
- **A** ctuators: Gli attuatori servono all'agente per compiere le azioni. In questo caso sarà il modello di machine learning addestrato.
- **S** ensors: I sensori sono dispositivi attraverso i quali l'agente riceve gli input percettivi. Nel nostro caso sarebbe la tastiera attraverso la quale scrive il messaggio.

2.1 Caratteristiche dell'ambiente

- **Singolo agente**: nell'ambiente è possibile avere solo un agente.
- **Tipo di ambiente**: classificatore di messaggi spam o non-spam.
- **Completamente osservabile**: attraverso i sensori l'agente conosce sempre l'ambiente. Ha una visione completa dei messaggi all'interno del database.
- **Non deterministico**: è un ambiente che non cambia nel tempo. Lo stato successivo dell'ambiente non è determinato dallo stato corrente e dall'azione eseguita dall'agente.
- **Episodico**: L'esperienza dell'agente è divisa in "episodi" atomici, dove ciascun episodio consiste dell'eseguire una singola azione. La scelta dell'azione in ciascun episodio dipende dall'episodio stesso.
- **Statico**: l'ambiente resta invariato mentre l'agente esegue le azioni.
- **Discreto**: l'ambiente fornisce un numero limitato di percezioni e azioni distinte.

3 Machine Learning

Il Machine Learning è una branca dell'Intelligenza Artificiale, al contrario di quello che si pensa comunemente con il termine Machine Learning non intendiamo l'intero concetto di Intelligenza Artificiale ma appunto solo una parte.

Attualmente, il Machine Learning è utilizzato ovunque. Quando interagiamo con le banche, acquistiamo online o utilizziamo i social media, vengono utilizzati gli algoritmi di Machine Learning per rendere la nostra esperienza efficiente, facile e sicura. Il Machine Learning e la tecnologia associata si stanno sviluppando rapidamente e noi abbiamo appena iniziato a scoprire le loro funzionalità. Questo è uno dei tanti motivi per cui la scelta è ricaduta su un agente capace di apprendere e non su un agente basato su utilità.

Dunque il Machine Learning è sostanzialmente lo studio che esplora algoritmi e tecniche in grado di apprendere dai dati a loro disposizione e sulla base di questi fare predizioni.

Ci sono vari tipi di apprendimento quando parliamo di Machine Learning. Apprendimento supervisionato, semi-supervisionato, non supervisionato e per rinforzo.
Per il nostro progetto ci siamo concentrati sull'apprendimento supervisionato. In questo progetto, quindi, si userà un algoritmo che usa dati etichettati, ovvero dati di cui si conosce la variabile dipendente che sarebbe il valore che il modello dovrà predire una volta addestrato.

4 CRISP-DM

Sono due le cose principali a cui pensare quando si vuole progettare una soluzione basata su Machine Learning; data & software engineering. Il modello CRISP-DM rappresenta il ciclo di vita di progetti basati su Intelligenza Artificiale e Data Science.

CRISP-DM è l'acronimo di **CR**oss-**I**ndustry **S**andard **P**rocess for **D**ata **M**ining. Possiamo paragonare il modello CRISP-DM ad un modello a cascata con feedback utilizzato per lo sviluppo di sistemi software tradizionali. Il CRISP-DM è un modello non sequenziale in cui le diverse fasi possono essere eseguite un numero illimitato di volte. Di seguito eseguo e spiego tutte le diverse fasi.

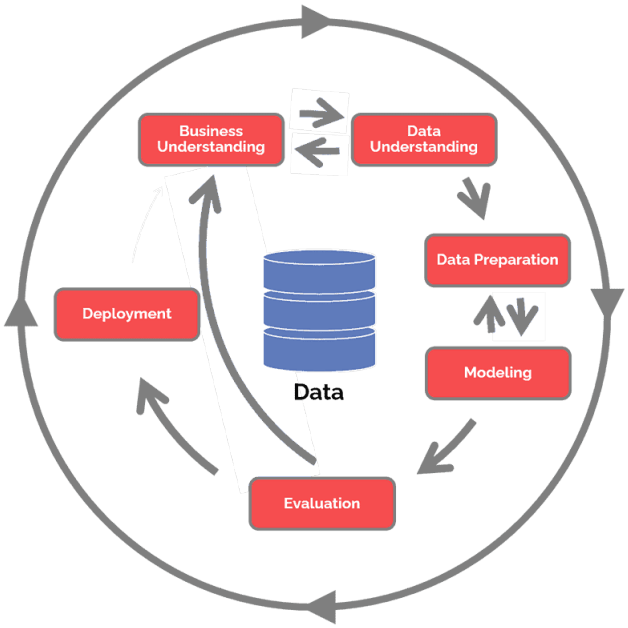

Figura 1: crisp-dm

4.1 Business Understanding

La prima fase è chiaramente quella di raccolta dei requisiti e di definizione degli obiettivi di business che si intende raggiungere (ovvero, che cosa deve fare il machine learner che stiamo progettando).

La fase di business understanding prevede la definizione dei cosiddetti business **success criteria**, ovvero i criteri secondo i quali potremo accertare che il sistema costruito è in linea con gli obiettivi di business.

In questa fase, bisogna inoltre determinare la disponibilità delle risorse, stimare i rischi, definire i relativi piani di contingenza e condurre una analisi costi-benefici. Oltre che definire i criteri di successo da un punto di vista di business, è inoltre necessario definire gli obiettivi tecnici che si intendono raggiungere. Infine, verranno selezionate le tecnologie ed i tool necessari agli obiettivi.

- **Obiettivi di business**: Il machine learner che intendiamo progettare avrà l'obiettivo di stimare o predire il valore della variabile dipendente di un messaggio che come abbiamo detto può essere spam o ham (non-spam) .
- **Risorse disponibili**: Per raggiungere l'obiettivo ci serviremo di un dataset di messaggi. Il dataset in questione consta di circa 5500 righe, in formato csv, e in ogni riga troviamo una stringa di testo, la quale rappresenta il messaggio da valutare, e un'etichetta in formato testuale autoesplicativa, con due valori ammessi (spam, ham). Facendo alcune ricerche, anche se questo dataset non ha tutte le occorrenze di messaggi che servirebbero, è sembrato il più adatto per lo sviluppo del machine learner. Abbiamo potuto scaricare il suddetto dataset da Kaggle.
- **Stima dei rischi**: Il dataset a nostra disposizione non ha abbastanza occorrenze per permettere al modello un apprendimento quasi perfetto. In compenso tutte le istanze del dataset hanno una qualità elevata in termini di lunghezza e studio del messaggio.
- **Tecnologie e strumenti**: Per acquisire, analizzare e modellare i dati abbiamo utilizzato il linguaggio Python poiché ho potuto fare uso di alcune librerie. Le librerie e gli aiuti di maggiore rilevanza sono:
  - Pandas che fornisce strutture e strumenti per l'analisi dei dati;
  - Seaborn per la creazione di grafici;
  - Basic Text Classification una guida di Tensorflow su come realizzare un modello di classificazione del testo.
  - Save and load model una guida di Tensorflow su come realizzare dei modelli persistenti.
  - Matrice di confusione una guida su come mostrare una matrice di confusione in un ambiente Python.

4.2 Data Understanding

Sulla base degli obiettivi definiti nella fase precedente, il secondo passo consiste nell'identificazione, collezione e analisi dei dataset che possono portare al raggiungimento degli obiettivi.

Innanzitutto, quindi, vengono acquisiti i dati necessari al raggiungimento degli obiettivi di business e i dati verranno poi caricati in un tool di analisi dei dati. I dati vengono quindi esaminati e documentati rispetto al loro formato, il numero di record e il significato di ciascun campo.

Il terzo task consiste nell'esplorazione dei dati: questi vengono visualizzati e, cosa più importante, eventuali relazioni vengono identificate. Infine, il processo di qualità dei dati, vengono identificati e documentati possibili problemi di qualità dei dati (ad esempio, dati mancanti).

Nel nostro caso il dataset in questione ha due colonne e circa cinquemilacinquecento righe. La prima colonna riguarda le etichette che come abbiamo già detto possono assumere due valori ovvero ham o spam. La seconda colonna contiene i messaggi che sono scritti in lingua inglese e contengono un massimo di novecentodieci caratteri.

Per leggere il dataset abbiamo usato una funsionalità della libreria *Pandas*, il dataset si presenta nella seguente forma:

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will ü b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

Figura 2: dataset

Inoltre con una funzione è stato possibile anche creare una tabella delle frequenze che esplicita in modo chiaro quali sono le etichette e i messaggi più frequenti:

	count	unique	top	freq
Category	5572	2	ham	4825
Message	5572	5157	Sorry, I'll call later	30

Figura 3: tabella frequenze

Infine abbiamo potuto anche visualizzare la distribuzione delle etichette:

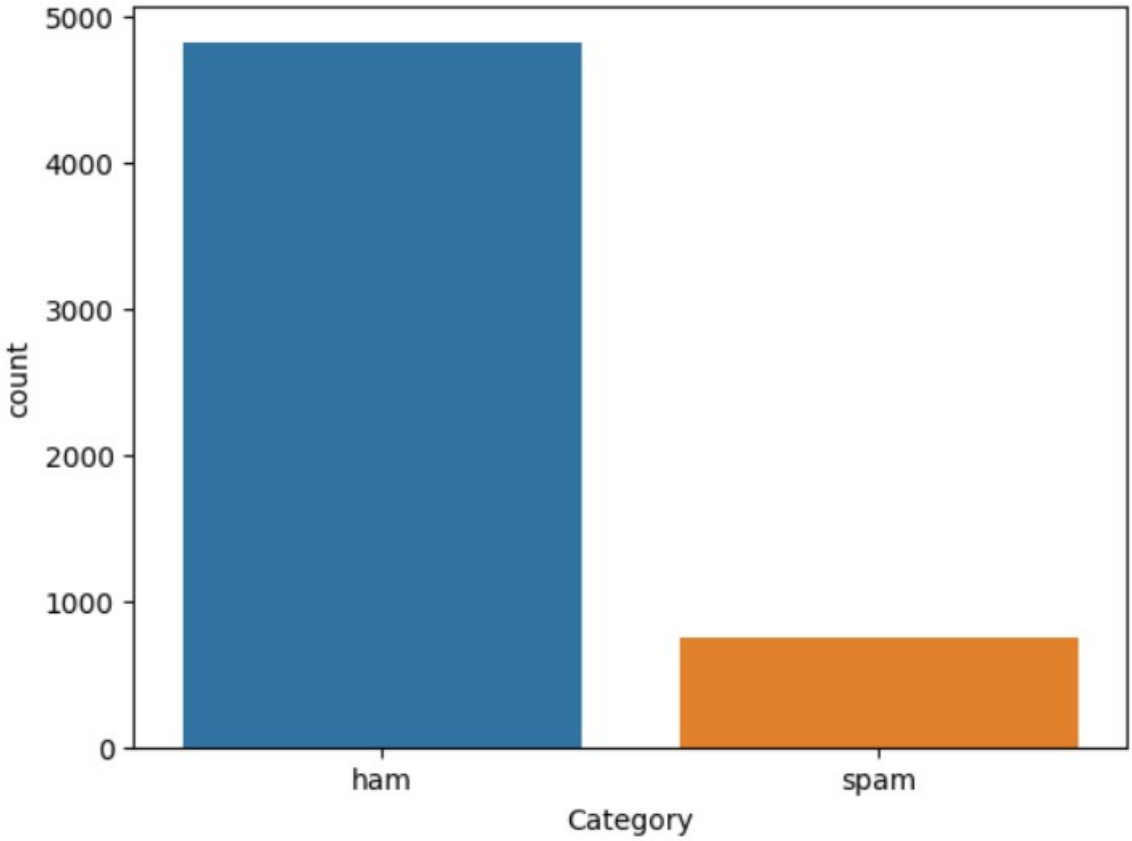

Figura 4: distribuzione etichette

Come si può osservare nel grafico a barre riportato di sopra, la distribuzione delle etichette è particolarmente sbilanciata, questo potrebbe portare a problematiche nelle fasi successive all'apprendimento.

4.3 Data Preparation

L'obiettivo di questa fase è quello di preparare i dati in maniera tale che possano essere utilizzati nei successivi passi del processo.

Questo include un processo fondamentale che è noto come feature engineering, ovvero la selezione delle caratteristiche del problema che hanno maggiore potenza predittiva. Inoltre, questa fase include l'implementazione dei processi di pulizia dei dati sulla base dei problemi di qualità riscontrati nella fase precedente.

Sulla base della pulizia fatta così come dell'analisi della potenza predittiva delle caratteristiche considerate, il progettista può considerare di estendere le caratteristiche da considerare.

Infine, i dati vengono formattati in una maniera tale che possano essere presi in input da un modello di machine learning, questo potrebbe dipendere dai tool selezionati in fase di business understanding.

La data preparation si divide in 4 fasi fondamentali che sono quelle descritte di seguito.

4.3.1 Data cleaning

La prima fase è quella di pulizia di dati. Con il **data imputation**, che è un insieme di tecniche che riescono a stimare il valore di dati mancanti sulla base dei dati disponibili. Queste tecniche ci permettono quindi di scartare le righe o colonne del dataset che presentano dati mancanti. Questa

soluzione è facile ma non sempre applicabile.
In alternativa, possiamo usare l'**imputazione statistica**, la quale si basa sull'applicazione di semplici tecniche statistiche per stimare il valore dei dati mancanti. Questa tecnica però non può essere applicata su valori non-numerici e non considera l'incertezza quando stima i dati. Possiamo fare questo in due modi, tramite **most frequent imputation** sostituendo i dati mancanti con i valori più frequenti oppure tramite **imputazione deduttiva**, deducendo, appunto, in maniera logica i valori mancanti.

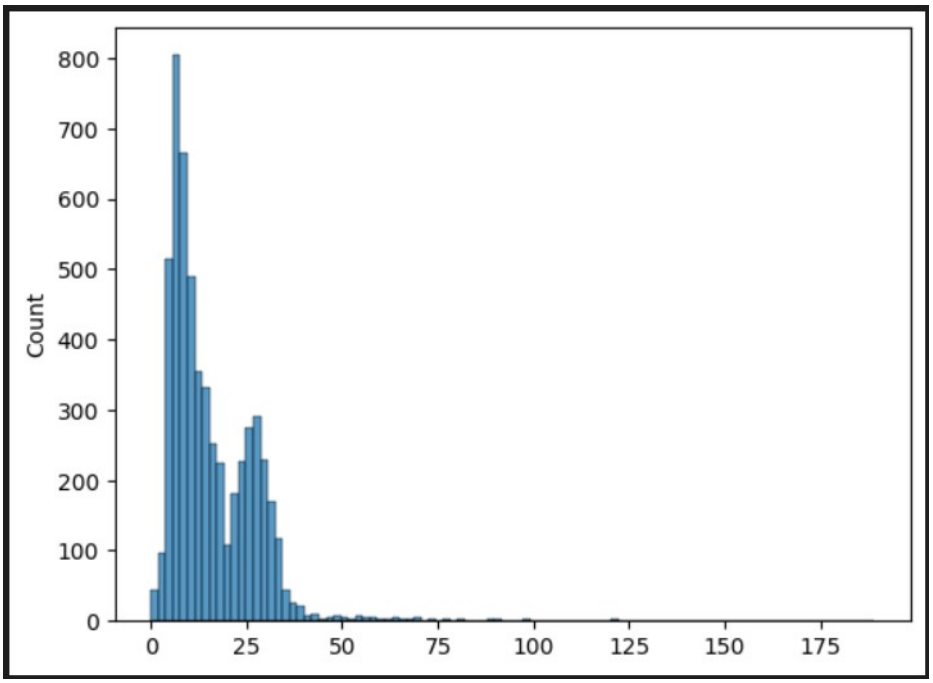

Figura 5: distribuzione per conteggio di parole

Inoltre per il nostro progetto abbiamo eseguito anche una fase riguardante la distribuzione dei conteggi delle parole. Lo svolgimento di questa fase si rende necessario nel momento in cui, nelle fasi successive all'addestramento, si voglia trovare una correlazione tra errori commessi dal modello e la lunghezza dei messaggi.

Individuare outlier per un problema del genere è molto complesso. Dovremmo prima stabilire un criterio di similitudine tra messaggi e poi trovare una correlazione tra gli errori delle predizioni e il fattore di similitudine. Ci limitiamo invece a verificare se esiste una correlazione tra gli outlier rispetto alla distribuzione del conteggio di parole e le predizione errate. Possiamo dire dopo vari controlli che la suddetta correlazione non è presente nel nostro caso.

4.3.2 Feature scaling

La seconda fase è il fueature scaling che sono un insieme di tecniche che consentono di normalizzare o scalare l'insieme di valori di una caratteristica. Il metodo più comune è la *min-max normalization*, un altro metodo è la *z-score normalization*.

Questi metodi però possono essere eseguiti solo su valori numerici. Abbiamo una tecnica ulteriore che ci permette di normalizzare il testo. Un caso tipico é quello di testo scritto in linguaggio naturale. L'analisi del linguaggio naturale è utilizzata in molti contesti e, non a caso, un intero campo dell'Intelligenza Artificiale è dedicato al **N**atural **L**enguage **P**rocessing. Indipendentemente dal contesto, un comune task di NLP segue alcuni step.

Nel nostro progetto avevamo il problema di trasformare testo, espresso in linguaggio naturale, in token numerici comprensibili e interpretabili dal modello.

La prima fase è stata la **segmentazione** in frase del messaggio ricevuto in input. Procedere su una frase per volta semplifica il problema e aiuta a raggiungere risultati migliori sul testo complessivo. Questa è un'operazione semplice, generalmente effettuabile identificando i punti di fine periodo all'interno del paragrafo di testo in analisi.

La seconda fase riguarda la **tokenizzazione**. La tokenizzazione è il processo in cui la frase viene divisa in token, ossia unità del periodo più piccole sulle quali si andrà a lavorare singolarmente. Un token è, generalmente una sotto stringa della frase o una piccola parola. Tale processo può essere effettuato seguendo diverse strategie, come il riconoscimento dei *white space* o particolari segni di punteggiatura. Nel nostro caso il modello ha memorizzato ogni singolo token del dataset come numero.

4.3.3 Feature selection

Per quanto riguarda il feature selection il progettista estrae le **caratteristiche principali** del problema. Identificare buone feature porta molti vantaggi pratici. Il feature selection è quindi il processo tramite il quale vengono selezionate le caratteristiche correlate al problema in esame, lo possiamo fare con l'**eliminazione di feature con bassa varianza** o **con l'eliminizaione univariata di feature**.

In questo progetto non c'è stato bisogno di eseguire nessuna azione di questa fase poichè tutte le feature sono di tipo **categorico** e non numerico.

4.3.4 Data balancing

Molti problemi potrebbero essere sbilanciati, ad esempio quando il numero di pazienti affetti da una malattia è molto inferiore al numero di pazienti non malati. Con il data balancing andiamo a bilanciare il dataset con la tecnica dell'**undersampling**, che sarebbe l'eliminazione di alcune righe appartenti alla classe con un numero di elementi maggiore, oppure con la tecnica dell'**oversampling** possiamo aggiungere nuove righe per la classe che ha meno istanze.

Nel nostro progetto c'è un problema abbastanza importante di data balancing. Come abbiamo potuto vedere, nel grafico inerente alla distribuzione delle etichette, il nostro dataset è sbilanciato. Nonostante questo, non abbiamo eseguito undersampling, perchè rischiavamo di avere troppe poche istanze nella totalità del dataset. Non abbiamo nemmeno aggiunto righe per quanto riguarda i messaggi considerati spam perché non è stato possibile trovare messaggi adatti al nostro scopo. Il nostro modello anche con i suddetti dati ha dato buoni risultati.

4.4 Data Modeling

Una volta sistemati i dati, è ora di iniziare la fase di modellazione, che è sempre particolarmente complicata e "unica", nel senso che la definizione di un algoritmo di machine learning dipende strettamente dal problema in esame e dai dati a disposizione.

In primo luogo, va selezionata la tecnica o l'algoritmo da utilizzare: ad esempio, conviene modellare il problema come un problema di classificazione o regressione? Quale soluzione sarà più adatta e utile al raggiungimento degli obiettivi? Dopodiché, si passa alla fase di addestramento. In questo caso, si configurano i parametri del modello selezionato, si addestra il modello e si descrivono i risultati ottenuti in fase di addestramento.

Molto spesso, il progettista sarà costretto a tornare nella fase di data preparation per effettuare ulteriori operazioni sui dati.

Tra le architetture di apprendimento profondo si annoverano le **deep neural network**, che sono state applicate nella visione artificiale, nel riconoscimento automatico del discorso, nell'elaborazione del linguaggio naturale, nel riconoscimento audio e nella bioinformatica.

In termini pratici le reti neurali sono strutture non-lineari di dati statistici organizzate come strumenti di modellazione. Esse possono essere utilizzate per simulare relazioni complesse tra ingressi e uscite che altre funzioni analitiche non riescono a rappresentare.

Una rete neurale artificiale riceve segnali esterni su uno strato di nodi (unità di elaborazione) di ingresso, ciascuno dei quali è collegato con numerosi nodi interni, organizzati in più livelli. Ogni nodo elabora i segnali ricevuti e trasmette il risultato a nodi successivi.

Una rete neurale artificiale è un modello matematico/informatico di calcolo basato sulle reti neurali biologiche. Tale modello è costituito da un gruppo di interconnessioni di informazioni costituite da neuroni artificiali e processi che utilizzano un approccio di connessionismo di calcolo. Nella maggior parte dei casi una rete neurale artificiale è un sistema adattivo che cambia la propria struttura in base a informazioni esterne o interne che scorrono attraverso la rete stessa durante la fase di apprendimento.

Come nel nostro caso una rete neurale utilizza un algoritmo di **retropropagazione** dell'errore (backpropagation), sarebbero le 15 iterazioni che il nostro modello esegue in fase di apprendimento. Esso permette di modificare i pesi delle connessioni in modo tale che si minimizzi una certa funzione errore **E**. Tale funzione dipende dal vettore h-esimo di output restituito dalla rete, dato il vettore h-esimo di ingresso e dal vettore h-esimo di output che noi desideriamo (che fa parte del training set). L'algoritmo di backpropagation può essere diviso in due passi:

- **Forward pass**: l'input dato alla rete è propagato al livello successivo e così via ai livelli successivi (il flusso di informazioni si sposta in avanti, cioè forward). Si calcola dunque E(w), l'errore commesso.
- **Backward pass**: l'errore fatto dalla rete è propagato all'indietro (backward) e i pesi sono aggiornati in maniera appropriata.

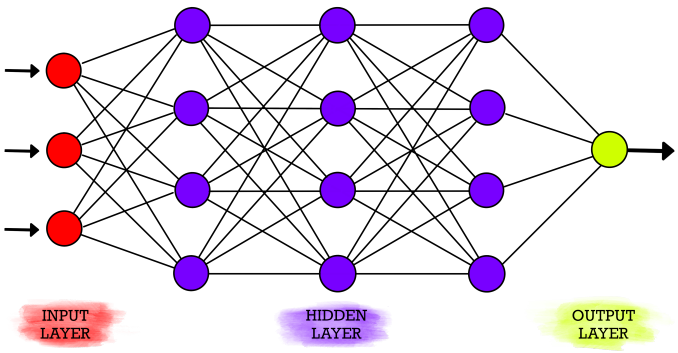

Figura 6: rete neurale

Inoltre, il modello che abbiamo addestrato per questo progetto, può essere definito come "binario" o "a due classi", ciò indica che potrà fornirci risposte di tipo 0 e 1 o nello specifico per il nostro caso, "HAM" e "SPAM".

Una delle scelte effettuate durante la fase di modeling, a grande impatto per le fasi successive, è quella di non arrotondare i valori di output della rete neurale al più vicino intero, dunque, sostituendo una più ovvia funzione di attivazione a passo unitario con una **funzione sigmoidea**, andando successivamente ad effettuare una arrotondamento con una funzione esterna alla rete.

Questa scelta è stata particolarmente influenzata dal vantaggio di poter, in un ambiente in cui il modello non dovrà essere distribuito ma valutato, utilizzare le classificazioni per effettuare valutazioni, non solo binarie, le quali ci fornirebbero solo informazioni del tipo "Classificazione Sbagliata" o "Classificazione Corretta", piuttosto avere anche una misura di quanto il modello sia sicuro della sua scelta. Inoltre la funzione sigmoidea ci consente di evitare che le valutazioni del modello si vadano a concentrare in valori vicini a 0.5, relativi a casi di "estrema indecisione".

Per la scelta della conformazione della rete neurale, ci siamo basati principalmente su ciò che Tensorflow suggeriva di utilizzare per problemi di questo calibro, andando poi ad effettuare piccole variazioni volte all'incremento dell'efficacia del nostro classificatore.

4.5 Evaluation

La fase di validazione ha l'obiettivo di valutare se i risultati sono chiari, se sono in linea con gli obiettivi di business e se rivelano delle prospettive aggiuntive alle quali il progettista non aveva pensato.

In questa fase, è inoltre importante verificare la consistenza e la solidità dell'intero processo. Ad esempio, ci sono degli aspetti che non sono convincenti? Ci sono delle alternative metodologiche che potrebbero portare a risultati diversi? E come queste alternative impattano i risultati ottenuti?

Una volta avute le risposte, si può procedere alla definizione dei prossimi passi da effettuare. Possiamo considerare la definizione dell'approccio completa? Oppure è necessario fare un passo indietro e valutare opzioni diverse?

Il seguente grafico mostra l'andamento dei miglioramenti ad ogni iterazione dell'addestramento che abbiamo eseguito. Per dei risultati soddisfacenti abbiamo deciso che quindici iterazioni sarebbero state abbastanza per l'addestramento dei nostri dati. Infatti come possiamo vedere dal grafico, che mostra sull'asse delle x il numero di iterazioni fatte dal modello e sull'asse delle y il valore che indica una penalità legata a una predizione sbagliata, c'è un netto miglioramento, anche se ci sono tante cose da precisare. La linea di colore arancione indica la funzione di perdita relativa ai dati di validazione che sarebbero una porzione dei dati di training. La linea blu, invece, indica la funzione di perdita per i dati di training.

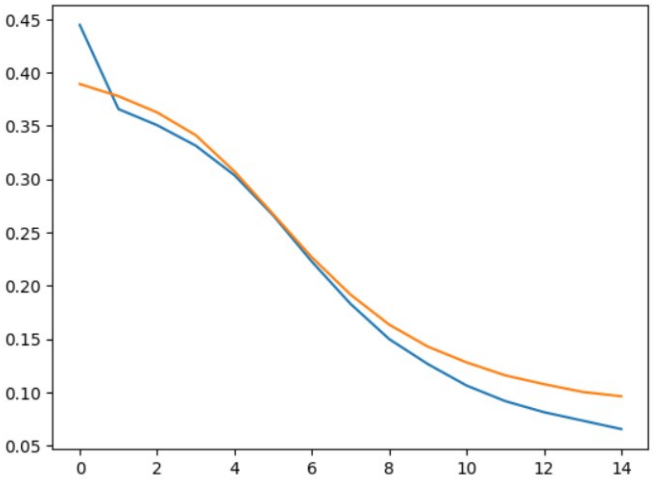

Figura 7: andamento miglioramenti

Il nostro modello è stato addestrato, con dei risultati apparentemente molto soddisfacenti. Ciò però NON significa necessariamente che il suo comportamento sia quello da noi desiderato. Adesso vedremo perché.

4.5.1 Matrice di confusione

Si rende molto utile in questa fase di comprensione dei dati, l'utilizzo di una **matrice di confusione** per una rappresentazione dei *risultati ottenuti dal modello*. Calcoliamo quindi per ogni etichetta quante sono le entry per le quali le etichette sono state correttamente predette dal modello nel **dataset di test** e quali invece sono state sbagliate.

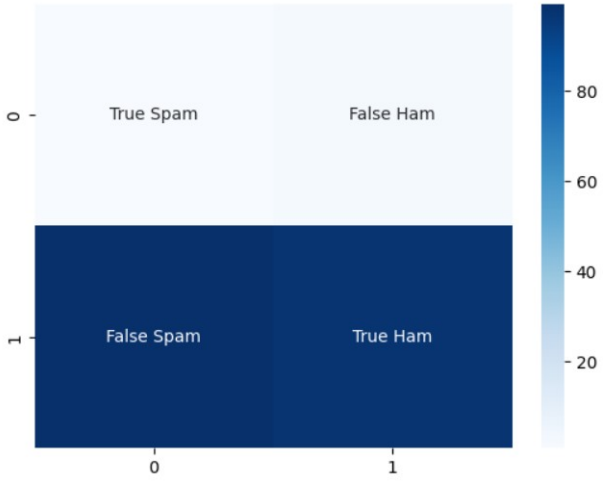

Figura 8: matrice di confusione

Seppure il modello abbia totalizzato un'accuratezza molto alta, andando a visualizzare in dettaglio i risultati, si nota che il modello è più bravo a riconoscere casi di non-spam piuttosto che casi di spam. Questo fattore potrebbe assolutamente essere riconducibile allo sbilanciamento delle etichette. Il nostro modello ha quindi problemi di **overfitting** e **data leakage**.

In pratica avendo a disposizione molte più istanze di massaggi ham e quindi essendo molto più allenato a riconoscere i messaggi ham che quelli spam, anche in fase di valutazione tende a riconoscere molto bene questo tipo di messaggi che quelli spam.

Inotre il modello crede di fare anche predizioni molto accurate perché avendo molte istanze di messaggi non-spam ne indovinerà tante e sbaglierà poche spam non rendendosi conto che dovrebbe riconoscere proprio quelle.

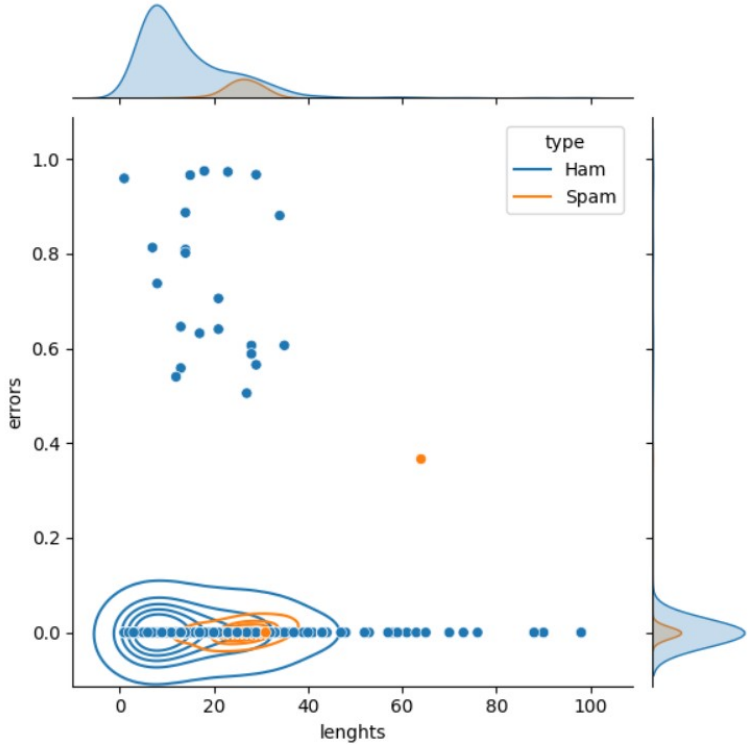

Figura 9: grafico errori

Come possiamo ben notare dal **grafico degli errori** il nostro modello fa molte predizione esatte per quanto riguarda i messaggi ham e ne sbaglia pochissimi. Contemporaneamente sbaglia molto spesso la predizione dei messaggi spam, in poche parole, molto spesso, categorizza i messaggi spam come ham.

Da questo grafico possiamo notare anche la mancata correlazione tra gli outlier rispetto alla distribuzione del conteggio di parole e le predizione errate.

4.6 Deployment

La fase di deployment ha l'obiettivo di mettere in funzione l'approccio definito e, quindi, renderlo usabile.

Il modello generato, precisamente, come dovrà essere reso disponibile? Quale sarà il grado di interazione con gli utenti? Ed in che modo gli utenti utilizzeranno il modello? In altri termini, questa fase vede il passaggio dall'ingegneria del machine learning all'ingegneria del software e all'ingegneria dell'usabilità!

Questo è ancora più evidente se consideriamo che questo modello non resterà nel suo stato in eterno e avrà bisogno di essere costantemente monitorato e manutenuto! Possiamo citare le prime due leggi di Lehman sull'evoluzione di sistemi software: (1) un sistema che non cambia è un sistema che diventerà presto inutile; (2) la complessità del sistema crescerà inesorabilmente nel tempo.

Il modello in questione può essere usato per vari scopi. Oggi questo tipo di modelli è usato tantissimo per il controllo delle mail spam. Questo però è solo il più comune tra gli usi. Possiamo pensare al controllo delle chiamate spam o meno, oppure per tutelare le carte di credito che ormai sono di uso quotidiano per tutti noi. Inoltre, il modello può essere usato per filtrare qualsiasi tipo di messaggio, in un ambiente in cui non ci interessa tenere conto di chi sia il mittente. Un caso reale potrebbe essere l'applicazione in ambienti di blogging o social, in cui si vuole effettuare una prima selezione di messaggi potenzialmente SPAM, i quali dovranno essere supervisionati da amministratori del sistema prima di essere resi pubblici

5 Conclusioni

5.1 Possiamo fare di meglio?

Nonostante l'elevata accuratezza del modello nella classificazione di dati provenienti dal Dataset utilizzato per l'apprendimento, il suo comportamento presenta molti dubbi nel caso in cui lo si utilizzi per predirre l'etichetta di nuove stringe (magari scritte a mano). Questo comportamento potrebbe essere una delle conseguenze della poca numerosità dei dati. Dopo aver reso visibili pregi e difetti del nostro machine learning possiamo dire che i risultati sono soddisfacenti ma non del tutto. Sicuramente, come abbiamo già detto, la causa principale sono i dati sbilanciati da noi posseduti e visto che questo progetto è nato soprattutto per sconfiggere il problema dei messaggi considerati spam possiamo dire che si può fare nettamente meglio.