Medical Diagnosis Dialogue

Michele Montano
Università degli studi di Salerno
m.montano8@studenti.unisa.it

Ferdinando Napolitano
Università degli studi di Salerno
f.napolitano56@studenti.unisa.it

***Abstract*—L'intelligenza artificiale risulta essere uno strumento di supporto nell'ambito medico moderno, in particolare nell'accelerazione dei tempi di diagnosi di malattie. Una delle sfide più grandi in questo ambito è la richiesta di sistemi che siano il più affidabili possibile, insieme alla difficoltà di reperire dati. In questo paper, presentiamo un sistema di diagnosi automatica basato su un algoritmo di reinforcement learning in grado di riconoscere fino a 12 malattie tramite i sintomi inseriti dall'utente e quelli ricavati dalla successiva interazione con esso. Inoltre, presentiamo lo studio di 3 modelli supervisionati (SVC, Random Forest, KNN) sullo stesso dataset al fine di un confronto tra due metodologie differenti.**

La prima fase è stata quella di effettuare un tuning dei parametri dell'algoritmo actor critic al fine di ottenere un sistema che fosse il più accurato possibile. Successivamente abbiamo sviluppato un modulo che, tramite tecniche di NLP, è in grado di dialogare con l'utente, in modo da ottenere man mano sempre più informazioni riguardo possibili sintomi. Il bot finale è quindi risultato un sistema modulare con una parte dedicata all'interazione e l'estrazione dei sintomi ed una dedicata all'elaborazione di queste informazioni.

Per i modelli supervisionati l'approccio è stato differente, abbiamo inizialmente proceduto a trasformare il dataset da una struttura a dizionario ad una tabellare trasformando ogni campione tramite la tecnica di one hot encoding in una stringa contente 1 se il sintomo era presente, 0 altrimenti. Questa fase è stata necessaria in quanto questo tipo di modelli non sono in grado di lavorare con feature categoriche. Successivamente utilizzando la tecnica della grid search abbiamo ricercato i migliori valori di ogni modello che me minimizzassero la funzione di loss, e sottoposto ognuno ad una fase di validazione per verificare che non presentassero problemi di overfitting.

Dalla fase di testing si è evinto che il bot presenta un' accuracy di circa il 71%, questo risultato è dovuto al fatto che il bot in presenza di sintomi comuni a più malattie risulta fare errori di classificazione più frequenti. Per patologie con sintomi piuttosto specifici e non presenti in nessun altro campione di malattie differenti, il bot risulta essere invece molto più accurato.

Per quanto riguarda i risultati dei modelli supervisionati, questi presentano valori di accuratezza maggiore, (RF: 82.5%, SVC: 81.4%, KNN: 74.6%, VotingClassifier: 84.6%), nonostante anche in questo caso gli errori commessi dai classificatori sono dovuti ai sintomi in comune a più malattie.

I. Introduzione

La diagnostica automatica nasce come uno strumento di supporto ai medici, in particolare nell'individuazione di tumori a partire da immagini radiologiche, spesso soggette ad un esame attento da parte del medico prima di una diagnosi corretta. In questo senso l'obiettivo di questi strumenti è di accelerare le tempistiche diagnostiche nel campo oncologico. Negli ultimi anni la diagnostica automatica si è estesa ad altri campi medici, in particolare nel riconoscimento di patologie tramite la sintomatologia caratterizzante. Attualmente, i moderni sistemi di diagnosi automatica sono single purpose ovvero si concentrano sulla diagnosi di singole malattie o a volte su gruppi di 3-4 patologie che condividono sintomi simili. Questi sistemi, inoltre, prevedono l'estrazione manuale dei sintomi dalle cartelle cliniche del paziente da parte del medico in modo da utilizzarli per fornire una diagnosi. L'obiettivo del nostro studio, quindi, è stato quello di provare ad estendere l'utilizzo di questi strumenti per applicare la diagnosi automatica non ad una singola patologia (o un gruppo ristretto) ma ad una serie di 12 patologie diverse sia per cause che per sintomi. Questa condizione porta ovviamente ad un aumento della difficoltà di riconoscimento da una decina di sintomi a circa un centinaio. Inoltre, tramite la conversazione di un bot con l'utente si vuole sostituire la fase di estrazione dei sintomi manuale da parte del medico. Queste modifiche permettono ai sistemi di diagnosi classici di avvicinarsi sempre di più al paziente, fornendo, in un certo senso, un supporto immediato in caso di necessità in modo da velocizzare gli eventuali trattamenti di cui necessita. Lo scopo della diagnostica automatica non è quindi quello di sostituire il medico ma fornire strumenti che li aiutino nell'accelerare le tempistiche delle diagnosi. Rispetto alle metodologie classiche, quindi, si vuole introdurre uno strumento più "general purpose" che possa portare benefici non solo al medico, ma anche ad un paziente che sottoponendo al medico le risposte del bot potrebbe facilitare le modalità di diagnosi e velocizzare la ricezione di un'eventuale cura.

II. Stato dell'arte

Uno studio recente nel campo della diagnostica predittiva medica è stato quello di Quan Zhang et al [1] finanziato dal National Natural Science Foundation of China in cui i ricercatori hanno utilizzato un algoritmo di deep learning per la diagnosi automatica della iperlipidemia attraverso informazioni relative ai parametri delle urine e del sangue. Dalla fase di testing è emerso che l'algoritmo ha ottenuto il 91.49% di accuratezza e l'87.5% di precisione. L'utilizzo dei parametri ematologici oltre a quelli relativi all'analisi delle urine ha permesso di ottenere diagnosi più accurate rispetto agli studi precedenti. Inoltre, poiché il modello lavora direttamente sulle feature grezze non è necessario l'intervento del medico nella fase di estrazione dei parametri dalle cartelle cliniche, il che velocizza le procedure di diagnosi della ipelipidemia. Recentemente, altri team di ricerca come Kermany et al [2], hanno utilizzato il modello Inception V3 addestrato da Google insieme ad un algoritmo di transfer learning per diagnosticare la

retinopatia e la polmonite infantile attraverso, rispettivamente, immagini OCT della retina e raggi X. Infine, vi sono i risultati della competizione organizzata dall'ente ICLR, il quale ha fornito la baseline ed il dataset costruito in associazione con il team SYSU-Med [3]. Su questo esperimento sono state effettuate diverse sperimentazioni dai vari team, i cui risultati sono noti solo in termini di accuratezza nella predizione delle malattie, senza però dettagli riguardo il loro sviluppo (ad esempio i parametri utilizzati per il modello e dettagli sulla predizione delle singole malattie). Detto questo, il cuore del nostro sistema è stata la baseline su cui abbiamo apportato leggere modifiche e costruito poi la restante parte del bot. Per valutarne i risultati abbiamo considerato, come metriche di valutazione, l'accuratezza nella predizione della malattia in modo da poterci confrontare con quelle dei team partecipanti alla competizione.

III. Dataset

Il dataset è composto da campioni rappresentati come dizionari python con 3 chiavi: "explicit inform slots" in cui sono presenti tutti i sintomi forniti dall'utente, "implicit inform slots" in cui sono presenti i sintomi richiesti al paziente e "disease tag" ovvero la malattia associata al campione.

```
{
 'explicit_inform_slots' : {'Acid reflux':True, 'Vomiting':True'}
 'implicit_inform_slots' : {Stomach ache:False}
 disease_tag' : 'Esophagitis'
}
```

Fig. 1. Esempio di un campione del dataset.

Il dataset presenta 12 classi ognuna con un numero di campioni associato piuttosto bilanciato, circa 160 ed una percentuale di campioni duplicati pari al 41.6%; ciò è dovuto probabilmente al fatto che ogni malattia ha un ristretto numero di sintomi che si ripetono in quasi tutti i campioni di quella classe. In questo senso i campioni associati ad una malattia si distinguono solo in presenza di sintomi che hanno una minore frequenza ma comunque presenti.

Utilizzando dei grafici a barre rappresentanti la frequenza dei sintomi per ogni malattia abbiamo notato come un gruppo ristretto di 3-5 sintomi fosse presente in tutti i campioni di quella classe; inoltre alcune malattie come l'esofagite e l'enterite presentano sintomi in comune; stessa situazione può essere riscontrata tra l'asma e la polmonite, al contrario della dermatite che presenta sintomi unici solo per quella classe.

Nella fig. 2 vi è una visualizzazione di campioni del dataset appartenenti a 3 classi diversi, in particolare "Coronary heart disease", "Traumatic brain injury", "Thyroiditis". La visualizzazione è stata effettuata tramite la libreria yellowbrick la quale utilizza tecniche di riduzione della dimensionalità per rappresentare dataset con centinaia di feature.

IV. Preprocessing per i modelli supervisionati

Il dataset è stato trsformato in modo da renderlo compatibile con l'addestramento dei modelli supervisionati scelti attraverso la tecnica di One Hot Encoding ottenendo un dataset con 118

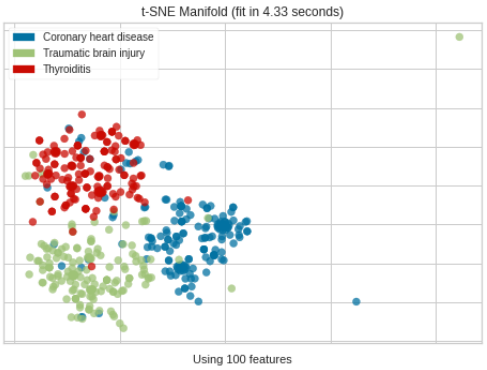

Fig. 2. Esempio della visualizzazione di 3 classi di malattie.

feature, una per ogni sintomo ed un'ultima colonna per la label contenete il nome della malattia. La scelta di questo tipo di encoding è data dal fatto che i dati su cui i modelli devono essere addestrati devono essere di tipo numerico e non categorico; in questo modo i modelli ricevono in input una stringa sparsa con "1" se il sintomo è presente nel campione, "0" altrimenti; inoltre questo tipo di encoding, in generale, potrebbe migliorare l'accuratezza dei modelli nella classificazione delle patologie e la loro capacità di generalizzazione. Poiché il dataset risulta essere piuttosto bilanciato, circa 160 campioni per ogni malattia, abbiamo proseguito verificando l'evenutale mancanza di dati sui sintomi. Abbiamo notato che tra i 118 sintomi, 18 erano sprovvisti di alcun tipo di informazione (tabella I), il che avrebbe portato ad avere un dataset con 18 colonne completamente nulle. Inoltre, tramite una sperimentazione dedicata, abbiamo constatato che l'eliminazione di queste colonne porta un incremento di circa 2-3 punti percentuali nell'accuratezza dei modelli.

TABLE I
RISULTATI

Sintomi eliminati
Twitch
Hematemesis
Hematuria
Painful urination
Radiating pain
Cyanosis
Mucus pus and blood in the stool
Tensile and heavy
Inspiratory tri-concave sign
Poor physical activity
Ulcer
Hand tremor
Proptosis
Incontinence
Limb numbness
Dysmenorrhea
Vaginal bleeding
Increased vaginal discharge

V. Modelli supervisionati

Inizalmente, abbiamo proceduto con un tuning dei parametri manuale, cosa che ha richiesto tempo e non ha fornito risultati soddisfacenti (accuratezza tra il 30-50%). Per questo motivo, per tutti i modelli, abbiamo effettuato una ricerca esaustiva dei migliori parametri in un insieme specificato tramite il metodo GridSearchCV. L'obiettivo era quello di massimizzare la capacità di generalizzazione dei modelli, trovando quei parametri che minimizzassero una data funzione di loss (nel nostro caso la negative log loss). Questo metodo permette di addestrare i modelli con tutte le combinazioni di parametri specificati fornendo in output gli iperparametri che minimizzano lo score specificato. Successivamente abbiamo proceduto ad una fase di validazione dei modelli attraverso una 5-fold cross validation utilizzando gli iperparametri ricavati dalla fase precedente, con l'obiettivo di verificare che i modelli non soffrissero di problemi di overfitting o underfitting. Il numero di fold è stato scelto pari a 5 in quanto un numero maggiore di fold avrebbe avuto come conseguenza un test set di piccole dimensioni e poco significativo rispetto al training set. Per ogni modello sono mostrate le curve di apprendimento e di loss, oltre ai parametri ottenuti dalla grid search. In un successivo capitolo saranno poi discussi in dettaglio i risultati ottenuti.

A. Random Forest

Il classificatore Random Forest è stato scelto essendo uno dei modelli migliori per gestire dati ad alta dimensionalità, nel caso specifico ci si riferisce alle feature del dataset, ovvero tutti i possibili sintomi. Il Random Forest addestra un numero di classificatori del tipo "decision tree" su diversi sotto-esempi del dataset fornito e usa la media dei risultati di questi estimatori per migliorare l'accuracy e il controllo dell'overfitting. La figura 3 mostra la curva di apprendimento del RF con un andamento crescente fino a raggiungere una media di accuratezza pari all'82%. La loss invece (fig. 4) presenta un'andamento decrescente fino ad un valore di circa 0.79 e successivamente risulta altalenante fino ad un valore medio dello 0.75.

TABLE II
GRID SEARCH RF

n estimators	*max depth*	*max features*	*min sample split*
1000	35	sqrt	10

B. Support Vector Classifier

Il support vector classifier, così come per il random forest, è stato scelto per le buone performance su dataset con molte feature. La tabella III mostra i migliori parametri ottenuti dalla grid search, in particolare il modello presenta un kernel di tipo rbf con un coefficiente (gamma) pari a 0.1 ed un parametro di regolarizzazione (C) con un valore pari a 4. Come detto nell'introduzione di questo capitolo, prima di procedere nella fase di training e test abbiamo sottoposto il modello ad una fase di validazione tramite una 5-fold cross validaton. La

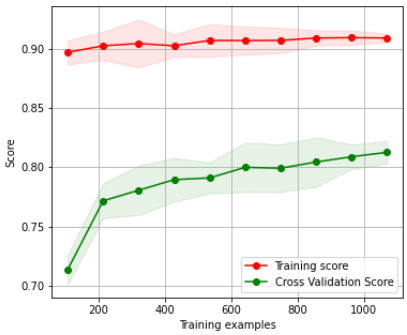

Fig. 3. Curve di apprendimento Random Forest.

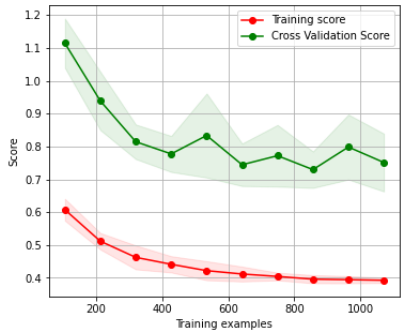

Fig. 4. Curve di loss Random Forest.

figura 5 mostra l'andamento della curva di apprendimento del modello SVC la quale ha un andamento crescente rispetto al numero dei campioni sottoposti al modello fino a raggiungere un valore medio dell'80%. Così come per il random forest, è da notare, che l'aggiunta di nuovi campioni potrebbe portare ad un ulteriore aumento nella crescita della curva con un conseguente aumento dell'accuratezza del modello. La curva di loss nella figura 6 invece presenta un andamento decrescente fino ad un valore medio dello 0.7 e poiché la diminuzione della loss di un modello porta a performance di generalizzazione migliori, anche in questo caso, visto l'andamento, possiamo dire che incrementando la mole di campioni nel dataset potrebbe essere possibile decrementarne il valore di loss.

TABLE III
GRID SEARCH SVC

kernel	*C*	*gamma*
rbf	4	0.1

C. K Nearest Neighbours & Voting Classifier

Il K Nearest Neighbours è stato scelto come terzo classificatore in modo da effettuare un confronto con i due classificatori precedenti, ma anche per realizzare un classificatore di tipo ensemble, ovvero il soft voting classifier. L'obiettivo era di verificare se la combinazione dei 3 classificatori fornisse dei risultati migliori rispetto alla loro applicazione in singolo. Nella figura 8 le curve di apprendimento del KNN risultano quasi sovrapposte e tendono ad appiattirsi verso un valore

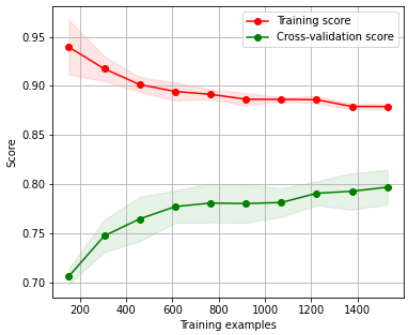

Fig. 5. Curve di apprendimento SVC.

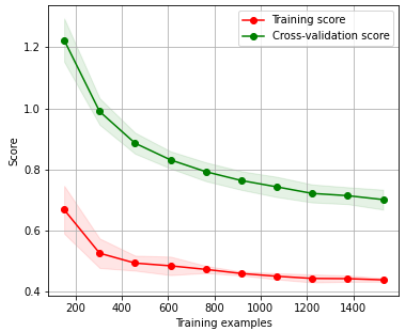

Fig. 6. Curve di loss SVC.

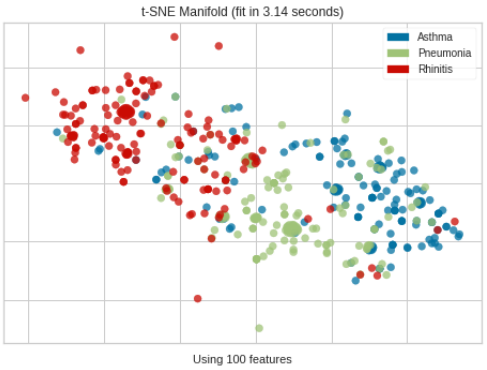

Fig. 7. Visualizzazione di 3 classi di malattie

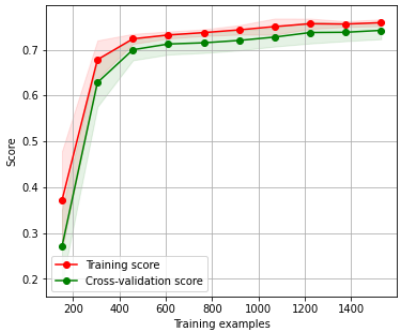

Fig. 8. Curve di apprendimento KNN.

medio di 75%. Un valore più basso rispetto ai due modelli analizzati in precedenza è probabilmente dovuto al fatto che i k vicini (in questo caso 82) utilizzati dal modello per classificare un dato campione, presentano in molti casi la presenza di più malattie diverse ma con alcuni sintomi in comune. Questo può essere constatato osservando la visualizzazione in 2 dimensioni di 3 classi di campioni diverse (fig. 7), in quanto non ci si trova di fronte a cluster ben definiti ma al contrario, molti campioni dell'asma si ritrovano ad essere molto vicini a campioni della polmonite poiché le due patologie presentano alcuni sintomi in comune. Questo risultato si riflette anche nella curva di loss (fig. 9) la quale, nonostante un andamento decrescente, presenta al termine dell'addestramento un valore medio di 1.2, più alto rispetto alla media dei valori ottenuti dagli altri modelli.

TABLE IV
GRID SEARCH KNN

neighbours	*metric*	*weights*	*algorithm*	*leaf size*
82	euclidean	uniform	ball_tree	50

Il primo voting classifier è stato costruito utilizzando i 3 modelli visti in precedenza, attraverso il meccanismo del voto a maggioranza. A causa della presenza del KNN, questo classificatore ha ottenuto valori di accuratezza e precisione media inferiori ai singoli classificatori, seppur maggiori ai valori ottenuti dal modello eliminato. Da qui la decisione di costruire un secondo voting classifier utilizzando solamente i modelli random forest e SVC, con l'obiettivo di effettuare un confronto con il primo voting classifier, rimuovendo il classificatore KNN il quale ha riportato performance nettamente inferiori rispetto ai primi due. Come ci aspettavamo, i risultati ottenuti sono stati decisamente migliori rispetto a quelli ottenuti con il primo voting classifier, ottenendo, durante la fase di test, valori di accuratezza e precisione più alti dei singoli modelli partecipanti. La figura 10 mostra come la curva di apprendimento presenta un'andamento crescente simile a quello dei modelli random forest ed SVC con un valore medio di accuratezza pari a 80.2%. Per quanto riguarda la loss, il voting classifier è stato il modello che ha fornito il valore di loss più basso rispetto ai singoli classificatore presi in esame, in particolare un valore medio pari a 0.65.

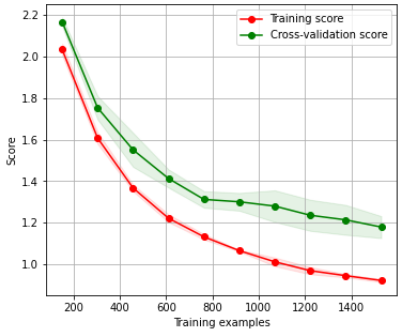

Fig. 9. Curve di loss KNN.

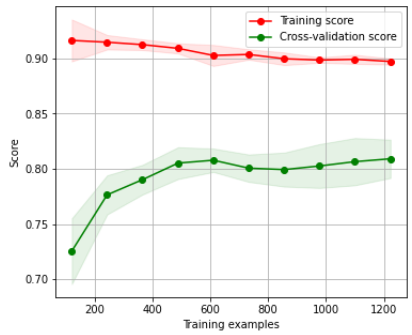

Fig. 10. Curve di apprendimento Voting classifier.

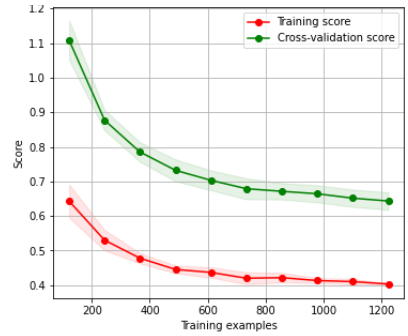

Fig. 11. Curve di loss Voting classifier.

VI. Architettura Medical Bot

Come accennato nell'introduzione il nostro bot presenta un'architettura modulare; in particolare, vi è un modulo dedicato all'interazione con l'utente, il cui compito è quello di ricevere le richieste, elaborarle ed estrarre i sintomi per procedere alla diagnosi. Il secondo modulo è invece predisposto per il processamento dei dati in modo da elaborarli e mandare al modulo di dialogo la prossima risposta del bot. Questo secondo modulo risulta essere la baseline fornita dall'ente ICLR.

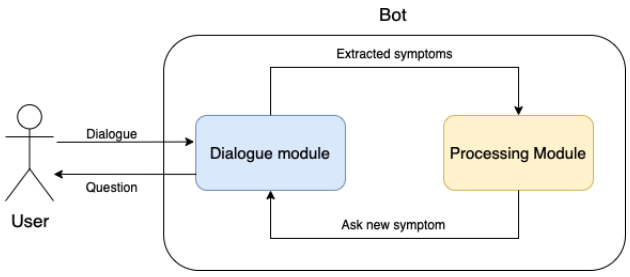

Fig. 12. Architettura del bot.

A. Baseline

La baseline è stata costruita utilizzando Tianshou, una libreria di supporto per il reinforcement learning insieme a diverse componenti create appositamente per il dominio del problema:

- ActorNet: la rete neurale utilizzata per il modulo Actor durante l'addestramento. Permette, tramite la modifica della variabile "least_ask", di modificare il numero minimo di richieste che l'actor può effettuare. L'architettura della rete prevede un layer di input, uno di output e 2 hidden layer, ognuno di 128 neuroni.
- Policy: gestisce tutto ciò che riguarda la politica dell'agente; in particolare, effettua un'azione a partire da un'osservazione fornita dall'ambiente. La policy viene aggiornata costantemente tramite batch di dati prelevati dal replay buffer, una struttura contenete tutte le azioni effettuate in precedenza con la relativa osservazione.
- Collector: è un modulo che permette alla policy di interagire con l'ambiente, si occupa di resettarlo, applicare le azioni della policy e fornire informazioni sulle osservazioni.
- Environment: L'ambiente consiste in un vettore di 130 elementi, dove le prime 12 posizioni sono occupate dalle malattie e le restanti 118 dai sintomi. Al primo step l'ambiente è inizializzato con 3 possibili valori:
  - 0 se il sintomo non è presente (non si ha informazione su quel sintomo)
  - 1 se l'informazione sul sintomo risulta essere positiva (il paziente dichiara di avere quel sintomo)
  - -1 se l'informazione sul sintomo risulta essere negativa (il paziente dichiara di non avere quel sintomo)

  Successivamente l'ambiente è aggiornato con i valori 1 e -1 in base alle richieste effettuate dall'algoritmo. Lo stato dell'ambiente ed il reward per quello step vengono forniti al modello in modo da ottenere l'azione da effettuare per lo step successivo.

Il nostro lavoro, dopo aver descritto l'architettura della baseline, è stato modificare alcune delle sue componenti per rendere migliore l'interazione con l'utente che, altrimenti, sarebbe stata meno naturale e meno precisa nei risultati. Il primo passo, quindi, è stato andare a modificare il lato dell'ambiente che si occupa della predizione della malattia, nel file MaskEnvironment.py. In particolare abbiamo implementato la funzione render(), andando ad aggiungere una interfaccia che permetta al bot di chiedere all'utente se presentasse un determinato sintomo e, in base alla risposta ("sì" o "no"), procedere alla prossima predizione e a nuove richieste fino a mostrare la malattia. Di base, questo metodo non era implementato e non permetteva di far interagire il sistema con l'utente, poiché lo scopo per cui la baseline è stata pensata è solamente quello di addestrare l'algoritmo addestrato attraverso campioni pre-strutturati in input. Il secondo passo, invece, è stato modificare la variabile "least_ask" presente nel file ActorNet per poter permettere un miglior comportamento del bot nei confronti dell'utente e ottenere anche risultati migliori in termini di accuratezza. Questa variabile, infatti, in base al valore assegnatole (intero), indica al modello il minimo numero di richieste di sintomi da sottoporre all'utente. Il valore di default di questa variabile era 4, ma al fine di ottenere più informazioni e di conseguenza ottenere una

maggiore precisione sulla malattia predetta abbiamo deciso di incrementarla a 7, ottenendo in media un numero di richieste che varia da 7 a 13. Questa decisione è stata supportata da test effettuati comparando le performance dell'agente variando soltato questo parametro; in particolare abbiamo notato un aumento notevole nell'accuratezza della classificazione della patologia al termine delle richieste. Il numero di epoche è stato impostato a 60 in quanto abbiamo notato che un numero maggiore non portava alcun miglioramento in termini di accuratezza nella predizione delle malattie. Il parametro "step per epoch" indica il numero di step per aggiornare la policy in una singola epoca; il parametro "collect per step" indica il numero di episodi che il collector ottiene prima dell'aggiornamento della policy. Valori piccoli per questi parametri comportano tempi di apprendimento decisamente rapidi ma con scarse performance da parte del modello; per questo motivo, effettuando vari test, i valori scelti per questi due parametri sono stati 512 e 128 rispettivamente, in quanto aumentandone il valore il modello presentava una loss molto più altalenante e valori di accuracy che variavano di molto durante la fase di testing. Infine la batch size è stata incrementata da 64 (valore di default) a 128 in modo da fornire più campioni da propagare attraverso la rete della policy.

TABLE V
PARAMETRI A2C

Epoch	*Buffer size*	*learning rate*
60	20480	5e-4

TABLE VI
PARAMETRI A2C

Step per epoch	*Collect per step*	*Batch size*
512	128	128

B. NLP & GUI

Il nostro bot, per comprendere le richieste utente utilizza tecniche di Natural Language Processing, un campo di ricerca interdisciplinare che abbraccia informatica, intelligenza artificiale e linguistica, il cui scopo è quello di sviluppare algoritmi in grado di analizzare, rappresentare e quindi "comprendere" il linguaggio naturale, scritto o parlato, in maniera similare o addirittura più performante rispetto agli esseri umani. Nel nostro caso si tratta di un task di keyword extraction, in quanto si vuole far in modo che il bot possa rispondere in modo corretto e coerente in base alla richiesta dell'utente. Nel nostro caso, quindi, il processo analizzerà le richieste e le dividerà in base ai tag (ovvero la "famiglia") a cui appartiene, in modo da far fornire al bot una risposta il più consona possibile. Questi sono del tipo:

- Benvenuto
- Saluti
- Ricerca della malattia

Per l'estrazione automatica abbiamo utilizzato una rete neurale artificiale composta da 3 layer, più precisamente, un layer input pari 29 nodi (ovvero dimensione del dataset dei pattern usati per il riconoscimento delle richieste divise per tipo da un tag, escluse le parole duplicate), un layer hidden con 8 nodi di input e un layer output con numero pari al numero di tag di riconoscimento come output. Ogni predizione ha una sua percentuale di affidabilità e viene scelta quella con valore maggiore e, in base al tipo di richiesta che corrisponde a quelle del documento, fornisce una risposta. I pattern usati per il riconoscimento delle richieste utente sono stati implementati all'interno del file json IntentIlls e, come detto in precedenza, ristrutturati in dataset per addestrare la rete in modo valutare le richieste utente.

Per quanto riguarda l'interfaccia utente abbiamo utilizzato la libreria tkinter per realizzare una semplice chat in cui l'utente può dialogare con il bot e far partire il processo di diagnosi mostrata in fig. 13. Se durante la conversazione il bot riconosce che l'utente ha fornito dei sintomi, inizierà a chiederne altri tramite una dialogue box con due possibili risposte: "si", "no".

Al fine di una migliore comprensione del processo di diagnosi, descriviamo brevemente un esempio di una possibile interazione con l'utente. Inizialmente al bot vengono sottomessi i sintomi: "cough", "fever" e "shortness of breath" attraverso la richiesta dell'utente. Tramite tecniche di analisi del linguaggo viste in precedenza il bot riconosce che l'utente ha effettuato questo tipo di richiesta, estrae i sintomi espressi e struttura un dizionario che sarà poi sottoposto al modulo dedicato all'elaborazione delle richieste per ottenere il primo (o prossimo) sintomo da chiedere. Infine, dopo aver terminato le richieste dei sintomi il bot mostra nella chat la malattia predetta.

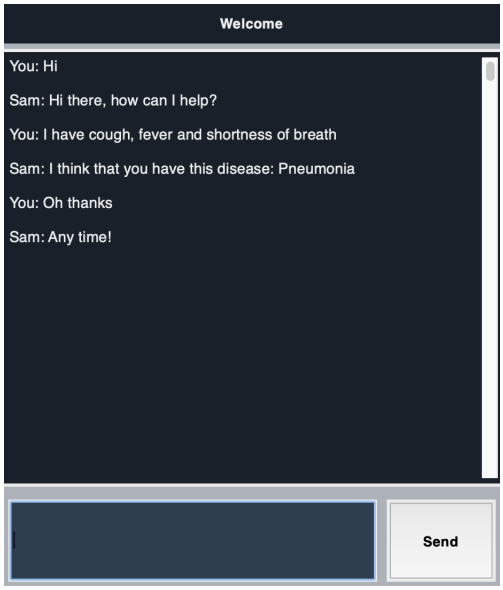

Fig. 13. Interfaccia del bot.

VII. Risultati

In questo capitolo tratteremo i risultati ottenuti durante la fase di train e test dei modelli supervisionati e della fase di addestramento del bot basato sull'algoritmo actor critic.

A. Modelli supervisionati

Per quanto riguarda SVC e RF, risultano ottenere al termine della fase di test un'accuratezza pari all'81.4% e 82.5% rispettivamente, mentre per il KNN otteniamo un'accuratezza decisamente inferiore, ovvero pari al 74.6%. Osservando le matrici di confusione del random forest e SVC, questi risultano avere comportamenti simili nella classificazione di campioni appartenenti alla stessa classe, seppur con alcune eccezioni: ad esempio il classificatore SVC riesce a classificare meglio patologie come la tiroidite e la dermatite. Il random forest invece riesce a distinguere meglio la mastite e l'enterite rispetto all'SVC. Le performance inferiori del KNN, come detto precedentemente, sono probabilmente dovute sia alla struttura dei campioni, sia al funzionamento del modello stesso. Un esempio di questo comportamento può essere verificando osservando la fig. 7 in cui molti campioni della rinite si trovano vicini a quelli dell'asma; ciò si riflette anche nella matrice di confusione (fig. 16) in cui campioni della rinite sono erroneamente classificati come asma. Bisogna notare però che questo tipo di errori di classificazione, sono presenti anche nel random forest e SVC, ma con un impatto decisamente minore. Infatti, in generale, le malattie che hanno messo più in difficoltà i 3 modelli sono state l'asma, la dermatite e la rinite, a causa di quei sintomi che spesso sono presenti in più di un campione di patologie differenti.

TABLE VII
Risultati

Modello	***Accuracy***	***Precision***	***Recall***
Random Forest	82.5%	83%	82.5%
SVC	81.4%	81.5%	81.1%
KNN	74.6%	77.7%	75.8%
Soft Voting Classifier	84.6%	84.7%	84.6%

Per quanto riguarda il voting classifier, presenta una percentuale di classificazione pari all'84.6%, migliore rispetto ad ogni singolo modello studiato. La fig. 14 mostra la matrice di confusione ottenuta dal testing del modello utilizzando solo SVC e random forest, in quanto, con l'aggiunta del modello KNN, abbiamo ottenuto percentuali di accuratezza, precisione e recall inferiore di circa il 5%, dovuto a performance nettamente inferiori da parte del modello KNN rispetto agli altri due. Detto ciò il voting classifier sarà composto solo dai primi due modelli in quanto presenta performance maggiori nelle metriche dei singoli modelli.

B. Medical bot

Il bot ha fornito un'accuracy del 71% ed una loss con andamento decrescente. Per verificare il comportamento del bot abbiamo effettuato una serie di esperimenti provando a riprodurre possibili conversazioni con pazienti. Inizialmente

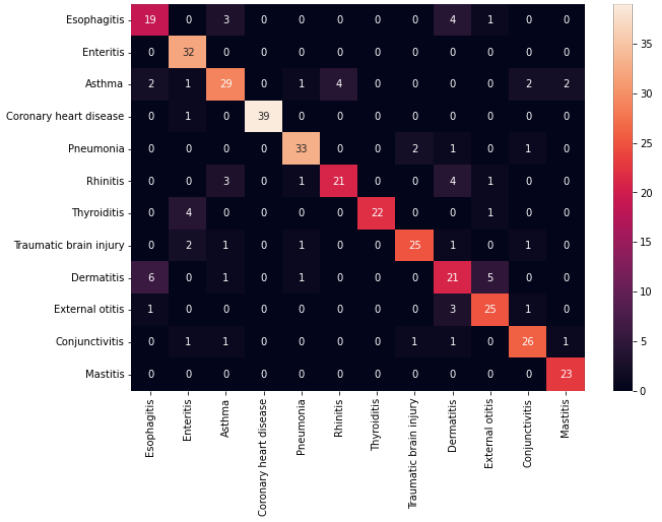

Fig. 14. Matrice di confusione random forest

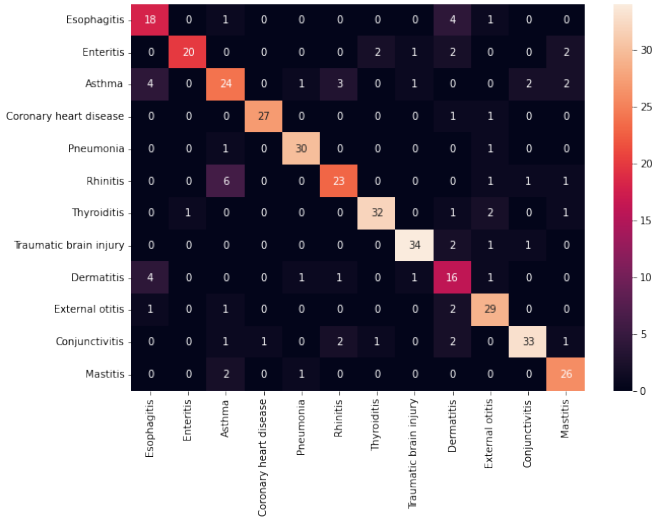

Fig. 15. Matrice di confusione SVC

abbiamo proposto al bot, per ogni patologia, alcuni dei sintomi più frequenti e discriminanti, in modo da verificarne il comportamento in quello che possiamo definire il "caso migliore". Durante la richiesta di sintomi aggiuntivi da parte del bot, abbiamo notato che per tutte le patologie venivano chiesti almeno altri 2 sintomi appartenenti alla patologia, cosa che nella maggior parte dei casi ha poi portato ad una corretta diagnosi finale. Successivamente abbiamo sottoposto al bot casi in cui veniva fornito un solo sintomo con alta frequenza nelle casistiche della patologia, e altri meno frequenti ma comuni a più malattie. In questi casi, il bot, nonostante le numerose richieste di sintomi aggiuntivi, fornisce predizioni meno accurate confondendo alcune delle patologie come l'asma con la polmonite, o la dermatite con l'esofagite. C'è da dire però, che in casi di malattie con sintomi discriminanti, ovvero sintomi presenti solo in campioni di quella patologia,

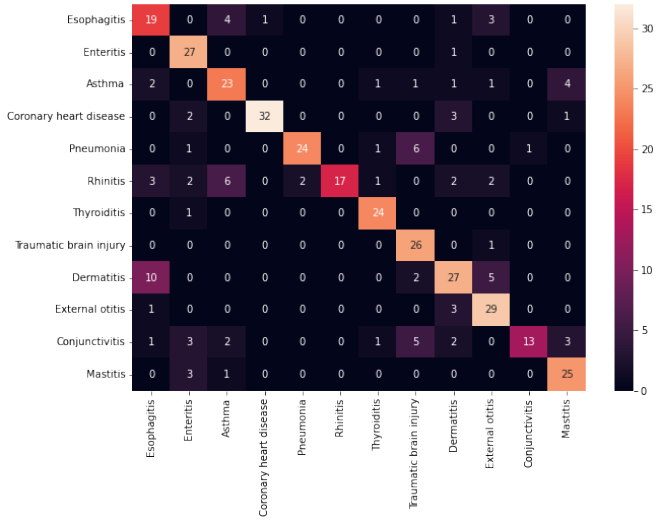

Fig. 16. Matrice di confusione KNN

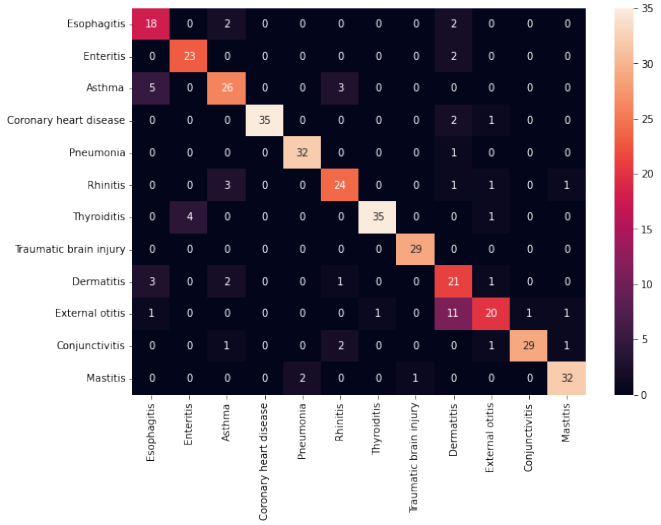

Fig. 17. Matrice di confusione soft voting classifier

il bot riesce con meno difficoltà a fornire diagnosi corrette; questo è il caso dell'otite o del trauma cranico.

Riassumendo, possiamo dire che i modelli presentano un comportamento migliore rispetto a quello del bot, ma ciò è dovuto al fatto che nel primo caso abbiamo un solo task di classificazione in cui il modello ha a disposizione i sintomi espliciti ed impliciti, mentre nel caso del bot, questo deve ottenere i sintomi impliciti attraverso predizioni in base a quelli ottenuti direttamente dall'utente. E' quindi difficile effettuare un confronto diretto tra le due categorie di modelli, ma sicuramente è possibile notare la loro dipendenza dai dati; osservando le curve di apprendimento possiamo supporre che aumentando il numero di campioni del dataset è possibile ottenere miglioramenti nelle performance. Inoltre, dalle discussioni precedenti, osservando la frequenza dei sintomi nei campioni, abbiamo notato una somiglianza tra i campioni di diverse malattie, cosa che potrebbe aver causato la confusione da parte dei modelli.

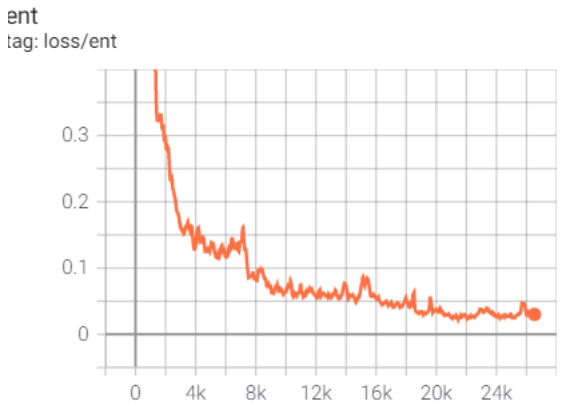

Fig. 18. Cross-entropy loss

VIII. Conclusioni

In questo paper è presentato uno strumento alternativo ai classici strumenti di supporto per la diagnostica predittiva, ovvero un bot in grado di dialogare con un paziente per l'estrazione e la verifica di sintomi con lo scopo di una diagnosi finale tra un insieme di 12 patologie differenti. Il bot ha fornito un'accuratezza media del 71% nella predizione considerando una maggiore inclinazione verso alcune patologie rispetto ad altre.

Inoltre, è stato presentato lo studio di 3 modelli supervisionati, Random Forest, Support Vector Classifier e KNN i quali hanno ottenuto valori di accuratezza rispettivamente pari a 82.5%, 81.4% e 74.6%. Successivamente è stato analizzato un voting classifier composto dai due migliori modelli (Random Forest e SVC) il quale ha fornito un'accuratezza ed una precisione maggiore rispetto ad ogni singolo modello.

Per quanto riguarda sviluppi futuri, potrebbe essere possibile effettuare una feature selection per trovare quei sintomi che risultano essere più discriminanti per ognuna delle malattie oppure pensare di utilizzare campioni che abbiano pochi sintomi in comune con altre malattie e molti relativi a quella da predire; c'è da dire però, che quest'operazione potrebbe andare in contrasto con un punto di vista medico in quanto un sintomo, seppur estremamente raro, potrebbe presentarsi in diverse malattie ed essere causa discriminante di essa.

References

[1] Zhang Q, Liu Y, Liu G, Zhao G, Qu Z, Yang W. An automatic diagnostic system based on deep learning, to diagnose hyperlipidemia. Diabetes Metab Syndr Obes. 2019;12:637-645. Published 2019 May 3. doi:10.2147/DMSO.S198547 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6510025/

[2] Kermany DS, Goldbaum M, Cai W, Valentim CCS, Liang H, Baxter SL, McKeown A, Yang G, Wu X, Yan F, Dong J, Prasadha MK, Pei J, Ting MYL, Zhu J, Li C, Hewett S, Dong J, Ziyar I, Shi A, Zhang R, Zheng L, Hou R, Shi W, Fu X, Duan Y, Huu VAN, Wen C, Zhang ED, Zhang CL, Li O, Wang X, Singer MA, Sun X, Xu J, Tafreshi A, Lewis MA, Xia H, Zhang K. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. Cell. 2018 Feb 22;172(5):1122-1131.e9. doi: 10.1016/j.cell.2018.02.010. PMID: 29474911. https://pubmed.ncbi.nlm.nih.gov/29474911/

[3] https://competitions.codalab.org/competitions/29706#participate