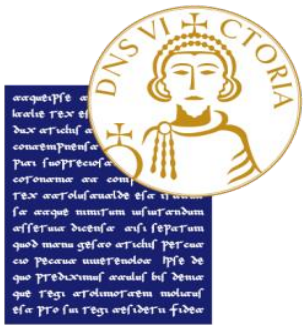


UNIVERSITÀ DEGLI STUDI DEL SANNIO
Dipartimento di Scienze e Tecnologie

Corso di Laurea in Biotecnologie

Tesi di Laurea in Bioinformatica

I.A. E FARMACOLOGIA: UN'ANALISI DI KNOWLEDGE PROBING SU DEEPSEEK

Relatore:
Prof. Francesco Napolitano

Candidato:
Manuel Maio

Matricola:
171004432

Anno Accademico 2024/25

INDICE

INTRODUZIONE

Da quando è stata sviluppata, l'intelligenza artificiale (IA) ha consentito di far progredire tanti settori, tra cui quello farmacologico, sia nello sviluppo di nuovi farmaci che nella comprensione di nuovi meccanismi chimici e biomolecolari. In particolare, l'evoluzione dei modelli linguistici di grandi dimensioni (Large Language Models, LLM) ha portato grandi benefici che hanno aperto a nuove prospettive per l'elaborazione e l'interpretazione della conoscenza scientifica, sia in ambito biomedico che in quello farmacologico. In particolare, le IA permettono di supportare i ricercatori nell'analisi dei dati farmacologici o nella generazione di ipotesi e per far ciò c'è bisogno che le conoscenze del modello siano corrette e di qualità. Purtroppo, le architetture delle IA sono chiuse e non permettono l'accesso diretto alle informazioni in esse codificate. Per questo esistono metodi per la verifica della conoscenza, tra cui il *Knowledge Probing*, ovvero un insieme di tecniche atte a valutare la coerenza, la profondità e la correttezza delle risposte date dai modelli di IA. Questi metodi sono basati su concetti simili a quelli di tecniche utilizzate in altri ambiti, tra cui quello scolastico, per testare la preparazione degli studenti. L'affidabilità delle risposte date da un LLM è importante in qualunque contesto, ma diventano cruciali in ambiti come la farmacologia. In questo lavoro di tesi è stata fatta, per l'appunto, un'analisi di Knowledge Probing sull'intelligenza artificiale cinese, DeepSeek, con l'obiettivo di valutare la sua conoscenza in ambito farmacologico. L'approccio di questo studio è stato di tipo sperimentale con lo sviluppo di due programmi, entrambi scritti in linguaggio Python, di cui quello principale ha permesso di interrogare tre modelli diversi di DeepSeek, con domande standard, in cui è stata modificata soltanto la coppia farmaco-malattia, al fine di studiarne le risposte, per condurre poi su di esse uno studio statistico per capirne l'affidabilità.

Questa tesi inizia spiegando in quale ambito farmacologico si posiziona questo studio, cioè in quello della Drug Discovery, ossia la scoperta di nuovi farmaci, parlando prima delle sue origini e poi delle tecniche che ne fanno parte. Prosegue poi con una panoramica sulle intelligenze artificiali, partendo dalle loro origini, passando per il loro sviluppo per poi arrivare a parlare di quelle più famose ed utilizzate oggi, tra cui figura proprio DeepSeek. Dopo aver parlato delle IA in generale, vi è un capitolo di approfondimento su una loro tipologia, ossia i Large Language Models (LLM), che sono basati sulle reti neurali e sono utilizzati per la comprensione e la generazione del linguaggio naturale. Si parlerà anche dei rischi e delle sfide etiche che portano con sé. Questi capitoli di approfondimento sono utili per comprendere meglio l'ambito di studio, le analisi effettuate e le loro motivazioni. Infine, è presente la bibliografia utilizzata per la scrittura di questa tesi.

1. DRUG DISCOVERY

1.1 COS'E' LA DRUG DISCOVERY

La drug discovery è il processo attraverso il quale vengono identificate e sviluppate nuove molecole con potenziale terapeutico. Si tratta di un ambito cruciale della farmacologia moderna, perché la disponibilità di nuovi farmaci è considerata un diritto fondamentale nei paesi sviluppati. Come riportato nella pagina enciclopedica *Progettazione di farmaci* della Treccani, il processo di sviluppo di un nuovo farmaco richiede in media circa dieci anni di lavoro, a partire dalla creazione di un nuovo principio attivo (New Chemical Entity, NCE), fino ad arrivare alla sua commercializzazione, e richiede investimenti dell'ordine di circa un miliardo di euro. La fase di scoperta vera e propria consiste nell'identificare innanzitutto un bersaglio biologico rilevante, come ad esempio un recettore, un enzima o un microrganismo patogeno, e poi trovare nuove molecole che lo modulino con efficacia terapeutica. Ciò implica un lavoro coordinato di diverse figure, tra cui biologi, chimici e farmacologi, che utilizzano sia approcci sperimentali in vitro e in vivo sia metodi computazionali per selezionare i composti più promettenti. Data la complessità dei sistemi biologici e la necessità di garantire efficacia e sicurezza, la scoperta di farmaci è storicamente un processo empirico e laborioso, sebbene oggi integrato con numerosi approcci innovativi, tra i quali spiccano quelli computazionali come l'intelligenza artificiale e il machine learning. In definitiva, la drug discovery si pone come missione fondamentale il poter fornire nuove armi terapeutiche contro le malattie, individuando delle molecole efficaci che possano modificare positivamente l'andamento delle patologie studiando il loro meccanismo d'azione.

1.2 STORIA DELLA DRUG DISCOVERY

Le prime forme di terapia farmacologica erano basate sull'utilizzo di estratti naturali di piante, erbe medicinali o composti prodotti da microrganismi. Per secoli, la scoperta di un farmaco ha seguito percorsi semplici ed empirici, ovvero basati sull'esperienza. Si osservavano gli effetti terapeutici delle piante o dei preparati naturali, come ad esempio le erbe officinali, e solo in seguito si cercava di isolarne il principio attivo. Moltissimi dei primi farmaci nacquero in modo fortuito (serendipità), o per esperienza clinica. Un esempio famoso è l'isolamento della morfina derivata dall'oppio alla fine del Settecento, così come la scoperta della penicillina a partire da un fungo da parte di Fleming nel 1928 (vedi ***Figura 1***). Tra il XIX e il XX secolo si sviluppò gradualmente un metodo più sistematico, basato sulla chimica organica. Tra gli anni '60 e '70, l'avanzamento della biochimica e della farmacologia rese possibile ottimizzare strutturalmente i farmaci già scoperti, vennero sviluppati dei farmaci analoghi più efficaci di antibiotici e di inibitori enzimatici già utilizzati. Negli anni '80 la disponibilità di tecniche di ingegneria genetica e di espressione di proteine ricombinanti segnò il passaggio alla ricerca basata sui *target molecolari*. Grazie a ciò fu possibile studiare a fondo nuovi bersagli biologici, recettori o gli enzimi specifici coinvolti nella malattia, e persino utilizzare proteine terapeutiche come l'insulina ricombinante, gli ormoni della crescita o gli interferoni. Allo stesso tempo lo sviluppo della chimica farmaceutica con l'avvento della *chimica combinatoria,* disciplina che permette la sintesi rapida e la simulazione al computer della molecola studiata per ottenere delle strutture analoghe, e la diffusione dei saggi biologici ad alto contenuto (High-Throughput Screening, HTS), permise di produrre e testare in breve tempo centinaia di migliaia di composti sintetici. Con il completamento del *Progetto Genoma Umano* nel 2001 si aprì una nuova frontiera.

Da allora sono stati identificati migliaia di geni associati a malattie, aumentando così il numero potenziale di bersagli farmaceutici. Negli ultimi decenni, la scoperta di farmaci si è spostata sempre più da un approccio basato sulla fortuna a strategie razionali e tecnologicamente avanzate, atte a colpire processi patologici specifici. Come riassunto da *Pina et al.,* la storia recente della drug discovery ha attraversato tre grandi fasi, partendo dall'era basata su regole empiriche del XIX secolo, passando per l'era degli antibiotici e delle tecniche moderne di screening degli anni '90, fino all'era odierna delle biotecnologie e delle scienze omiche. In questo contesto evolutivo, l'integrazione di nuove tecnologie, in particolare dell'intelligenza artificiale, rappresenta oggi l'ultima frontiera per velocizzare e perfezionare il processo di scoperta farmaceutica.

Figura 1. *Alexander Fleming (1881-1955), medico, biologo e farmacologo scozzese vincitore di un premio Nobel per la medicina nel 1945 per la scoperta della Penicillina, avvenuta nel 1928, e del Lisozima nel 1922. (Foto: National Geographic)*

1.3 TECNICHE MODERNE

Le tecniche moderne per la scoperta dei farmaci sono basate sostanzialmente su tre approcci generali: lo *screening fenotipico*, la *chimica combinatoria associata ad HTS* e l'approccio *target-based.*

- **Screening fenotipico (o fenomenologico):** questo metodo consiste nell'esporre cellule, tessuti o organismi modello a un elevato numero di composti chimici, sia naturali che sintetici, senza sapere a priori quale sia il loro bersaglio molecolare. In base agli effetti osservati sul fenotipo, ovvero il carattere osservabile, come la mortalità cellulare, la riduzione di uno marcatore o la guarigione di un sintomo, si identificano le molecole attive che in seguito venivano studiate per comprenderne meglio il meccanismo d'azione. Questo approccio era storicamente il primo utilizzato ed è tuttora utile quando il bersaglio è sconosciuto o complesso. Molti antibiotici e agenti antitumorali emersero da questi approcci applicati su colture cellulari o modelli animali. L'aspetto negativo di questo approccio è la mancanza di informazioni sul *target*, che rende più lento lo sviluppo di tali molecole.

- **Chimica combinatoria e HTS:** come visto precedentemente questo approccio permette lo sviluppo di molecole analoghe ad una di partenza, ed è stato sviluppato durante gli anni '80-'90. Questo metodo ha permesso di creare librerie chimiche molto estese di molecole analoghe, che hanno quindi la struttura simile, prodotte in modo automatico variando sistematicamente i gruppi chimici. Con l'ausilio dell'HTS automatizzato, tali librerie possono essere testate rapidamente su bersagli biologici in vitro. Questo approccio ha aumentato notevolmente l'efficienza del processo, dato che dalla sintesi di un gran numero di molecole emergono i cosiddetti *lead,* molecole *guida* utilizzate come base per lo sviluppo di nuovi farmaci, più promettenti da ottimizzare.

- **Approccio target-based:** Questo approccio è stato sviluppato grazie alla biologia molecolare che ha dato la possibilità di studiare un bersaglio biologico definito, come una proteina recettoriale o un enzima, isolandolo oppure esprimendolo come proteina ricombinante. Si poteva quindi progettare e testare delle molecole specifiche che permettessero di modulare quella proteina. Dopo aver selezionato un target *druggable,* ovvero un bersaglio modulabile da un farmaco, ad esempio coinvolto in una via metabolica patologica, si utilizzano degli screening mirati per individuare composti con affinità di legame. Questo approccio basato sul meccanismo d'azione ha dominato lo sviluppo farmaceutico negli ultimi decenni, grazie alla possibilità di prevedere razionalmente le modifiche chimiche dei composti. Tuttavia, un rischio del target-based è che molecole efficaci su un target singolo possano essere inefficaci in vivo per ridondanze biologiche o problemi di farmacocinetica.

1.4 TECNICHE SPERIMENTALI

Tra le tecniche sperimentali possiamo annoverare l'uso dell'*intelligenza artificiale* (IA) e del *machine learning* (ML), che negli anni recenti hanno aperto nuove strade e nuove opportunità per la progettazione di nuovi farmaci (vedi ***Figura 2***). L'articolo "*The Potential of Artificial Intelligence in Pharmaceutical Innovation"* permette di capire come l'IA (vedi ***Cap.2***) sia largamente utilizzata in diversi modi durante tutto il processo di *discovery* con lo scopo di ridurre i tempi di sviluppo e i costi, aumentando anche le probabilità di successo di scoprire nuove molecole.

Gli algoritmi di ML e le reti neurali permettono di approfondire la complessità delle interazioni molecolari, riuscendo a predire le proprietà biologiche dei nuovi composti e a generare delle strutture ottimizzate. Ad esempio, utilizzando tecniche di *QSAR* (Quantitative Structure - Activity Relationship), i modelli di ML possono associare strutture chimiche ad attività farmacologiche note per prevedere l'efficacia o la tossicità di un composto prima di che questo venga utilizzato. Un campo in rapida crescita è quello dei *modelli generativi,* reti neurali complesse addestrate su basi di dati di molecole note utilizzate poi per produrre molecole *de novo,* ovvero da 0, con le caratteristiche desiderate. Questi modelli generativi permettono di esplorare vaste opzioni chimiche finora inesplorate ed accelerano la fase di ottimizzazione dei lead: l'IA suggerisce variazioni strutturali che migliorano l'affinità col target o l'ADMET (assorbimento, distribuzione, metabolismo, escrezione, tossicità). Gli *algoritmi predittivi* e le *pipeline* automatizzate permettono di filtrare i candidati non promettenti, riducendo il numero di esperimenti richiesti, come ad esempio, l'uso di simulazioni computazionali e modelli ML che ha consentito a molte aziende di ridurre drasticamente i cicli di ottimizzazione chimica e i costi preclinici, in alcuni casi delle molecole sono passate da target nominato a fase clinica in meno di due anni. Un altro dei metodi utilizzato è quello basato sulla trascrizione indotta dai farmaci, cioè ciò che accade ad una cellula dopo che è stata esposta a un farmaco. Questa risposta viene chiamata profilo di espressione genica indotto, ed è come una firma che racconta quali geni si sono attivati e quali no. Analizzando queste firme, è possibile capire se una certa molecola è in grado di spingere una cellula verso uno stato benefico, come ad esempio farla smettere di comportarsi in modo tumorale, oppure stimolare un cambiamento positivo.

Un esempio dell'applicazione di questo metodo è riportato in Li et al., uno studio in cui è stato usato un modello di intelligenza artificiale per cercare farmaci capaci di far differenziare le cellule staminali tumorali del seno. Il modello, basato su reti neurali, ha imparato a riconoscere i profili genetici tipici di una cellula sana e differenziata, per poi cercare tra centinaia di composti quelli che potevano produrre un effetto simile. I risultati di questo studio sono stati promettenti, alcune molecole individuate dal modello si sono dimostrate capaci di ridurre le caratteristiche staminali e aggressive delle cellule tumorali. Questo metodo può essere applicato anche alla medicina personalizzata, cercando di prevedere come potrebbe rispondere un paziente a un farmaco basandosi proprio sui suoi dati genetici. In uno studio recente, un modello ha imparato a collegare i profili di espressione genica di cellule tumorali con la sensibilità a diversi farmaci, e ha persino fatto previsioni sulla sopravvivenza clinica dei pazienti, come riportato in Zhang et al. Infine, ci sono i *Large Language Models* (LLM), ossia i modelli linguistici di grandi dimensioni (vedi ***Cap.3***), di cui fa parte DeepSeek, LLM preso in esame in questo studio. Sono una categoria di modelli addestrati su immense quantità di dati che li rendono in grado di comprendere e generare linguaggio umano e altri tipi di contenuti, per eseguire un'ampia gamma di attività. Uno degli LLM biologici più famosi ed utilizzati è ESMFold, progettato e sviluppato da Meta AI, e serve per predire la struttura tridimensionale delle proteine a livello atomico. Lo studio "*Evolutionary-scale prediction of atomic-level protein structure with a language model*", pubblicato su *Science,* descrive come ESMFold superi i tradizionali approcci basati su allineamenti multipli (MSA), permettendo una predizione diretta, quindi senza tentativi, della struttura 3D a partire da una singola sequenza amminoacidica.

L'articolo evidenzia l'efficienza del modello, utilizzato per generare oltre 600 milioni di strutture, inoltre, dimostra come l'incremento della capacità del *language model* migliori direttamente la precisione strutturale. Un altro modello importante nel campo biologico è *AlphaFold2*, svilupapto da Google DeepMind e ben descritto nell'articolo di Yang et al., pubblicato nel 2023 e dal titolo "*AlphaFold2 and its applications in the fields of biology and medicine"*. AlphaFold2 rappresenta una delle più rilevanti innovazioni nel campo della biologia strutturale computazionale. Attraverso un'architettura basata su deep learning e meccanismi di attenzione, è in grado di predire la conformazione tridimensionale delle proteine a partire dalla sola sequenza amminoacidica, con un'accuratezza paragonabile a quella delle tecniche sperimentali, come la cristallografia a raggi X e la criomicroscopia elettronica. L'impatto di AlphaFold2 si estende ben oltre la semplice predizione strutturale: ha permesso la generazione di un database pubblico contenente oltre 200 milioni di strutture proteiche predette, con implicazioni significative per la ricerca biomedica, la progettazione di farmaci e la comprensione delle basi molecolari delle malattie. Sebbene ci siano ancora dei limiti, come nella modellazione di complessi proteici o di regioni intrinsecamente disordinate, questo modello rappresenta un punto di svolta nella biologia moderna. Tutti questi modelli condividono delle problematiche molto importanti basate soprattutto sulla conoscenza. Tra le più frequenti ci sono le allucinazioni, la dipendenza tra la particolare formulazione della domanda posta al modello e la relativa risposta, ovvero il framing, e i limiti temporali dati dal momento in cui l'IA è stata addestrata. Questo studio si è concentrato su uno dei metodi che vengono attuati per testare la conoscenza dei modelli, cioè il *Knowledge Probing (vedi **Cap.3**)*; questo metodo verrà spiegato nel dettaglio più avanti, assieme alle problematiche sopracitate.

Questi esempi mostrano come studiare l'effetto dei farmaci sull'attività dei geni, unito all'uso dell'intelligenza artificiale, stia diventando un modo sempre più utile e potente per scoprire nuovi trattamenti in modo più mirato, veloce e personalizzato. Di seguito è riportata una tabella (vedi ***Tabella 1***), scritta sulla base di alcuni articoli di riferimento citati nella bibliografia, utile per mostrare la comparazione tra le due tipologie di approcci. Nel complesso, l'integrazione dell' IA e del ML nella fase di scoperta di un nuovo farmaco permette di accelerare anche le fasi successive, contribuendo ad accorciare complessivamente i tempi per lo sviluppo e abbassare i costi.

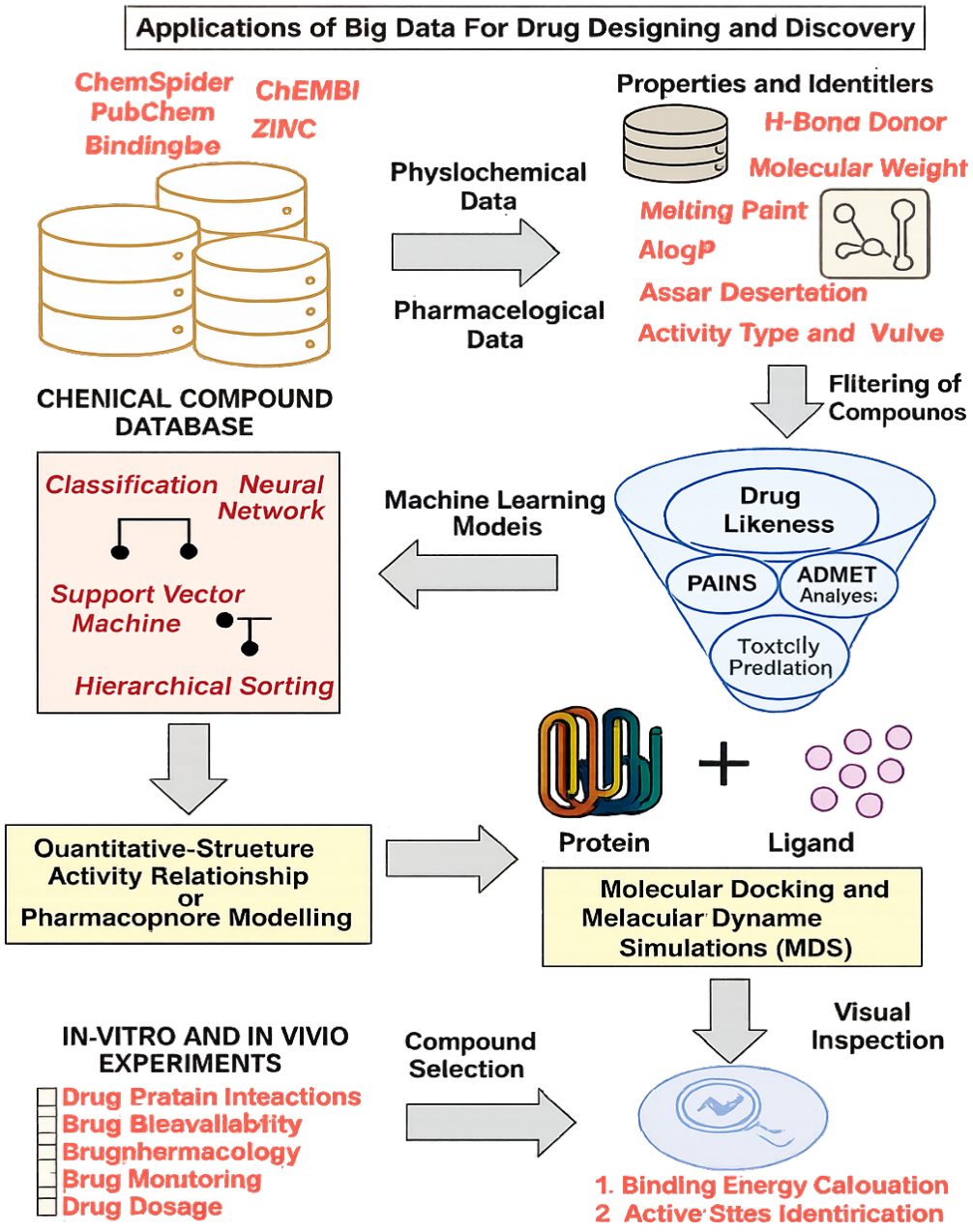


Figura 2. *Rappresentazione schematica del processo di Drug Discovery con I.A: i database forniscono grandi volumi di composti chimici e dati associati, i composti vengono filtrati sulla base di proprietà chiave (es. solubilità, tossicità), vengono poi impiegate tecniche di ML e reti neurali per prevedere attività biologica della molecola. Si utilizzano tecniche (es. QSAR) per analizzare relazioni tra struttura chimica e attività biologica, si verificano i legami tra ligando e proteina bersaglio (es. MDS, Molecular Drug Simulation) e, infine, il composto viene testato per verificare bioattività, biodisponibilità e tossicità. (Foto: Gupta et al., «Artificial intelligence to deep learning»)*

Modalità	Tradizionale	IA
Durata	10 – 15 anni	Pochi mesi – pochi anni
Costo	Più di 1 mld di dollari	100 – 500 mln di dollari
Approccio	Screening ad alto rendimento, chimica combinatoria	Machine learning, deep learning, reti neurali
Tasso di successo	~10% dei candidati entra in fase clinica	Maggiore accuratezza nella selezione dei candidati
Analisi dati	Limitata, basata su ipotesi ed esperimenti	Guidata da Big Data, pattern discovery automatizzato
Personalizzazione	Trattamenti generali	Farmaci personalizzati
Innovazione	Miglioramenti incrementali	Scoperta e sviluppo rapidi di nuovi farmaci
Regolamentazione	Ben definita ma lenta	In evoluzione, richiede validazione sperimentale dei modelli predittivi

Tabella 1. *Confronto tra Drug Discovery tradizionale e Drug Discovery implementato con IA.*

2. INTELLIGENZA ARTIFICIALE

2.1 ORIGINI E FASI STORICHE DELL'IA

L'Intelligenza Artificiale (IA) è nata ufficialmente nel 1956, quando alcuni ricercatori del tempo, tra cui Marvin Minsky, John McCarthy, Claude Shannon e Nathaniel Rochester, decisero di organizzare il *Dartmouth Summer Research Project on AI*. Questa conferenza, della durata di due mesi, è considerata l'evento fondante dell'intelligenza artificiale. Durante quella conferenza McCarthy coniò per la prima volta il termine *Artificial Intelligence* (intelligenza artificiale), dando così un nome a un campo interdisciplinare che punta a sviluppare macchine in grado di effettuare compiti cognitivi umani. Da quanto riportato nell'articolo *"A Comprehensive Review of Deep Learning: Architectures, Recent Advances", and Applications*, negli anni '50 e '60 viene sviluppata l'*IA Simbolica*, anche detta *Good Old-Fashioned AI*, ossia un metodo per codificare la conoscenza umana tramite regole logiche e sistemi esperti. Già nel 1955-56 Allen Newell, John McCarthy, Herb Simon e altri crearono il *Logic Theorist*, uno dei primi programmi capaci di ragionare in modo simbolico. Negli anni '70 l'IA simbolica vive un grande successo con la creazione di sistemi esperti basati su regole, software che racchiudono conoscenze specialistiche e aiutano gli utenti a risolvere problemi specifici. Tuttavia, proprio alla fine di quegli anni l'IA andò in contro ad uno stop del suo sviluppo a causa di limiti hardware, come memoria o velocità di calcolo, e l'incapacità di risolvere problemi elementari. Gli investimenti nella ricerca calarono drasticamente portando a due periodi di stagnazione dello sviluppo, il primo tra il 1974 e il 1980, mentre il secondo periodo lo si avrà tra il 1987 e il 1993. Nel frattempo, già dalla fine degli anni '50 nacquero approcci basati sulle reti neurali. Nel 1958 Rosenblatt inventò il *Perceptron*, un modello ispirato ai neuroni biologici.

Negli anni ’60 Minsky e Papert mostrarono i limiti del perceptron semplice, ma negli anni ’80, grazie allo sviluppo dell’algoritmo del *Backpropagation*, le reti neurali multilivello iniziarono a tornare di moda. Ricercatori come Geoffrey Hinton (vedi ***Figura 3***) e Yann LeCun studiavano già da tempo queste architetture complesse ed erano convinti che il *Deep Learning* sarebbe stato il futuro dell’IA, come riportato in un loro articolo del 1986 pubblicato su Nature dal titolo “*Learning Representations by Back-Propagating Errors”,* basato sulla backpropagation e diventato fondamentale per lo sviluppo delle IA.
In sintesi, l’IA ha vissuto diverse fasi, dall’IA simbolica degli anni ’50 e ’70, fino alle reti neurali classiche e ai sistemi esperti degli anni ’80. Dal 2010 in poi, con l’avvento del *Big Data* e l’aumento della potenza di calcolo moderna, è esploso il deep learning, metodo di addestramento basato su reti neurali complesse e tecniche come le *reti neurali convoluzionali* (CNN) e i *transformer* (Trasformatori). Queste tecniche, all’interno del più ampio contesto del machine learning (ML), consentono alle macchine di imparare da esempi e dati forniti, migliorando le proprie decisioni; quindi, sfruttano modelli matematici capaci di riconoscere pattern nei dati, imparare regole e fare previsioni con efficacia.

***Figura 3**. Geoffrey Everest Hinton (1947), psicologo e informatico britannico vincitore di un Nobel per la fisica nel 2024 per lo sviluppo dell'apprendimento automatico tramite reti neurali artificiali. È considerato, insieme a Yoshua Bengio e Yann LeCun, uno dei padri fondatori del Deep Learning. (Foto: Wikipedia)*

2.2 TIPOLOGIE DI IA

L'intelligenza artificiale si può classificare in due categorie principali, a seconda delle capacità cognitive e dell'autonomia del sistema. L'*IA debole* (o ristretta) si riferisce a sistemi progettati per svolgere compiti specifici senza alcuna comprensione generale. Questi applicano algoritmi mirati, come ad esempio un modello di visione artificiale per il riconoscimento delle immagini, e non possiedono capacità cognitive al di fuori dell'ambito per cui sono stati addestrati. In pratica, quasi tutte le applicazioni attuali di IA (dagli assistenti vocali alla diagnostica automatizzata) sono forme di IA debole o ristretta.

Al contrario, l'*IA forte* descrive sistemi con capacità cognitive equivalenti o superiori a quelle umane, tali sistemi dovrebbero essere in grado di apprendere e ragionare in modo flessibile e autonomo su qualsiasi problema, manifestando coscienza e comprensione generale. Correlata all'idea dell'IA forte c'è quella dell'*intelligenza artificiale generale* (AGI), ovvero un'entità capace di operare e adattarsi a un'ampia gamma di compiti, proprio come un essere umano. In sostanza la differenza tra IA debole e IA forte, soprattutto le AGI, è data dal fatto che queste ultime presentano un'autonomia cognitiva completa. Alcuni studi prevedono che l'AGI potrà alterare profondamente la società, benché resti ancora un obiettivo lontano e controverso. Infine, si ipotizza anche il concetto di *superintelligenza artificiale* (ASI), vale a dire un'IA che superi le capacità umane in ogni ambito. Questa tipologia di modello andrebbe oltre l'AGI, possedendo capacità di apprendimento e adattamento di molto superiori a quelle umane.

2.3 TECNICHE DI ADDESTRAMENTO

Ogni tipologia o modello di IA deve essere addestrata per poter adempiere al proprio compito, e per farlo si utilizzano diverse tecniche, alcune già citate in precedenza. La tecnica utilizzata da diversi anni è il deep learning, cioè una sottocategoria del machine learning che utilizza reti neurali profonde. Queste architetture consistono in numerosi strati di nodi *interconnessi*, detti neuroni artificiali, dove ogni strato apprende progressivamente rappresentazioni sempre più *astratte* dei dati. Nel dettaglio, i nodi, detti appunto neuroni artificiali, sono interconnessi poiché quelli dello strato precedente raccolgono informazioni, le elaborano, ciò che ne risulta viene passato ai nodi dello strato successivo e così per ogni strato.

Un solo nodo riceve informazioni da tutti gli altri nodi dello strato precedente per elaborarle a sua volta e passarle ai nodi successivi. Per rappresentazioni astratte dei dati si intende che i nodi non leggono l'informazione completa ma una versione più aggiornata, un esempio può essere un'immagine dove i nodi iniziali prendono tutte le informazioni, quelli dello strato intermedio definiscono le forme e i tratti del soggetto, o soggetti, mentre gli strati finali comprendono l'informazione; per riassumere l'informazione va raffinata passo dopo passo. L'introduzione delle reti profonde ha mostrato come un numero elevato di strati permetta di riconoscere pattern complessi nei dati, ponendo le basi delle moderne tecniche di apprendimento automatico. Il successo del deep learning è dovuto anche alla disponibilità di grandi volumi di dati e potenza computazionale. Dal punto di vista dell'apprendimento, le reti neurali e altri modelli possono essere addestrate secondo vari paradigmi fondamentali:

- **Apprendimento supervisionato:** il modello viene istruito su un dataset *etichettato*, ossia in cui a ciascun input viene associata una risposta corretta, ovvero l' output. L'algoritmo impara a mappare gli input in base agli output corretti minimizzando l'errore con la backpropagation. Si tratta della forma di apprendimento più diffusa, utilizzata ad esempio nella classificazione di immagini o testi. I dati etichettati richiedono però intervento umano per l'annotazione e sono disponibili in misura limitata.

- **Apprendimento non supervisionato:** il modello riceve dei dati *non etichettati* e deve individuare autonomamente delle strutture o dei pattern nascosti. Queste tecniche cercano similitudini e differenze nei dati (clusterizzazione, associazioni) senza un output predefinito. In pratica è come dare all'IA un insieme di foto e deve riconoscere cosa

rappresentano, senza sapere le risposte corrette. L'apprendimento non supervisionato può estrarre degli insight, cioè risposte non ovvie ottenute attraverso altri risultati, da grandi dataset grezzi ed è utile quando etichettare è impraticabile. La sua valutazione è più complessa poiché non esiste un output corretto con cui confrontarsi.

- **Apprendimento per rinforzo:** l'IA impara attraverso un sistema di *ricompense e penalità*; quindi, impara come comportarsi in un determinato contesto non perché gli diciamo in anticipo le risposte giuste, ma perché prova, sbaglia e impara dai propri errori. Ogni sua azione può portarle una ricompensa, *feedback positivo*, o una penalità *feedback negativo*, e in questo modo impara a prendere le decisioni che massimizzano il risultato finale. È un modello ispirato dai metodi di apprendimento delle persone, per esempio tramite il rinforzo, come quando veniamo premiati per aver preso una buona decisione. Grazie a questo meccanismo, il modello può affrontare sfide complesse senza che nessuno glielo spieghi in anticipo. Per aggiornare le sue strategie, l'IA si serve delle equazioni di Bellman, che lo aiutano a valutare quanto vale ogni mossa in base alle ricompense future. Inoltre, deve trovare l'equilibrio giusto tra esplorare nuove strade e sfruttare quelle che conosce già per ottenere il risultato migliore.

Oltre a queste tecniche classiche, si stanno esplorando delle varianti come l'apprendimento semi-supervisionato, che combina pochi dati etichettati con molti non etichettati, il *self-supervised learning*, dove il modello genera etichette automatiche dai dati stessi, e l'apprendimento per rinforzo profondo o *Deep Reinforcement Learning*. In particolare, il deep RL fonde reti neurali profonde con l'apprendimento per rinforzo.

3. LARGE LANGUAGE MODELS

3.1 STORIA ED EVOLUZIONE DEGLI LLM

I grandi modelli linguistici (LLM) sono una tipologia di IA, basate sulle reti neurali, che sono in grado di leggere, interpretare e scrivere in linguaggio naturale. Come accennato in precedenza le reti neurali nacquero tra gli anni 40' e 50', quando Warren McCulloch e Walter Pitss introdussero il modello matematico del neurone. Negli anni 50' si ebbe una svolta grazie a Claude Shannon, che applicò la teoria dell'informazione, disciplina legata alla matematica ed informatica che permette di comprimere, memorizzare, quantificare e trasmettere l'informazione, allo studio del testo naturale introducendo semplici modelli *n-gramma* per la previsione e compressione delle sequenze linguistiche. Questi approcci statistici erano alla base di molte applicazioni iniziali, come ad esempio il riconoscimento vocale o la traduzione automatica, e hanno permesso di gettare le prime fondamenta della modellizzazione linguistica. L'articolo "*A Comprehensive Review of Deep Learning: Architectures, Recent Advances, and Applications",* descrive lo sviluppo delle reti neurali a partire dalle reti neurali semplici (SNN), diventate poi, tra gli anni '90 e 2000, *reti ricorrenti* (RNN) e *Long Short-Term Memory* (LSTM). Entrambe erano in grado di superare le limitazioni degli n-grammi, imparando rappresentazioni dense dei token con il processo di *word embedding*, cioè il processo per cui non vengono più salvate le parole come entità separate ma bensì come vettori a cui sono associate più parole. (Vedi ***Figura 4***). Ciò ha permesso di poter comprendere anche la semantica delle frasi e ottenere una maggiore flessibilità dei modelli.

Con la disponibilità di grandi quantità di testi (web e libri digitali) è emersa la strategia di pre-addestramento generico, modelli neurali, come Word2Vec ed ELMo, venivano pre-addestrati su compiti generici non supervisionati e poi specializzati (*fine-tuned)*, per compiti specifici. Questo approccio si è affermato con due modelli rilasciati entrambi nel 2018, *Bidirectional Encoders Representations from Transformers* (BERT) e *Generative Pre-trained Transformer* (GPT), che hanno dimostrato l'efficacia del pre-addestramento, come riportato nell'articolo già citato. Entrambi i modelli si basano su una tecnologia, cioè il *Transformer*, proposta in un articolo di *Vaswani et al*. del 2017 (vedi ***Figura 5***) dal titolo "*Attention is all you need"*. Questa tecnologia è basata su un'architettura fondata esclusivamente su meccanismi di attenzione, che ha superato le RNN nel gestire sequenze lunghe. Dall'avvento dei Transformer, i modelli di linguaggio hanno avuto una crescita incredibile in quanto a capacità e potenza, culminando negli LLM odierni, come ChatGPT e DeepSeek, caratterizzati da miliardi o trilioni di parametri e abili in attività generali di task solving.

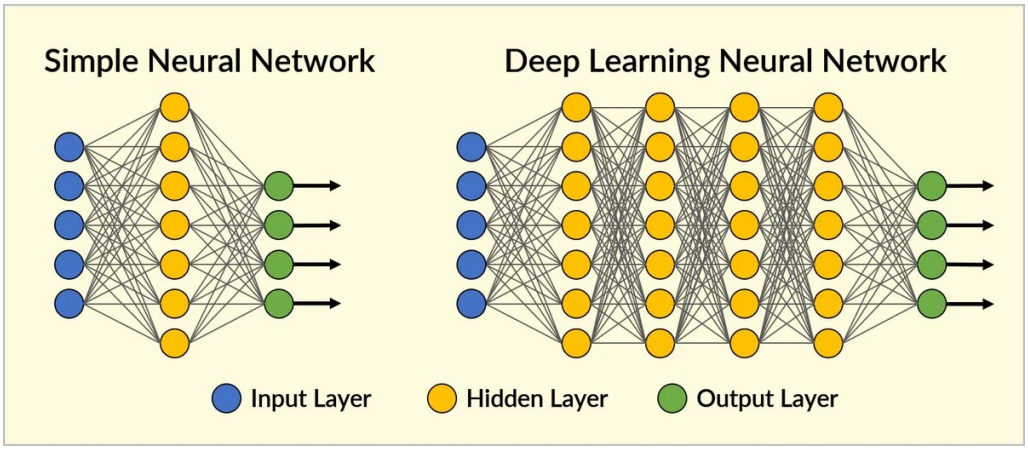


Figura 4. *Rappresentazione schematica della differenza tra le SNN e le reti neurali basate sul Depp Learning. Ciò che cambia in modo netto è la quantità di strati e, di conseguenza, la quantità di nodi (neuroni artificiali) che ne fanno parte.*

Figura 5. *Ashish Vaswani (1986) è un informatico indiano, ha lavorato come ricercatore presso il Google Brain and Information Sciences Institute, oggi è co-fondatore e CEO di Essential AI. Noto per il contributo nel campo del Deep Learning, in particolare per lo sviluppo della rete neurale Transformer, riportato nell'articolo di cui è co-autore, dal titolo "Attention is all you need". (Foto: AI Blog)*

3.2 ARCHIETTURA DEGLI LLM

La maggior parte degli LLM moderni è basata sull'architettura *Transformer* (vedi ***Figura 6***), introdotta da Vaswani et al. (2017), concepita per superare i limiti computazionali delle RNN e delle LSTM. Anziché processare il testo sequenzialmente, utilizza un meccanismo di *self-attention* per mettere in relazione ogni parola con tutte le altre presenti nella frase. In pratica, ogni token, cioè le parole, è in grado di vedere tutte le altre sia alla sua sinistra che alla sua destra ed è in grado di pesarle, ovvero assegnare un peso alle parole in base all'importanza che hanno nel contesto della frase. Ciò permette di cogliere dipendenze a lungo raggio, ovvero di mettere in relazione parole molto distanti nel discorso.

La struttura è divisa in due parti, *encoder* e *decoder*, ma esistono anche varianti con solo encoder o solo decoder. L'encoder permette di leggere l'input, come ad esempio una frase, e lo trasforma in una rappresentazione astratta, in pratica lega ogni parola e ne fa un riassunto che la macchina riesce a comprendere. Il decoder invece è la parte che produce il risultato, partendo dalla rappresentazione precedente. Questa struttura consente di leggere, pesare e processare tutte le parole dell'input in parallelo e ciò rende più rapido l'addestramento e migliora l'efficienza computazionale. Nei Transformer, lo strato di self-attention è *multi-head*, cioè replicato più volte, vi sono moduli di embedding per convertire i token in vettori continui e questo ha fornito la base per i modelli linguistici che comprendono e generano testo con elevata qualità, grazie alla sua capacità di modellare efficacemente il contesto globale di una frase. Il Self-attention in particolare permette al modello di focalizzarsi sui dettagli più rilevanti di un input, costruendo una rappresentazione dettagliata e completa.

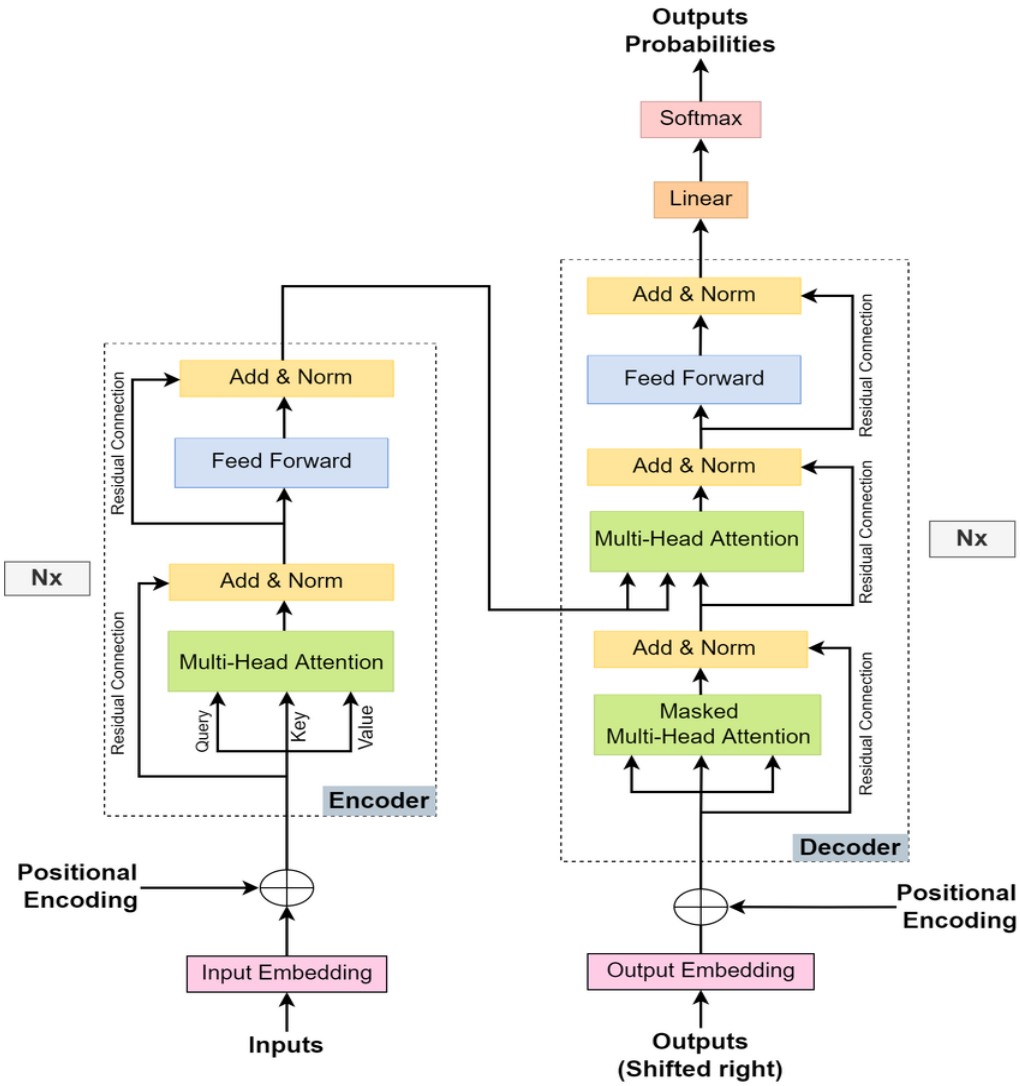


Figura 6. *Rappresentazione grafica della struttura del Trasformer e sia encoder che decoder sono formati da diversi moduli (Nx) che si susseguono. L'Encoder prende in input una frase, convertita in vettori tramite embedding, per poi aggiungerci un encoding posizionale utile per ricordare la posizione delle parole. Ogni suo modulo è formato da due sottosezioni, ovvero Multi-head Attention e Feed Forward. La prima permette ad ogni parola di vedere tutte le altre della frase e di pesarle in base al contesto, ciò avviene grazie a tre parametri, Query, Key e Value. Entrambe le sezioni sono seguite da Add & Norm, dove la prima parte serve per la somma residua, lega l'output del modulo precedente a quello appena ottenuto, mentre la seconda va a bilanciare i valori dei token. Il Decoder, invece, riceve in input i risultati dell'encoder spostati verso destra, così da non vedere l'ultima parola, cosa utile per non rendere il modello casuale. Applica anche lui embedding e positional encoding, ma i suoi moduli sono un più complessi, il Masked Multi-head Attention è simile a quello dell'encoder ma maschera le parole successive al token per non rendere il modello casuale. Il Multi-head Attention che troviamo dopo l'output dell'encoder permette di usare il contesto globale, mentre il Feed Forward è uguale. Ogni sezione è seguita da Add & Norm, che sono uguali all'encoder. Il layer Linear permette di ottenere un vettore lungo che rappresenta tutte le parole del vocabolario che potrà venire generata, mentre il layer Softmax permette di scegliere la parola da generare con probabilità più alta. (Foto: Islam et al. «A comprehensive survey on applications of transformers for deep learning tasks»)*

3.3 MODELLI E APPLICAZIONI

Negli ultimi anni sono stati sviluppati diversi LLM, oggi molto diffusi ed utilizzati in molti campi, tra cui GPT, BERT, LLaMA e DeepSeek (vedi ***Cap.3***), tutti spiegati nell'articolo già citato in precedenza, ovvero "*A Comprehensive Review of Deep Learning: Architectures, Recent Advances, and Applications"*. Nella nomenclatura dei modelli di LLM è presente una B (billion), che rappresenta il numero, in miliardi, di parametri che sfrutta per ragionare. GPT (Generative Pre-trained Transformer) è un LLM sviluppato da OpenAI ed è una famiglia di modelli decoder-only, orientati alla generazione autoregressiva di testo. Il primo modello GPT-1, rilasciato nel 2018, riuscì a dimostrare che addestrando un Transformer su testo non etichettato si potevano ottenere buoni risultati in compiti linguistici generici. GPT-2, rilasciato nel 2019 con 1.5B parametri, confermò che un modello sufficientemente grande, addestrato su un vasto *corpus web*, ovvero una raccolta di testi presi da internet, filtrati e organizzati, creato da OpenAI e denominato *WebText*, riesce a eseguire compiti specifici anche senza supervisione. Il passo successivo fu GPT-3, rilasciato nel 2020 con 175B di parametri, che è considerato il primo vero LLM. GPT-3 non solo era molto più grande dei precedenti, ma mostrava per la prima volta abilità emergenti (emergent abilities), come l'in-context learning, ovvero la capacità di apprendere nuovi compiti a partire da pochi esempi forniti nel prompt, senza necessità di ulteriori aggiustamenti dei pesi. Riuscì ad ottenere prestazioni importanti in traduzione, QA (Question & Answer) e ragionamento verbale grazie a questa flessibilità. Le versioni successive, come Codex, dedicato alla generazione di codice, o ChatGPT, ottimizzato per dialoghi, derivano dalla stessa famiglia GPT attraverso fine-tuning e addestramenti con feedback umano.

BERT (Bidirectional Encoder Representations from Transformers), è un LLM sviluppato da Google, e a differenza di GPT è un modello encoder-only bidirezionale. È stato pre-addestrato utilizzando compiti di *masked language modeling*, ovvero cercare di indovinare parole mascherate, e *next sentence prediction*, così da apprendere rappresentazioni contestuali di frasi. BERT ha contribuito in modo significativo al miglioramento in molti compiti di comprensione del linguaggio, grazie alla sua visione bidirezionale del contesto. Dopo il suo successo, sono apparsi molti derivati e ottimizzazioni come RoBERTa, ALBERT e DeBERTa, che ne migliorano l'efficienza. L'ultimo è LLaMA, sviluppato da Meta e rilasciato nel 2023, ha diversi modelli open-source, che variano da 7B a 65B, ottimizzati per la ricerca scientifica. Sono pre-addestrati su trilioni di token ottenuti da dati pubblici e hanno un'architettura simile a GPT-3 con poche modifiche. Grazie all'approccio open model, i modelli LLaMA hanno favorito la possibilità di sviluppo da parte di università e aziende. Le versioni più utilizzate sono LLaMA-13B, che risulta essere più performante di GPT-3 (175B) e la sua versione successiva, ovvero LLaMA-2, con tutte le varianti chat. Esistono anche delle versioni dette *distillate*, cioè versioni ottimizzate, più veloci e meno costose rispetto a quella di partenza. Come detto all'inizio, tutti questi LLM vengono utilizzati in un'ampia gamma di applicazioni in campi diversi. Le funzioni più comuni includono la generazione di testo e la comprensione semantica su larga scala. Ad esempio, sono impiegati per chatbot e assistenti virtuali nel servizio clienti, in ambito aziendale e marketing servono a generare descrizioni di prodotti, scrivere e-mail o riassumere feedback dei clienti e, grazie alla loro capacità di produrre testo coerente, sono utilizzati anche nella creazione automatica di articoli, post sui social media, sceneggiature e persino codici software.

Possono assistere nella traduzione automatica fra lingue diverse, nella correzione grammaticale e nel riassunto di testi complessi, nel settore sanitario, vengono utilizzati per analizzare cartelle cliniche e dati biomedici; infatti, un LLM può identificare pattern nei sintomi dei pazienti per supportare la diagnosi o stilare piani di cura personalizzati. Si potrebbero citare tanti altri utilizzi in molti altri campi, ma questi bastano a far capire la loro grande utilità e le potenzialità che possiedono e che ancora non sono state espresse.

3.4 LIMITI E SFIDE ETICHE

Gli LLM non hanno solo grandi potenzialità ma anche dei limiti importanti, tra cui figurano le *allucinazioni*, il *framing*, il *knowledge cutoff* e il *catastrophic forgetting.* Le allucinazioni sono un problema che si può riscontrare quando si generano delle risposte; infatti, queste possono essere non verificate o inventate, e ciò è dato dal fatto che il modello è probabilistico e non conosce la verità. Inoltre, non sono in grado di rilevare errori logici e quindi non garantiscono l'affidabilità delle informazioni che restituisce. Il framing è invece dovuto al come vengono poste e formulate le frasi date in input al modello. La formulazione della frase fatta in un determinato modo può essere interpretata in modo errato, ciò porta ad una possibile manipolazione dei risultati e può indurre dei bias involontari nel modello. Le ultime due problematiche sono strettamente legate, il knowledge cutoff è un problema che affligge la conoscenza del modello. In pratica è l'avere una conoscenza non aggiornata e quindi non poter fornire delle risposte corrette o complete. A questo si lega il catastrophic forgetting, cioè la perdita di conoscenza da parte del modello nel momento in cui si va a riaddestrarlo per aggiornare le informazioni, questo impedisce ai modelli di imparare in modo continuativo e stabile.

In questo studio è stata applicata una tecnica, ovvero il Knowledge Probing, per poter testare la conoscenza di un modello di IA. Questa tecnica permette di sondare la conoscenza del modello, scoprire dei bias o delle lacune, capire la consistenza, cioè in quali ambiti è forte e in quali è debole e di confrontare la conoscenza di più modelli. Per attuarla si sottopongono delle domande mirate all'LLM e bisogna capire se ha risposto correttamente oppure no. Dal punto di vista etico e sociale si sono presentate diverse sfide per nulla banali. Dall'articolo di *Raman et al*. dal titolo "*Navigating artificial general intelligence development: societal, technological, ethical, and brain-inspired pathways*", i modelli LLM vengono addestrati su grandi quantità di dati che sono tratti dal web e da fonti digitali, di conseguenza, essi incorporano anche i bias presenti nei dati di training, e quindi anche pregiudizi culturali o ideologici. Ciò può indurre i modelli a produrre delle risposte parziali o discriminatorie in certi contesti. Ci sono inoltre questioni legate alla privacy e alla proprietà intellettuale, poiché i dati utilizzati per l'addestramento potrebbero includere informazioni sensibili o protette da copyright, con possibili implicazioni legali e morali nell'uso dei modelli. Infine, gli LLM possono essere strumentalizzati per generare disinformazione su vasta scala, sollevando quindi preoccupazioni sulla veridicità dei risultati. Tutte queste problematiche portano a dover utilizzare questa tecnologia in modo responsabile, e per farlo è necessario implementare strategie di mitigazione, come possono essere i filtri, la supervisione umana, o le tecniche di feedback, per poi valutare in modo attento i loro output.

3.5 DEEPSEEK

Tra le IA più famose ed utilizzate in questo momento c'è DeepSeek, che non è solo il nome dell'IA ma anche dell'azienda che l'ha sviluppata, e che si sta affermando come una delle più interessanti nello scenario internazionale dell'intelligenza artificiale. Si tratta di una società cinese relativamente giovane, fondata nel maggio 2023 a Hangzhou da Liang Wenfeng, ex fondatore di un hedge fund tecnologico, che sta sfidando i giganti del settore con una strategia piuttosto semplice: realizzare modelli avanzati open source, contenendo allo stesso tempo i costi di sviluppo. È in questo contesto che DeepSeek ha lanciato, a gennaio 2025, *DeepSeek-R1*, un modello di ragionamento che, secondo quanto affermato dalla società e può competere con l'O1 di OpenAI. Come riportato nell'articolo "*Incentivizing Reasoning Capabiliti in LLMs via Reinforcement Leraning"*, è uno dei modelli più avanzati nel portare a termine compiti complessi come la soluzione di problemi matematici, la programmazione e l'elaborazione di testo. Ciò che colpisce è che questo modello è stato addestrato con un budget di circa sei milioni di dollari, una cifra relativamente bassa se paragonata alle centinaia di milioni che società, come OpenAI, investono per costruire e addestrare modelli di questo livello, da quanto riportato nell'articolo "*DeepSeek explained: Everything you need to know"*. È quindi importante sottolineare come l'assistente di DeepSeek, reso disponibile tramite un'applicazione mobile, abbia rapidamente scalato le classifiche delle più scaricate, superando per la prima volta l'app di ChatGPT. La sua popolarità non è passata inosservata nemmeno ai mercati, dove le azioni di colossi come NVIDIA e Microsoft, per la prima volta da mesi, hanno subito una flessione in seguito all'annuncio. Un evento che sembra dimostrare come DeepSeek non sia più una realtà di nicchia, ma un potenziale competitor a livello globale.

Dopo l'R1 è seguito *DeepSeek-V3*, un modello di linguaggio di tipo *Mixture of Experts,* modello formato da sottoreti neurali ognuna esperta in un campo, con 671 miliardi di parametri, di cui però solamente 37 miliardi vengono attivati ogni qualvolta si elabora testo, come riportato nell'articolo *DeepSeek-V3 Tecnichal Report*. Grazie a innovazioni come la Multi-head Latent Attention e a una gestione avanzata delle risorse, questo modello può ottenere prestazioni simili a quelle delle *IA closed-source*, cioè modelli di cui non è permessa la modifica o la distribuzione del codice da parte di terzi, pur consumando una quantità di potenza di calcolo relativamente contenuta. È proprio l'approccio *open model* a costituire uno degli aspetti più innovativi di DeepSeek. A differenza di società come OpenAI, che tendono a mettere a pagamento l'accesso al proprio modello, DeepSeek mette a disposizione di ricercatori, università, società di ogni dimensione e comunità di sviluppatori l'accesso gratuito al suo modello, con alcune funzionalità particolari comunque a pagamento. In questo modo organizzazioni che non avrebbero le risorse per acquistare l'accesso a modelli closed-source, possono partire da una base avanzata per costruire le loro applicazioni. È per questo che l'open model di DeepSeek può essere importante tanto per le aziende, quanto per le università o le startup, che in passato non avrebbero potuto utilizzare questo genere di intelligenza a causa dei costi elevati o delle restrizioni sulle licenze. Inoltre, l'azienda cinese ha ampliato la sua offerta con modelli specializzati, come DeepSeek Coder, dedicato alla programmazione, o Janus Pro, multimediale e in grado di gestire testo e immagini. Le applicazioni di DeepSeek spaziano dalla creazione di contenuti, all'analisi di testo, dalla soluzione di problemi matematici, all'assistenza alla programmazione, senza dimenticare la conversazione multilingue. La sua natura aperta, a basso costo e facilmente adattabile, lo colloca come una soluzione adatta tanto a contesti di ricerca quanto ad applicazioni industriali, dove la flessibilità è importante.

4. ANALISI DI KNOWLEDGE PROBING SU DEEPSEEK

4.1 OBIETTIVO E DATABASE

Questo lavoro di tesi si è posto l'obiettivo di comprendere quanto fossero affidabili le risposte di DeepSeek in ambito farmacologico attraverso un'analisi basata su *Knowledge Probing* (vedi ***Cap.3***). Quest'analisi è stata effettuata ponendo all'IA una domanda standard, modificando ogni volta la coppia farmaco-malattia. I dati relativi ai farmaci e alle malattie ad essi associati sono stati estratti da un database, il *Therapeutic Target Database* (TTD), riportato nello studio *TTD:Therapeutic Target Database describing target druggability information*. Questo database è stato sviluppato da due istituzioni accademiche, cioè il Laboratorio di Ricerca Innovativa sui Farmaci e Bioinformatica (IDRB) presso il *College of Pharmaceutical Sciences*, della Zhejiang University in Cina, e il Bioinformatics and Drug Design Group (BIDD) del *Department of Pharmacy*, della National University of Singapore. Il TTD è continuamente aggiornato dal team coordinato dal Dr. Ying Zhou e dal Dr. Yintao Zhang, è accessibile pubblicamente ed è stato progettato per fornire informazioni dettagliate e aggiornate sui target terapeutici, i farmaci associati, le malattie corrispondenti, integrando anche dati sullo stato clinico e sulla *druggability*, ovvero la capacità di questi bersagli molecolari di essere modulati tramite l'uso di farmaci. Permette anche di esaminare dati clinici e preclinici, sia di farmaci in sviluppo sia di quelli già in commercio, e di poter identificare possibili target molecolari per determinate patologie. La versione del database utilizzata è la più recente, ovvero la 10.1.01 del 30 marzo 2024, al cui interno sono presenti più di 39 mila farmaci e più di 3700 target biologici. Con target biologici si intendono recettori, proteine bersaglio, enzimi, canali ionici o geni.

Come detto con questo database è possibile ottenere diverse informazioni, tra cui lo stato di ricerca in cui si trova un farmaco per una determinata malattia. I farmaci presenti nel database sono stati annotati per stato di sviluppo, ad esempio *approved*, *phase 1*, ecc (vedi ***Grafico 1***). I diversi stati riportati come Altri sono dettagliati in ***Tabella 2***.

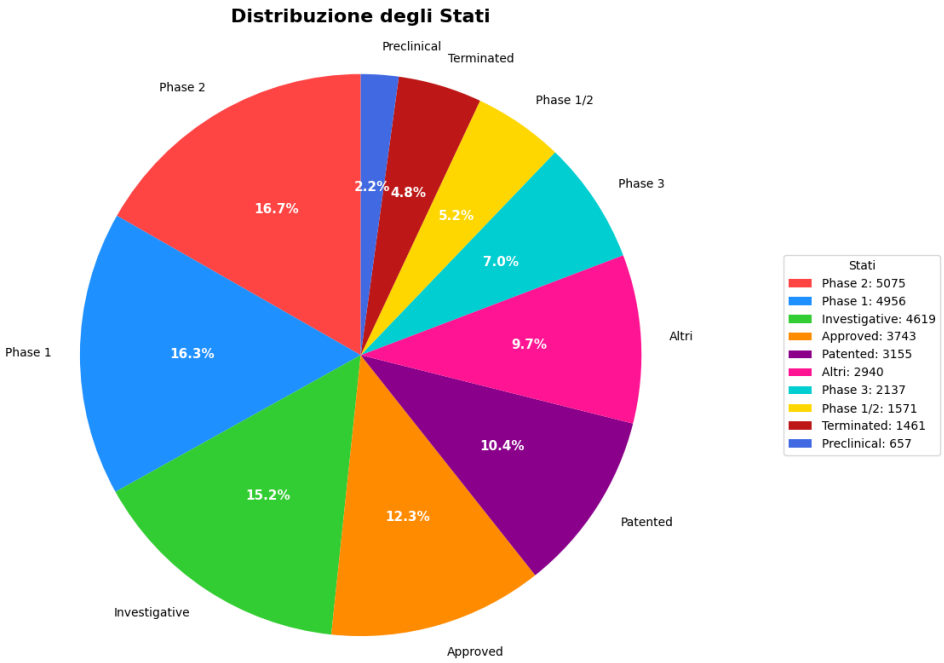


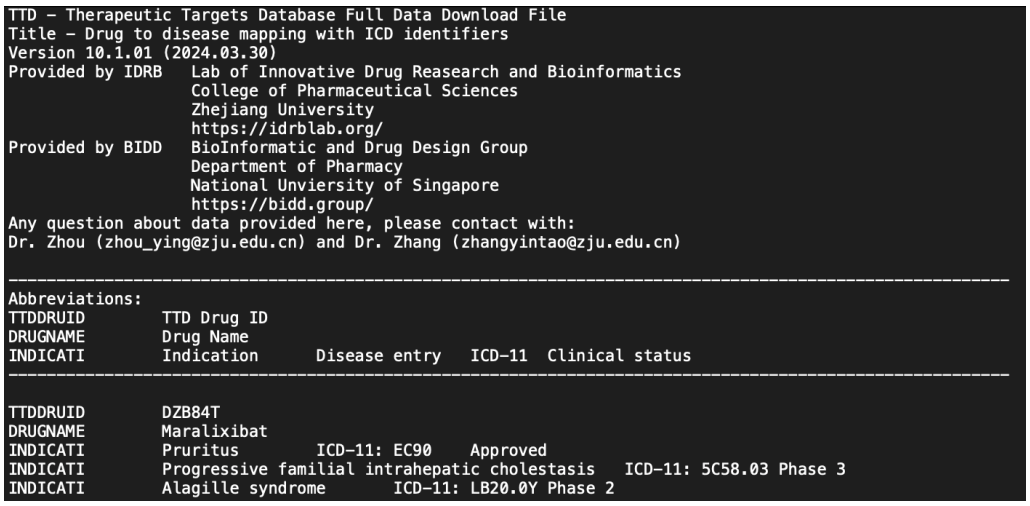
Grafico 1. *Grafico a torta che rappresenta la quantità, in percentuale, di farmaci presenti in ogni stato. Nel database sono presenti 36 stati diversi, di cui però la maggior parte sono rappresentati dalla sezione "Altri".*

Stato	Quantità
Phase 2/3	269
Discontinued in phase 3	268
Clinical trial	115
Withdrawn from market	92
Application submitted	72
Phase 4	61
Discontinued in phase 1/2	57
Discontinued in preregistration	41
Preregistration	22
Phase 2a	15
Registered	14
Phase 0	11
Phase 2b	10
Approved in EU	8
Phase 1b	8
Ind submitted	7
Nda filed	6
Discontinued in phase 2/3	6
Approved in China	5
Approved (orphan drug):	5
Phase 1b/2a	3
Phase 1/2a	2
Discontinued in phase 4	2
Discontinued in phase 2b	2
Approval submitted	1
Bla submitted	1
Discontinued in phase 2a	1

Tabella 2. *Rappresentazione tabellare decrescente delle quantità di farmaci presenti nei 27 stati raggruppati in "Altri".*

4.2 MATERIALI E METODI

L'analisi è stata eseguita su DeepSeek (vedi ***Capitolo 3***), ed in particolare sono state utilizzate diverse distillazioni del modello *DeepSeek-r1*, l'1.5B, il 7B e il 32B. Sono stati sviluppati due *script*, entrambi scritti in linguaggio Python, nella versione 3. Il database (vedi ***Figura 7***), scaricato localmente, è stato denominato *P1-05 Drug_disease.txt* e ha un'intestazione in cui sono riportate informazioni riguardanti lo studio da cui deriva e da chi è stato redatto. I dati al suo interno sono organizzati per sezioni divise da spazi. Ogni sezione è formata dall'identificativo del farmaco (ID), riportato nel campo TTDDRUID, dal nome del farmaco, nel campo DRUGNAME, dal nome della malattia nel campo INDICATI, abbreviazione di *Indication*, e per ultimo dallo stato clinico, nel campo ICD-11.



```
TTD - Therapeutic Targets Database Full Data Download File
Title - Drug to disease mapping with ICD identifiers
Version 10.1.01 (2024.03.30)
Provided by IDRB   Lab of Innovative Drug Reasearch and Bioinformatics
                   College of Pharmaceutical Sciences
                   Zhejiang University
                   https://idrblab.org/
Provided by BIDD   BioInformatic and Drug Design Group
                   Department of Pharmacy
                   National Unviersity of Singapore
                   https://bidd.group/
Any question about data provided here, please contact with:
Dr. Zhou (zhou_ying@zju.edu.cn) and Dr. Zhang (zhangyintao@zju.edu.cn)

--------------------------------------------------------------------------------------------------------
Abbreviations:
TTDDRUID        TTD Drug ID
DRUGNAME        Drug Name
INDICATI        Indication      Disease entry   ICD-11  Clinical status
--------------------------------------------------------------------------------------------------------

TTDDRUID        DZB84T
DRUGNAME        Maralixibat
INDICATI        Pruritus        ICD-11: EC90    Approved
INDICATI        Progressive familial intrahepatic cholestasis   ICD-11: 5C58.03 Phase 3
INDICATI        Alagille syndrome       ICD-11: LB20.0Y Phase 2
```

Figura 7. *File del database utilizzato per l'analisi, in alto c'è l'intestazione con le informazioni relative al database stesso, poi è presente una legenda delle etichette che troviamo al suo interno e, infine, un esempio di farmaco e di come si presenta nel file.*

Il primo programma, denominato *Farmaco_malattia.py*, permette di aprire il file del database, scaricato localmente e denominato *P1-05 Drug_disease.txt*, creare una matrice con tutte le coppie farmaco-malattia possibili, generare una *Query* (vedi ***Figura 8***), cioè la domanda da inviare a DeepSeek, riceverne la risposta e, una volta completate tutte le query, effettuare una statistica sui risultati ottenuti. Il programma genera una matrice formata da tante righe quante sono le coppie farmaco-malattia e da tre colonne. La prima colonna contiene il nome del farmaco, la seconda il nome delle malattie e la terza lo stato clinico del farmaco (vedi ***Grafico 1***). La query è costruita in modo dinamico, presenta una struttura ricorrente, in cui però vengono sostituiti i nomi del farmaco e della malattia, al fine di verificare se le risposte di DeepSeek siano corrette o meno. Infatti, la prima metà delle domande sottoposte all'IA erano vere, cioè doveva rispondere 'Yes', mentre la seconda metà erano false, e doveva quindi rispondere 'No'. In questo modo è stato possibile ottenere sia le risposte corrette sia quelle date da DeepSeek. Le domande sono state fatte in inglese e scritte in modo stringente, affinché le risposte date fossero esclusivamente 'Yes' e 'No', inoltre, per migliorare la chiarezza e la leggibilità dei risultati, il ragionamento svolto dall'IA per giungere alla risposta è stato rimosso dalla stampa, ma è stato comunque salvato in un file CSV esterno.

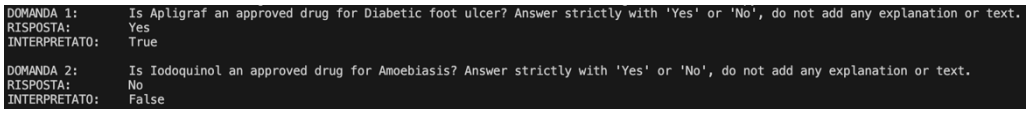


Figura 8. *Esempio di query poste a DeepSeek, con relative risposte e interpretazione da parte del programma.*

L'analisi è stata fatta ponendo a tutti e tre i modelli distillati citati prima 500 domande vere, quindi solo con stato *approved* e nessun'altra eccezione a questa dicitura, e 500 domande false.

Le coppie sono formate dagli stessi farmaci usati prima, ma con malattie che non hanno nessuno stato in relazione a quel farmaco, per un totale di 1000 domande. I tempi di esecuzione dello script dipendono dal modello a cui si fa riferimento. Per quanto riguarda l'1.5B sono bastate 6/7 ore per completarlo, per il 7B è servito poco tempo in più, circa 10 ore, mentre per il 32B i tempi sono stati più lunghi, quasi 24 ore. Il primo modello è stato scaricato e fatto girare localmente su un Mac con processore Apple M1 e 8GB di memoria RAM, mentre gli altri due, sono stati scaricati sul *cluster* in dotazione al Dipartimento di Scienze e Tecnologie dell'Università del Sannio. È formato da 8 CPU Intel Xeon Platinum 8280 @2.70GHz, ognuna delle quali ha 28 core fisici, per un totale di 224 core fisici, 1.5 TB di memoria RAM, 150 TB di memoria storage e una GPU NVIDIA V100. Il cluster è un insieme di computer, detti nodi, legati tramite la rete e che lavorano come fosse una macchina sola.

```
from ollama import chat #Libreria per chattare con deepseek
import random #Libreria per ottenere la randomicità
import re #Librerie per le espressioni regolare
from tabulate import tabulate #Libreria per ottenere tabellazione
from scipy.stats import fisher_exact #Libreria per test di Fisher
import csv #Libreria per creare CSV

file = "P1-05-Drug_disease.txt" #File farmaci e malattie
version = "deepseek-r1:1.5b" #Versione di deepseek utilizzata
NUM_COPPIE = 500 #Numero di coppie farmaco-malattia da testare

#Funzione apertura file e creazione matrice farmaco-malattie-stato
def load(file):
    with open(file) as f:
        content_file = f.readlines()

    #Matrice con farmaco.malattia-status
    matrice_completa = []
    current_drug = None
    for line in content_file:
        if "DRUGNAME" in line:
            current_drug = line.split("\t")[1].strip()
        elif "INDICATI" in line and current_drug is not None:
            parts = line.split("\t")
            malattia = parts[1].strip()
            stato = parts[3].strip()
            matrice_completa.append((current_drug, malattia, stato))
    return matrice_completa

def create_datasets(matrice_completa):
    #Raccoglie tutte le coppie approvate
    tutte_approvate = []
    #Dizionario malattie in qualsiasi stato per ogni farmaco
    farmaci_con_malattie = {}
    #Set di tutte le malattie disponibili
    tutte_malattie = set()
```

```
    for farmaco, malattia, stato in matrice_completa:
        tutte_malattie.add(malattia)
        #prende come stato solo "approved", eliminando gli altri casi
        if "approved" == stato.lower():
            tutte_approvate.append((farmaco, malattia, stato))
        #Controlla se il farmaco è già nel dizionario
        if farmaco not in farmaci_con_malattie:
            #Se il farmaco non è nel dizionario, inizializza un set vuoto
            farmaci_con_malattie[farmaco] = set()
        #Aggiungi la malattia al set del farmaco (in qualsiasi stato)
        farmaci_con_malattie[farmaco].add(malattia)
    #Seleziona coppie approvate casuali
    approvate = random.sample(tutte_approvate, min(NUM_COPPIE,
len(tutte_approvate)))
    matrice_unica = []

    #Aggiungi coppie approvate con True
    for farmaco, malattia, stato in approvate:
        matrice_unica.append((farmaco, malattia, stato, True))

    #Genera coppie non approvate usando gli stessi farmaci
    tutte_malattie = list(tutte_malattie)

    for farmaco, _, stato in approvate:
        #Trova malattie che NON sono in qualsiasi stato con il farmaco
        malattie_non_approvate = [m for m in tutte_malattie
                                  if m not in farmaci_con_malattie[farmaco]]
        if malattie_non_approvate:
            #Scegli una malattia casuale tra quelle non del farmaco
            malattia_scelta = random.choice(malattie_non_approvate)
            matrice_unica.append((farmaco, malattia_scelta, "N/A", False))

    return matrice_unica

#Funzione domanda/risposta deepseek
def ask_ds(drug, disease, i):
    question = f"Is {drug} an approved drug for {disease}?
    Answer strictly with 'Yes' or 'No',
    do not add any explanation or text."
    risposta_completa = chat(model = version, messages =
[{"role":"user","content":question}]).message.content
    print("DOMANDA",i+":\t", question)

    #Estrai il ragionamento dai tag <think>
    ragionamento_match = re.search(r'<think>(.*?)</think>', risposta_completa,
flags=re.DOTALL)
    ragionamento = ragionamento_match.group(1).strip()
                   if ragionamento_match else ""

    #Rimuove il ragionamento
    risposta_pulita = re.sub(r'<think>.*?</think>', '', risposta_completa,
flags=re.DOTALL).strip()
    #Rimuove tutta la punteggiatura
    risposta_per_parsing = re.sub(r'[^\w\s]', '', risposta_pulita.lower())

    print("RISPOSTA:\t", risposta_pulita)

    #Interpreta la risposta
    interpretato = False
    for parola in risposta_per_parsing.split():
        #ControllA se la risposta è yes o no
        if parola in ["yes", "no"]:
            interpretato = parola == "yes"
            break

    print("INTERPRETATO:\t", interpretato, end="\n\n")
    return interpretato, ragionamento
```

```
matrice_completa = load(file)
matrice_dataset = create_datasets(matrice_completa)

#Ciclo per ottenere le risposte da deepseek per tutte le coppie
i = 1
for j, (farmaco, malattia, stato, approvato) in enumerate(matrice_dataset):
    risposta, ragionamento = ask_ds(farmaco, malattia, str(i))
    matrice_dataset[j] = (farmaco, malattia, stato,
                          approvato, risposta, ragionamento)
    i = i + 1

#Conversione dei booleani in interi (1 for True, 0 for False)
#Risposte interpretate come interi
approvato_deepseek_int = [int(x) for x in [risposta[4] for risposta in
matrice_dataset]]
#Stato reale (True/False) convertito in intero
approvato_reale_int = [int(row[3]) for row in matrice_dataset]

#Calcolo campioni totali e predizioni corrette
total_samples = len(matrice_dataset)
correct_predictions = sum(1 for r, s in zip(approvato_deepseek_int,
approvato_reale_int) if r == s)
#Calcola matrice di confusione
correct_positives = sum(1 for r, s in zip(approvato_deepseek_int,
approvato_reale_int) if r and s) #Veri positivi
incorrect_positives = sum(1 for r, s in zip(approvato_deepseek_int,
approvato_reale_int) if not r and s) #Falsi negativi
correct_negatives = sum(1 for r, s in zip(approvato_deepseek_int,
approvato_reale_int) if not r and not s) #Veri negativi
incorrect_negatives = sum(1 for r, s in zip(approvato_deepseek_int,
approvato_reale_int) if r and not s) #Falsi positivi

accuracy = correct_predictions / total_samples

#Precision, recall, F1 score
precision = correct_positives / (correct_positives + incorrect_negatives) if
(correct_positives + incorrect_negatives) > 0 else 0
recall = correct_positives / (correct_positives + incorrect_positives) if
(correct_positives + incorrect_positives) > 0 else 0
f1_score = 2 * (precision * recall) / (precision + recall) if (precision +
recall) > 0 else 0

contingency_table = [[correct_positives, incorrect_positives],
                     [incorrect_negatives, correct_negatives]]

#Calcolo del test esatto di Fisher
_, fisher_p_value = fisher_exact(contingency_table, alternative='two-sided')

print("\nStatistiche:")
print(f"Numero totale di campioni: {total_samples}")
print(f"Risposte corrette: {correct_predictions} ({accuracy:.2%})")
print(f"Precisione: {precision:.2%}")
print(f"Recall (Sensibilità): {recall:.2%}")
print(f"F1 Score: {f1_score:.2f}")

#Stampa matrice di confusione
print("\nMatrice di Confusione:")
print(tabulate([
    ["", "Predetto Positivo", "Predetto Negativo"],
    ["Realmente Positivo", correct_positives, incorrect_positives],
    ["Realmente Negativo", incorrect_negatives, correct_negatives]
], headers="firstrow"))
```

```
#Stampa p-value di Fisher
print(f"\nP-value del test di Fisher: {fisher_p_value}")
if fisher_p_value < 0.05:
    print("Il risultato è statisticamente significativo (p < 0.05)")
else:
    print("Il risultato non è statisticamente significativo (p >= 0.05)")

#Scrittura di un file CSV in cui salvare le domande e le risposte
output_file = "risultati_dataset.csv"
with open(output_file, mode="w", newline="", encoding="utf-8") as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(["farmaco", "malattia",
                     "stato", "risposta_vera",
                     "risposta_deepseek", "ragionamento"])
    for farmaco, malattia, stato, risposta_vera, risposta_deepseek,
ragionamento in matrice_dataset:
        writer.writerow([farmaco, malattia, stato, risposta_vera,
risposta_deepseek, ragionamento])
print(f"Risultati salvati in: {output_file}")
```

Algoritmo 1. *Farmaco_malattia.py, script principale per inviare domande a DeepSeek, ricevere le risposte e analizzare statisticamente i risultati.*

Lo script, dopo aver completato tutte le 1000 query e aver ottenuto tutte le risposte, esegue un'analisi statistica tramite un test di Fisher. È stato scelto questo test perché permette di capire se esiste una relazione statistica tra le risposte di DeepSeek e i dati reali oppure se DeepSeek risponda in modo casuale. Vengono calcolati anche altri parametri, ovvero la precisione, la sensibilità, l'F1 score e la percentuale di risposte corrette. Dopo aver eseguito l'analisi, lo script stampa i risultati con la matrice e le statistiche, tra cui anche la significatività del test eseguito. Quest'ultimo è considerato significativo se il p-value ottenuto è inferiore a 0.05. Infine, viene creato un file CSV denominato *risultati(modello).CSV*, in cui sono salvati diversi parametri, ossia il nome del farmaco, il nome della malattia, lo stato clinico (incluso lo stato N/A, ovvero *Not Approved,* che è uno stato assegnato alle coppie negative generete dall'algoritmo) la risposta corretta, la risposta di DeepSeek e il ragionamento dell'IA. Di seguito sono riportati gli esempi di tutte e quattro le categorie di domande (vedi ***Figure 9,10,11,12***), come le relative risposte di DeepSeek, inclusa la parte di ragionamento (vedi ***Ragionamenti 1,2,3,4***).

farmaco	malattia	stato	risposta_reale	risposta_deepseek
Vitamin B1	Vitamin B1 deficiency	Approved	True	True

Figura 9. *Risposta realmente vera, ossia dove la risposta di DeepSeek combacia con la realtà dei dati clinici ed entrambi concordano con 'Yes'.*

Okay, so I need to figure out if Vitamin B1 is an approved drug for treating a deficiency in Vitamin B1. Let me start by understanding what Vitamin B1 is. I know that it's also called thiamine and it's one of the B vitamins. It plays a role in energy production and nerve function. Now, when someone has a deficiency in Vitamin B1, they might experience symptoms like fatigue, weight loss, and neurological problems. So, treating this deficiency would logically involve replenishing the vitamin that's lacking. I remember that vitamins are essential nutrients, meaning our bodies can't produce them in sufficient quantities, so we need to get them from our diet or supplements. Vitamin B1 is commonly found in foods like whole grains, meat, and fish. But the question here is about whether Vitamin B1 is an approved drug for its deficiency. I think that vitamins aren't classified as drugs per se; rather, they're considered dietary supplements. However, when used to treat a deficiency, especially under medical supervision, they can be prescribed or recommended by healthcare professionals. In some cases, severe deficiencies might require more than just dietary changes. For example, if someone has a condition where their body isn't absorbing nutrients properly, like in chronic alcoholism which affects Vitamin B1 absorption, they might need supplementation. I should check if there are any official guidelines or approvals from health authorities regarding this. Organizations like the FDA regulate drugs and supplements differently. Drugs require approval for specific uses, while supplements are regulated under different laws with less stringent requirements. Vitamin B1, as a supplement, is generally recognized as safe (GRAS) by the FDA when used appropriately. Medical professionals can recommend it to treat deficiencies without it needing to be approved as a drug in the traditional sense. So putting it all together: Vitamin B1 isn't classified as a drug but is an essential nutrient. Treating a deficiency would involve ensuring adequate intake, either through diet or supplements. Since it's not a drug, it doesn't go through the same approval process as medications do. However, healthcare providers can prescribe or recommend its use when necessary. Therefore, I think the answer is Yes because while it's not a drug per se, it's an essential nutrient that's approved for use in treating deficiencies.

Ragionamento 1. *Estratto dal file "risultati32B.CSV", ragionamento di una risposta vera positiva.*

Ramosetron	Alopecia	N/A	False	False

Figura 10. *Risposta realmente negativa, ossia dove la risposta di DeepSeek combacia con la realtà dei dati clinici ed entrambi concordo con 'No'.*

Okay, I need to determine whether Ramosetron is an approved drug for Alopecia. First, I should understand what Ramosetron is. From my knowledge, Ramosetron (also known as Ro-51-6384) is a selective 5-HT2B receptor agonist that's primarily used in the treatment of chemotherapy-induced nausea and vomiting. It works by affecting serotonin receptors in the brain. Now, looking at Alopecia, it refers to hair loss, which can be due to various causes like genetics, hormonal imbalances, stress, or certain medical conditions. There are different treatments available for Alopecia, such as topical solutions (like minoxidil), oral medications (like finasteride), and sometimes biotin supplements or other vitamins. I need to check if Ramosetron is approved by regulatory bodies like the FDA for treating Alopecia. I recall that Ramosetron's main indication is related to nausea and vomiting, particularly in chemotherapy patients. I don't remember it being used for hair loss conditions. Additionally, when considering drug approvals, it's important to look at clinical trials and studies that have been conducted on Ramosetron. If there are no significant studies showing its efficacy and safety for Alopecia, then it wouldn't be approved for that use. Furthermore, consulting reputable sources or the FDA website would confirm whether Ramosetron has any approved indications beyond its primaryÎ£Since I don't have access to real-time databases, I can infer based on existing knowledge that Ramosetron isn't listed as an approved treatment for Alopecia. In summary, after reviewing what I know about Ramosetron and its uses, it doesn't seem to be approved for treating Alopecia.

Ragionamento 2. *Estratto dal file "risultati32B.CSV", ragionamento di una risposta vera negativa.*

Octreotide	Hemorrhage	N/A	False	True

Figura 11. *Risposta falsa positiva, ovvero quando DeepSeek risponde 'Yes' e la risposta reale è 'No'.*

Okay, so I'm trying to figure out whether octreotide is an approved drug for hemorrhage. First, I should probably understand what octreotide is. From what I remember, it's a medication used in various medical conditions. I think it's related to hormones because it has something to do with somatostatin. Maybe it's used in managing hormone-related disorders or tumors.Now, the question is about its approval for hemorrhage. Hemorrhage refers to excessive bleeding, which can be internal or external. So, I need to check if octreotide is used specifically to treat bleeding. I recall that somatostatin analogs like octreotide are sometimes used in gastrointestinal conditions. For instance, they might help reduce blood flow in certain situations. Wait, there's something about variceal hemorrhage. Varices are enlarged veins, often in the esophagus or stomach, and they can bleed. I think that's where somatostatin comes into play. Octreotide could be used to treat bleeding from varices because it helps reduce portal hypertension, which is a cause of such bleeding. But is octreotide specifically approved for this purpose? I'm not entirely sure about the FDA approval status. Maybe I should look up if it's an indication listed on the drug label. Alternatively, it might be used off-label in some cases where other treatments aren't effective. Also, I know that in the past, somatostatin was used for acute variceal bleeding, but octreotide is a longer-acting version. So perhaps it's approved for that use. It would make sense because managing portal pressure can help stop the bleeding. I'm also thinking about other types of hemorrhage, like traumatic or post-surgical bleeding. I don't recall octreotide being used for those situations. It seems more specialized to conditions where hormonal regulation plays a role. So putting it all together: Octreotide is likely approved for treating variceal hemorrhage, which is a specific type of gastrointestinal bleeding. Therefore, the answer would be yes, but maybe only in certain contexts related to portal hypertension or varices.

Ragionamento 3. *Estratto dal file "risultati32B.CSV", ragionamento di una risposta falsa positiva.*

Undecylenic acid	Dermatitis	Approved	True	False

Figura 12. *Risposta falsa negativa, ovvero quando DeepSeek risponde 'No' e la risposta reale è 'Yes'.*

Okay, so I need to figure out whether Undecylenic acid is an approved drug for treating dermatitis. Let me start by understanding what Undecylenic acid is. From what I remember, it's a naturally occurring fatty acid that's often used in skincare products because of its anti-inflammatory and antifungal properties. Now, Dermatitis refers to inflammation of the skin, which can have various causes like eczema, contact dermatitis, or other conditions. The treatment for dermatitis usually depends on the type and severity. Common treatments include corticosteroids, moisturizers, and in some cases, immunomodulators. I'm trying to recall if Undecylenic acid is specifically approved by regulatory bodies like the FDA for treating dermatitis. I know it's commonly found in over-the-counter products, especially those aimed at fungal infections like athlete's foot because of its antifungal properties. However, whether it's formally approved as a drug for dermatitis is something I'm not entirely sure about. I think Undecylenic acid might be more of a complementary ingredient rather than a primary treatment. It could help with symptoms by reducing inflammation and preventing infections, which are common in cases of dermatitis. But does that make it an approved drug for the condition itself? Or is it just used adjunctively? Maybe I should check some reliable sources or guidelines on skincare treatments. The American Academy of Dermatology might have information on this. From what I can gather, while Undecylenic acid is recognized for its benefits in skin health, official approval as a standalone treatment for dermatitis isn't clear to me. I also remember that for a substance to be approved as a drug, it needs to go through rigorous testing and meet specific criteria set by regulatory agencies. Since Undecylenic acid is often found in OTC products, its role might be more about supporting skin barrier repair or reducing infection risk rather than directly treating the inflammation of dermatitis. In summary, while Undecylenic acid has properties that could be beneficial for someone with dermatitis, I'm not certain if it's specifically approved as a drug for that condition. It seems to be used more in supportive care or in products targeting related issues like fungal infections rather than being an officially sanctioned treatment.

Ragionamento 4. *Estratto dal file "risultati32B.CSV", ragionamento di una risposta falsa negativa. Il secondo programma, denominato Analisi_dati.py, permette di ottenere dei grafici a partire da un file esterno, che nel caso di questo studio sono stati i file CSV mostrati in precedenza. I grafici ottenuti, insieme alle statistiche e alle matrici di confusione di ogni modello, saranno mostrati nel prossimo paragrafo come risultati dello studio.*

```
import csv
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import r2_score
import seaborn as sns

def analizza_ragionamenti_da_csv(csv_file="risultati 32b.csv"):
    lunghezze = {'Si': [], 'No': []}

    with open(csv_file, 'r', encoding='utf-8') as file:
        for row in csv.DictReader(file):
            lunghezza = len(row.get('ragionamento', ''))
            categoria = 'Si' if row['risposta_reale'].lower()
            == 'true' else 'No'
            lunghezze[categoria].append(lunghezza)

    #Boxplot
    fig, ax = plt.subplots(figsize=(10, 6))
    data = [lunghezze['Si'], lunghezze['No']]
    bp = ax.boxplot(data, labels=['Yes', 'No'], patch_artist=True)

    #Colori boxplot
    colors = ['#1062A5', '#C61616']
    for patch, color in zip(bp['boxes'], colors):
        patch.set_facecolor(color)
        patch.set_alpha(0.8)

    ax.set_title('Lunghezza ragionamenti 32B', fontsize=14, fontweight='bold')
    ax.set_xlabel('Tipo di risposta', fontsize=12)
    ax.set_ylabel('Lunghezza caratteri', fontsize=12)
    ax.grid(True, alpha=0.3)
    plt.tight_layout()
    plt.show()

    return lunghezze['Si'], lunghezze['No']

def crea_diagramma_torta_stati(file="P1-05-Drug_disease.txt"):
    stati_count = {}

    #Conteggio stati
    with open(file) as f:
        for line in f:
            if "INDICATI" in line:
                stato = line.split("\t")[3].strip().lower()
                stati_count[stato] = stati_count.get(stato, 0) + 1

    #Raggruppamento intelligente
    total = sum(stati_count.values())
    soglia = max(10, total * 0.02)
    stati_nascosti = ['discontinued in phase 1', 'discontinued in phase 2']

    stati_principali = {}
    altri_count = 0

    for stato, count in stati_count.items():
        if stato in stati_nascosti or count < soglia:
            altri_count += count
        else:
            stati_principali[stato] = count
```

```
    if altri_count > 0:
        stati_principali["Altri"] = altri_count

    #Ordina per dimensione
    stati_ordinati = dict(sorted(stati_principali.items(),
                                 key=lambda x: x[1],
                                 reverse=True))

    #Diagramma a torta
    labels = [stato.capitalize() for stato in stati_ordinati.keys()]
    sizes = list(stati_ordinati.values())
    colors = ['#FF4444', '#1E90FF', '#32CD32', '#FF8C00', '#8B008B',
              '#FF1493', '#00CED1', '#FFD700', "#BE1717", '#4169E1']

    fig, ax = plt.subplots(figsize=(12, 8))
    wedges, texts, autotexts = ax.pie(sizes, labels=labels,
                                      colors=colors[:len(labels)],
                                      autopct='%1.1f%%',
                                      startangle=90,
                                      textprops={'fontsize': 10})

    for autotext in autotexts:
        autotext.set_color('white')
        autotext.set_fontweight('bold')
        autotext.set_fontsize(11)

    ax.set_title('Distribuzione degli Stati', fontsize=16,
                 fontweight='bold', pad=20)
    ax.axis('equal')

    #Legenda
    ax.legend(wedges, [f'{label}: {size}' for label,
              size in zip(labels, sizes)],
              title="Stati", loc="center left",
              bbox_to_anchor=(1, 0, 0.5, 1))

    plt.tight_layout()
    plt.show()

    #Report compatto
    print("Distribuzione degli stati:")
    for stato, count in stati_ordinati.items():
        percentage = (count / total) * 100
        print(f"  {stato.capitalize()}: {count} ({percentage:.1f}%)")

    if altri_count > 0:
        print(f"\nTotale stati in 'Altri': {altri_count} occorrenze
({(altri_count/total)*100:.1f}%)")

def crea_diagramma_a_barre(dati_70b):
    #Diagramma a barre moderno con predizione 70B evidenziata
    plt.style.use('seaborn-v0_8-whitegrid')
    sns.set_palette("husl")

    #Dati organizzati
    metriche = ['Sensibilità', 'Precisione', 'F1 Score', 'Risposte Corrette']
    dati = {
        '1.5B': [26.00, 61.90, 37, 55.00],
        '7B':   [47.00, 74.13, 58, 65.30],
        '32B':  [59.20, 96.73, 73, 78.60],
        '70B':  [dati_70b['sensibilita'], dati_70b['precisione'],
                 dati_70b['f1_score'], dati_70b['risposte_corrette']]
    }
```

```
    #Setup grafico
    fig, ax = plt.subplots(figsize=(14, 8))
    x = np.arange(len(metriche))
    width = 0.2
    colori = ['#C61616', '#228B22', '#1062A5', '#f39c12']

    #Crea barre
    bars = []
    for i, (modello, valori) in enumerate(dati.items()):
        is_pred = (modello == '70B')
        bars.append(ax.bar(x + i*width - 1.5*width, valori, width,
                           label=f'{modello}{" (Predetto)" if is_pred else ""}',
                           color=colori[i], alpha=0.9,
                           edgecolor='darkred' if is_pred else 'black',
                           linewidth=2 if is_pred else 1,
                           hatch='///' if is_pred else None))

    #Personalizzazione
    ax.set_title('Confronto Performance LLM',
                 fontsize=18, fontweight='bold', pad=20)
    ax.set_xlabel('Metriche', fontsize=14, fontweight='bold')
    ax.set_ylabel('Performance (%)', fontsize=14, fontweight='bold')
    ax.set_xticks(x)
    ax.set_xticklabels(metriche, fontsize=12)
    ax.legend(title='Modelli', fontsize=12, title_fontsize=13,
              loc='upper left', frameon=True, shadow=True)
    ax.grid(axis='y', alpha=0.3, linestyle='-', linewidth=0.5)

    #Annotazioni sui valori
    for i, (modello, valori) in enumerate(dati.items()):
        for j, valore in enumerate(valori):
            if valore > 0:
                bar = bars[i][j]
                height = bar.get_height()
                color = 'red' if modello == '70B' else 'black'
                weight = 'bold' if modello == '70B' else 'normal'
                ax.text(bar.get_x() + bar.get_width()/2., height + 1,
                        f'{valore:.1f}%', ha='center', va='bottom',
                        fontsize=10, color=color, fontweight=weight)

    #Nota predizione
    ax.text(0.02, 0.98, '/// = Predizione',
            transform=ax.transAxes, fontsize=11,
            verticalalignment='top', bbox=dict(boxstyle='round',
            facecolor='wheat', alpha=0.7))

    #Limiti y eleganti
    max_val = max([max(valori) for valori in dati.values()])
    ax.set_ylim(0, max_val * 1.15)

    plt.tight_layout()
    plt.show()

def predici_70b_e_crea_grafici():
    #Predice performance 70B con modelli avanzati e visualizzazioni eleganti
    plt.style.use('seaborn-v0_8-whitegrid')
    sns.set_palette("husl")

    #Dati performance
    X = np.array([1.5, 7, 32]).reshape(-1, 1)
    metriche = {
        'Sensibilità': np.array([26.00, 47.00, 59.20]),
        'Precisione': np.array([61.90, 74.13, 96.73]),
        'F1 Score': np.array([37.0, 58.0, 73.0]),
        'Risposte Corrette': np.array([55.00, 65.30, 78.60])
    }
```

```
def predici_metrica(X, y, metrica_nome):
    #Seleziona automaticamente il miglior modello di predizione
    X_70 = np.array([[70]])
    modelli = []


    #Polinomiale
    poly = Pipeline([('poly', PolynomialFeatures(degree=2)),
                     ('lr', LinearRegression())])
    poly.fit(X, y)
    poly_pred = poly.predict(X_70)[0]
    poly_score = r2_score(y, poly.predict(X))
    modelli.append(('Polinomiale', poly_pred, poly_score))

    #Log-lineare
    log_model = LinearRegression()
    log_model.fit(np.log(X), y)
    log_pred = log_model.predict(np.log(X_70))[0]
    log_score = r2_score(y, log_model.predict(np.log(X)))
    modelli.append(('Log-lineare', log_pred, log_score))

    #Saturazione per precisione
    if metrica_nome == "Precisione":
        sat_pred = min(y[-1] + (99.5 - y[-1]) / 20, 99.5)
        modelli.append(('Saturazione', sat_pred, 0.95))

    #Seleziona migliore (valori validi 0-100%)
    modelli_validi = [(n, p, s) for n, p, s in modelli if 0 <= p <= 100]

    #Per precisione, preferisci saturazione
    if metrica_nome == "Precisione":
        saturazione = [m for m in modelli_validi if m[0] == 'Saturazione']
        if saturazione:
            return saturazione[0][1], saturazione[0][0]

    #Altrimenti scegli per R² migliore
    migliore = max(modelli_validi, key=lambda x: x[2])
    return migliore[1], migliore[0]

#Predizioni
predizioni, metodi = {}, {}
for metrica, valori in metriche.items():
    pred, metodo = predici_metrica(X, valori, metrica)
    predizioni[metrica] = np.clip(pred, 0, 99.5
                          if metrica == "Precisione" else 95)
    metodi[metrica] = metodo
    print(f"{metrica}: {predizioni[metrica]:.2f}% ({metodo})")

#Visualizzazione moderna
fig, axes = plt.subplots(2, 2, figsize=(16, 12))
axes = axes.flatten()
colori = ['#C61616', '#228B22', '#1062A5', '#f39c12']
dimensioni = [1.5, 7, 32, 70]
etichette = ['1.5B', '7B', '32B', '70B']

for i, (metrica, valori_orig) in enumerate(metriche.items()):
    ax = axes[i]
    valori_completi = np.append(valori_orig, predizioni[metrica])

    #Plot dati e predizione
    ax.scatter(dimensioni[:3], valori_orig, s=120, c=colori[i], alpha=0.8,
               label='Performance reale', zorder=3)
    ax.scatter([70], [predizioni[metrica]], s=200, c='red', marker='^',
               alpha=0.9, label='Performance estrapolata',
               edgecolors='darkred', linewidth=2, zorder=4)
    ax.plot(dimensioni, valori_completi, '--', color=colori[i],
            alpha=0.6, linewidth=2)
```

```
        #Personalizzazione
        ax.set_title(f'{metrica}', fontsize=16, fontweight='bold', pad=20)
        ax.set_xlabel('Dimensione Modello', fontsize=12)
        ax.set_ylabel(f'{metrica} (%)', fontsize=12)
        ax.grid(True, alpha=0.3)
        ax.legend(loc='best', fontsize=10)
        ax.set_xscale('log')
        ax.set_xticks(dimensioni)
        ax.set_xticklabels(etichette)

        #Annotazioni eleganti
        for j, (x, y) in enumerate(zip(dimensioni, valori_completi)):
            offset, color, weight = (15, 'black', 'normal') if j < 3
            else (20, 'red', 'bold')
            ax.annotate(f'{y:.1f}%', (x, y), xytext=(0, offset),
                        textcoords='offset points', ha='center',
                        fontsize=11, color=color, fontweight=weight)

        #Limiti y
        y_min, y_max = min(valori_completi) * 0.9,
                        min(max(valori_completi) * 1.1, 100)
        ax.set_ylim(y_min, y_max)

    plt.tight_layout()
    plt.suptitle('Performance LLM',
                 fontsize=18, fontweight='bold', y=1.02)
    plt.show()

    return {
        'sensibilita': predizioni['Sensibilità'],
        'precisione': predizioni['Precisione'],
        'f1_score': predizioni['F1 Score'],
        'risposte_corrette': predizioni['Risposte Corrette']
    }

if __name__ == "__main__":
    print("="*50)
    print("ANALISI PREDITTIVA MODELLO 70B")
    print("="*50)
    predizioni_70b = predici_70b_e_crea_grafici()

    #Esegui analisi complete
    analizza_ragionamenti_da_csv("risultati 32b.csv")
    crea_diagramma_torta_stati("P1-05-Drug_disease.txt")
    crea_diagramma_a_barre(predizioni_70b)
```

Algoritmo 2. *Analisi_dati.py, script utilizzato per realizzare i grafici.*

4.3 RISULTATI

Lo strumento principale utilizzato per riassumere i risultati di questa analisi è la *matrice di confusione (*vedi ***Tabelle 3,5,7)***, una tabella dalle dimensioni 2x2, utilizzata in ambito di machine learning (ML) e statistica per mettere in relazione le risposte corrette e le predizioni del modello. Da questa matrice è possibile calcolare i parametri statistici citati nel paragrafo precedente: la percentuale di *risposte corrette*, ossia l'accuratezza, la *precisione*, la *sensibilità*, l'*F1 Score*. L'*accuratezza* (vedi ***Grafico 5***) rappresenta quante volte il modello risponde correttamente, non importa che sia vero positivo o vero negativo, e si calcola facendo il rapporto tra la somma dei veri positivi (TP) e dei veri negativi (TN) sulla somma dei veri con i falsi positivi (FP) e falsi negativi (FN).

$$Accuratezza = \frac{TP + TN}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Il secondo parametro, ossia la *precisione* (vedi ***Grafico 3***), permette di capire quante volte il modello rispondendo positivamente, quindi 'Yes', lo ha fatto correttamente e quante volte invece ha dato dei falsi positivi. Si calcola facendo il rapporto tra i veri positivi e la somma tra i veri positivi e i falsi positivi.

$$Precisione = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

Poi abbiamo la *sensibilità* (vedi ***Grafico 2***), che permette di capire quanti dei veri positivi siano stati trovati dal modello, in pratica su tutti gli 'Yes' presenti quanti ne ha trovati il modello. Questo parametro si calcola facendo il rapporto tra i veri positivi e la somma dei veri positivi con i falsi negativi.

$$Sensibilità = \frac{TP}{\mathrm{TP} + \mathrm{FN}}$$

Infine, c'è l'*F1 Score* (vedi ***Grafico 4***), questo è un punteggio che rappresenta la media armonica tra sensibilità e precisione. Questo parametro è importante quando sia precisione che sensibilità sono importanti o quando i dati sonno sbilanciati, ad esempio con 95% positivi e 5% negativi. Si usa la media armonica poiché essa, a differenza di quella aritmetica, non assegna lo stesso peso ad ogni valore, se precisione o sensibilità sono bassi anche l'F1 score risulterà basso. Lo si calcola facendo il doppio del rapporto tra il prodotto della sensibilità e della precisione per la loro somma.

$$F1\ Score = 2 * \frac{\text{Precisione}*\text{Sensibilità}}{\text{Precisione}+\text{Sensibilità}} = 2 * \frac{2\text{TP}}{2\text{TP}+\text{FN}+\text{FP}}$$

Di seguito sono riportate le matrici di confusione, le statistiche per ogni modello (vedi ***Tabelle 4,6,8***) e un grafico a barre (vedi ***Grafico 6***) di comparazione.

Matrice 1.5B	Predetto positivo	Predetto negativo
Realmente positivo	130	370
Realmente negativo	80	420

Tabella 3. *Matrice di confusione del modello 1.5 B, si può notare come il modello presenti numeri elevati di falsi positivi (80) e di falsi negativi (370). Al contrario i veri positivi sono meno della metà dei falsi positivi, ossia (130).*

Statistiche 1.5B	
Risposte corrette	55%
Precisione	61.90%
Sensibilità	26%
F1 Score	37%
P-value Fisher	0.00013482620484274495

Tabella 4. *Statistiche del modello 1.5B, i valori vengono riportati tutti come percentuali, anche quelli delle risposte corrette e dell'F1 score che normalmente si troverebbero, in ordine, intero o decimale. Il p-value di Fisher è riportato per intero poiché se troncato prima risulterebbe essere 0.*

Matrice 7B	Predetto positivo	Predetto negativo
Realmente positivo	235	265
Realmente negativo	82	418

Tabella 5. *Matrice di confusione del modello 7B, qui si può notare come siano aumentati veri positivi (235), con relativa diminuzione dei falsi negativi (265) rispetto al modello precedente. Invece, i reali negativi e i falsi positivi sono rimasti sostanzialmente invariati.*

Statistiche 7B	
Risposte corrette	65.30%
Precisione	74.13%
Sensibilità	47%
F1 Score	58%
P-value Fisher	9.551512776605723e-26

Tabella 6. *Statistiche del modello 7B, il p-value è espresso con un esponenziale poiché è troppo piccolo.*

Matrice 32B	Predetto positivo	Predetto negativo
Realmente positivo	296	204
Realmente negativo	10	490

Tabella 7. *Matrice di confusione del modello 32B, i veri positivi sono un numero elevato (296) anche più alto rispetto ai falsi negativi (204), mentre i veri negativi sono aumentati rispetto agli altri modelli (490) e i falsi positivi sono drasticamente diminuiti (10).*

Statistiche 32B	
Risposte corrette	78.60%
Precisione	96.73%
Sensibilità	59.20%
F1 Score	73%
P-value Fisher	1.4877890966188543e-100

Tabella 8. *Statistiche del modello 32B, il p-value è espresso con un esponenziale poiché è troppo piccolo.*

Sono state fatte delle stime dei parametri sopracitati relative la modello 70B, che non è stato utilizzato poiché richiedeva hardware più potenti di quelli a nostra disposizione, in particolare di memoria GPU. Per stimare le performance del modello 70B lo script analizza i dati reali osservati per i modelli più piccoli, ovvero 1.5B, 7B e 32B. Per ciascuna metrica, cioè *sensibilità*, *precisione*, *F1 Score*, e *accuratezza* si confrontano tre modelli di regressione: la *regressione polinomiale di secondo grado*, che adatta una curva quadratica ai dati per catturare trend non lineari; la *regressione log-lineare*, che usa il logaritmo della dimensione del modello per rappresentare una crescita decrescente; e un *modello di saturazione* (solo per la precisione), che impone un limite massimo realistico per evitare stime sovrastimate. Ogni modello viene valutato tramite il *coefficiente di determinazione* R^2, che misura la qualità dell'adattamento del modello di regressione ai dati noti. La previsione finale per il 70B viene scelta dal modello con R^2 più alto, salvo per la Precisione, dove si privilegia il modello di saturazione per garantire stime realistiche. Questo approccio consente di ottenere stime più accurate e coerenti sull'evoluzione delle performance in relazione alla scala del modello.

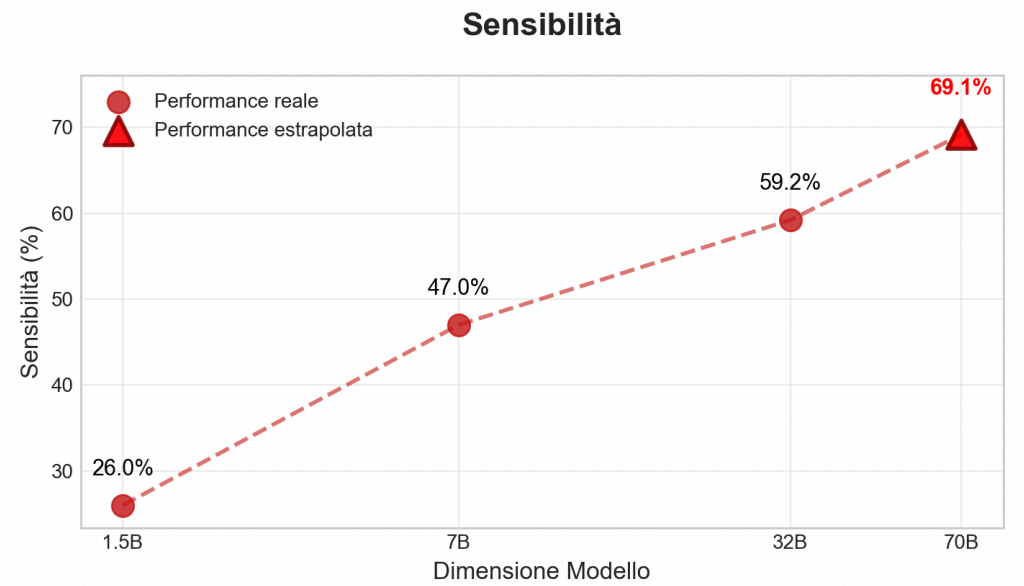


Grafico 2. *Grafico con predizione della sensibilità, ossia il parametro che permette di capire su quanti positivi abbia trovato il modello sulla loro totalità.*

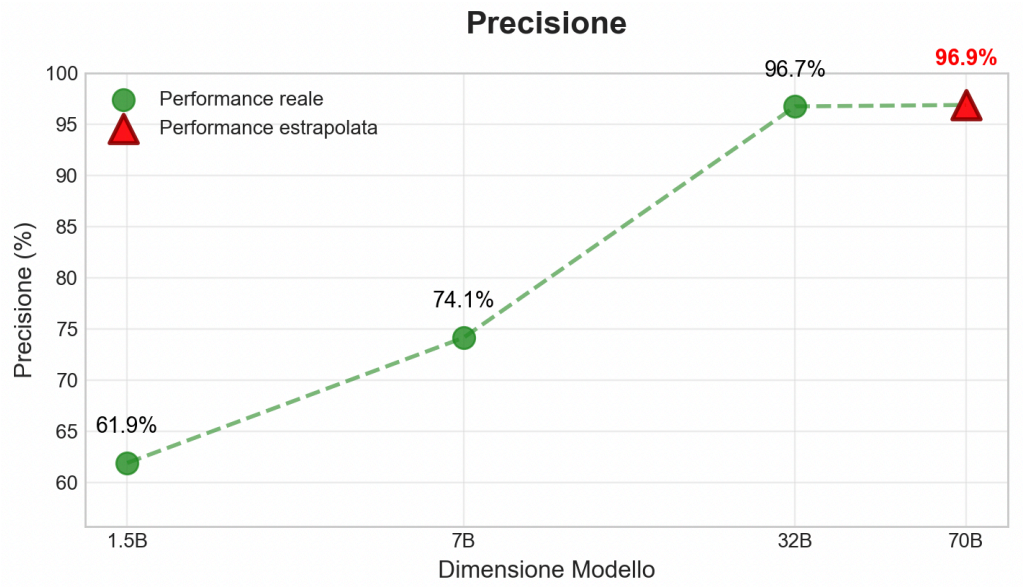


Grafico 3. *Grafico con predizione della precisione, ossia il parametro che permette di capire quanti positivi siano reali sulla totalità di quelli trovati dal modello.*

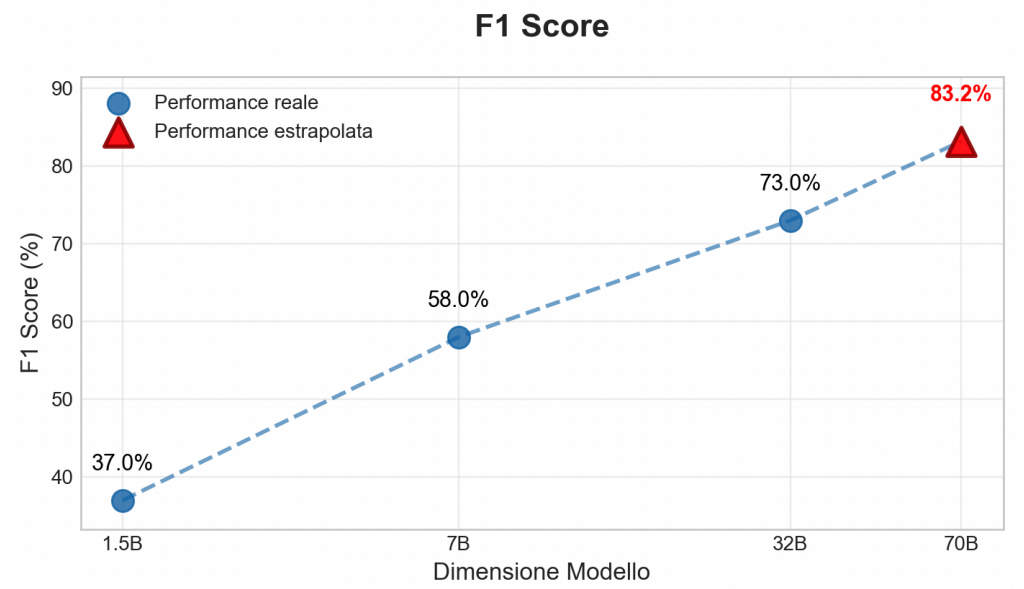


Grafico 4. *Grafico con predizione dell'F1 score, ossia il parametro che permette unisce sensibilità e precisione facendone la media armonica.*

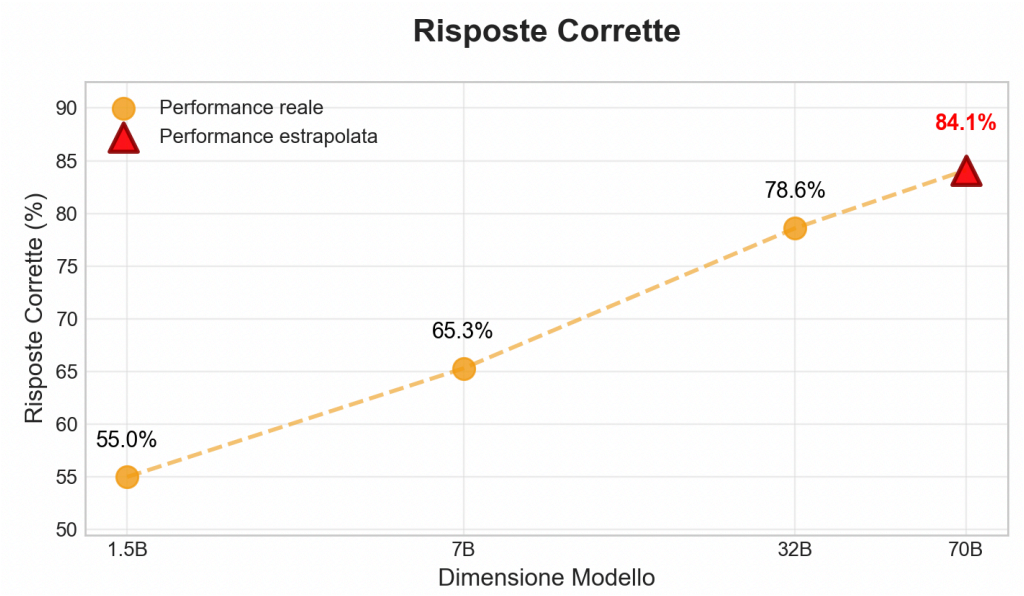


Grafico 5. *Grafico con predizione delle risposte corrette, ossia il parametro che permette di capire l'accuratezza del modello.*

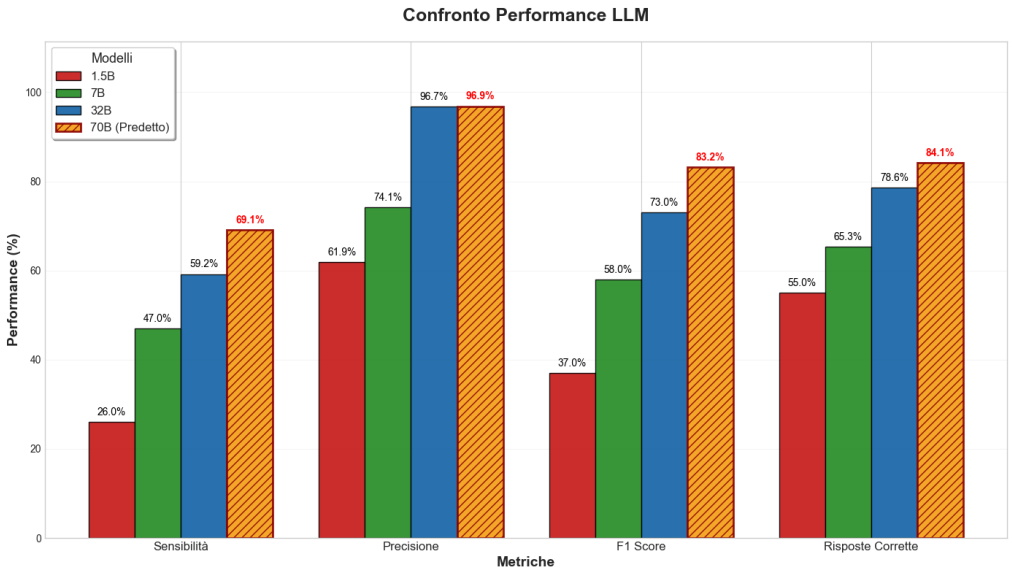


Grafico 6. *Diagramma a barre utile per una comparazione chiara delle performance tra i modelli.*

Dalle statistiche si è notato che con l'aumentare della potenza del modello si è avuto anche un incremento delle performance, in particolare il 32B presenta dei dati nettamente migliori rispetto agli altri modelli, e ciò è visibile anche dalle matrici di confusione. Proprio da quest'ultime si può notare come il modello più potente abbia una precisione vicina al 100%, presentando su 306 positivi totali, 290 reali e solo 10 falsi. Per quanto riguarda i negativi, su un totale di 694 troviamo 490 reali e 204 falsi, ciò implica che circa il 70% dei 'No' che il modello restituisce sono veri. Questo dato ha portato a fare ulteriori analisi sui ragionamenti che l'IA esegue per rispondere alla domanda, in particolare è stata analizzata la lunghezza, sia per le risposte 'Yes', sia per le risposte 'No'. Per fare ciò sono stati salvati i ragionamenti nel file CSV esterno, per poi analizzarli tramite lo script *Analisi_dati.py* ottenendo dei boxplot (vedi ***Grafici 7,8,9***), grafici utili per visualizzare la distribuzione dei dati.

All’interno dei boxplot ogni *box* è diviso in *quartili* (Q) che vanno da 0 a 4, dove Q0 corrisponde all’estremo del baffo inferiore, Q1 è l’inizio del box, Q2 è la mediana, Q3 è la fine del box e Q4 è l’estremo del baffo superiore. Per ogni modello preso in esame sono stati generati due boxplot, uno per la lunghezza dei ragionamenti delle risposte ‘Yes’ e uno per la lunghezza dei ragionamenti delle risposte ‘No’.

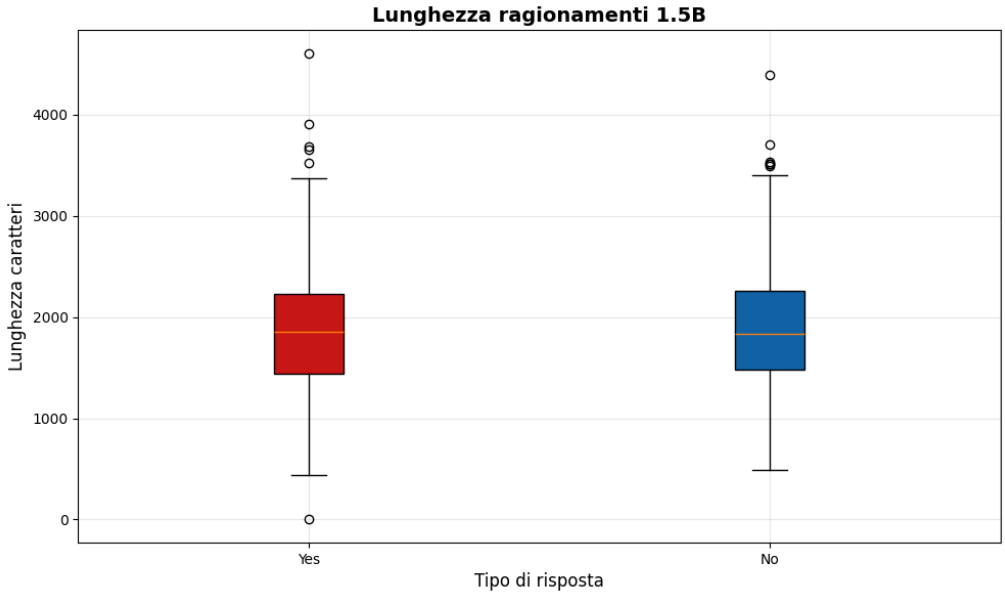


Grafico 7. *Boxplot 1.5B, i box sono alla stessa altezza e ciò indica che la variabilità della lunghezza dei ragionamenti è simile. Anche la mediana è simile, si aggira attorno ai 1800 caratteri ed entrambi presentano valori anomali alti/molto alti, ma solo in ‘Yes’ troviamo un valore prossimo allo 0. L’altra differenza è data dalla lunghezza dei baffi, che nei no sono più corti nel boxplot dei ‘No’.*

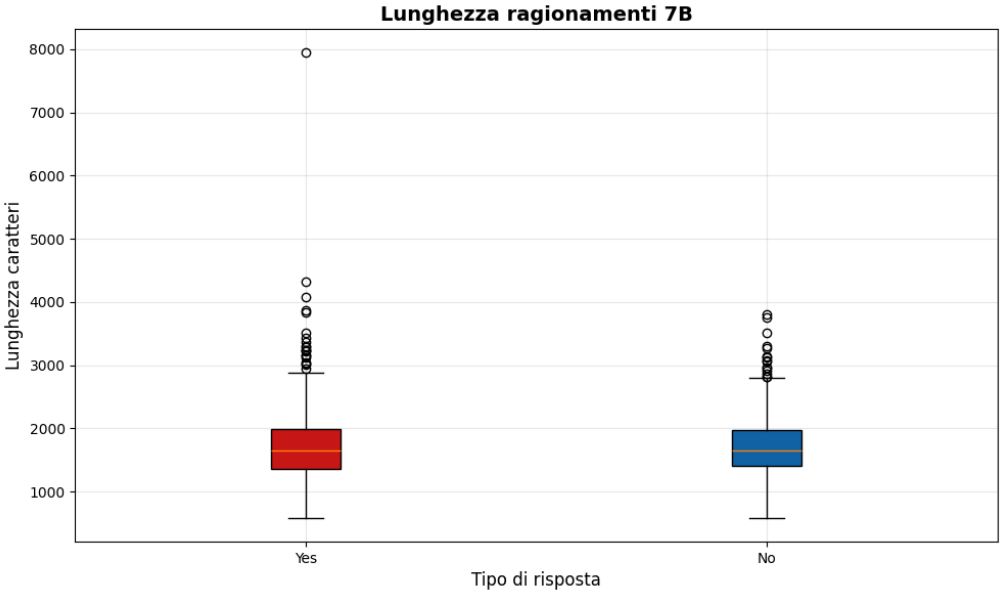


Grafico 8. *Boxplot 7B, i box sono alla stessa altezza e ciò indica che la variabilità della lunghezza dei ragionamenti è simile. La mediana è simile, si aggira attorno ai 1700/1800 caratteri , entrambi presentano valori anomali tutti alti/molto alti, ma solo nelle risposte 'Yes' è presente un valore prossimo ad 8000. I baffi inferiori sono uguali per entrambi i grafici, mentre il baffo superiore del box delle risposte 'No' è un po' più basso rispetto all'altro.*

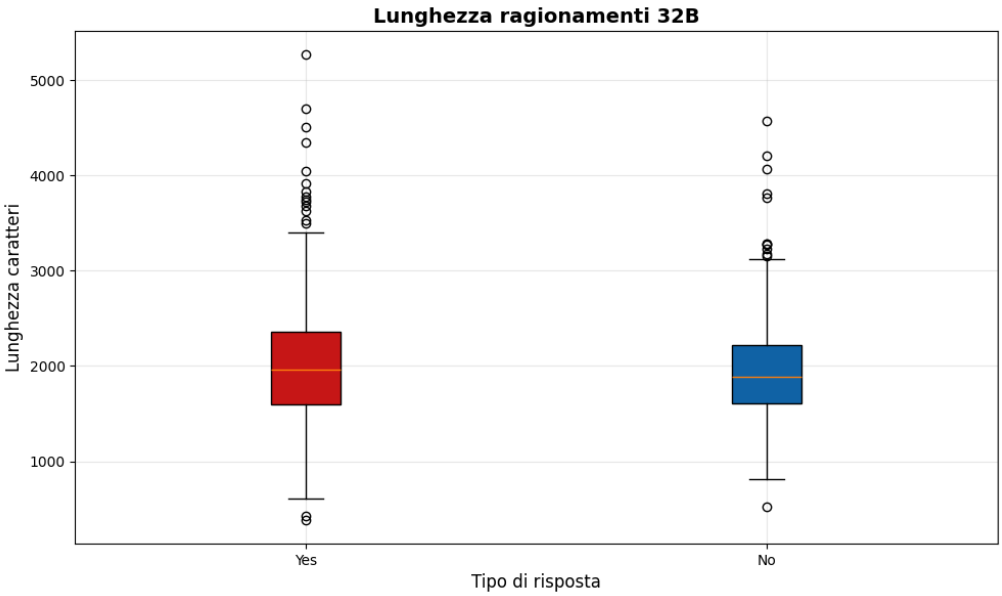


Grafico 9. *Boxplot 32B, i box hanno grandezze diverse e ciò indica che le risposte 'No' hanno una lunghezza minore, in generale, rispetto alle risposte 'Yes'. Entrambi presentano valori anomali sia alti che bassi, ma solo nelle risposte 'Yes' troviamo un valore oltre i 5000, in più i baffi di 'No' sono più corti rispetto a 'Yes'.*

Dalle matrici di confusione si può notare come i primi due modelli presentano un numero elevato di falsi positivi e falsi negativi, numero che invece diminuisce nel 32B e che, presumibilmente, diminuisce ancora nel 70B. Dalle statistiche sono stati ottenuti i p-value per ogni modello e ognuno di essi è risultato essere minore di 0.05, ossia il valore soglia per rigettare l'ipotesi nulla, che in questo caso è rappresentata dall'affidabilità delle risposte del modello. Non è stato calcolato il p-value del 70B ma a partire dai valori stimati si può presupporre che anch'esso avrebbe un p-value al di sotto del valore soglia. Sono state analizzate le lunghezze dei ragionamenti sia delle risposte 'Yes' che delle risposte 'No' per verificare se quest'ultime fossero più lunghe in caso di veri negativi e dar vita quindi a fenomeni di allucinazione. Per quanto riguarda i grafici (vedi ***Grafici 7,8,9***) si può notare che tutti e tre presentano una distribuzione delle lunghezze simile tra i due boxplot, presenti per ogni modello. Il primo presenta dei box simili con solo alcuni valori anomali, tra cui solo uno prossimo allo 0 per i valori del 'Yes', il secondo anche presenta delle distribuzioni simili con valori anomali, ma in questo caso soltanto positivi e uno solo oltre i 5000 caratteri. Il terzo ha distribuzioni leggermente diverse con la presenza di alcuni valori anomali, in questo caso però sono presenti diversi valori anomali, sia positivi sia prossimi allo 0. Dallo studio sono stati eliminati casi particolari tra gli stati, come ad esempio tutti gli altri stati *approved*, i *withdrawn from market* o i *discontinued*. Per questo motivo c'è bisogno di effettuare altri test e fare le statistiche considerando questi casi particolari. Per il futuro ci si aspetta che questo studio possa essere ampliato con le casistiche citate sopracitate.

4.4 CONCLUSIONI

Sulla base di quanto detto nel paragrafo precedente si può concludere che tutti e tre i modelli presi in esame hanno una buona conoscenza delle indicazioni terapeutiche delle malattie e che esiste una proporzionalità diretta tra la potenza del modello e la sua conoscenza. Dalle stime effettuate sul modello 70B si è notato che i modelli più grandi e potenti hanno una conoscenza più accurata e ci si aspetta, in futuro, di avere modelli sempre più potenti e accurati. Dai dati si evince anche che tutti i modelli tendono a fare più fatica nel rispondere a domande negative rispetto a quelle positive, ma c'è la possibilità che tra questi ci siano dei reali positivi che potrebbero diventare in futuro delle possibilità da seguire per trovare nuove soluzioni. Per il futuro si spera che le IA saranno in grado di aiutare ancora di più, risultando uno strumento di utilizzo quotidiano in tutti gli ambiti e che potrà supportare in decisioni cruciali come quella della diagnosi, potendo prevedere degli eventi avversi portati da un farmaco, poter capire con precisione quale farmaco funziona meglio su un paziente in base al suo profilo genetico oppure filtrare in tempo reale migliaia di paper per poter trovare dei pattern emergenti.

BIBLIOGRAFIA

1. Treccani. «Progettazione dei farmaci - Enciclopedia». https://www.treccani.it/enciclopedia/progettazione-dei-farmaci_(XXI-Secolo)/.
2. Pina, Ana Sofia, Abid Hussain, e Ana Cecília A. Roque. «An Historical Overview of Drug Discovery». *Methods in Molecular Biology (Clifton, N.J.)* 572 (2009): 3–12. https://doi.org/10.1007/978-1-60761-244-5_1.
3. Gupta, Rohan, Devesh Srivastava, Mehar Sahu, Swati Tiwari, Rashmi K. Ambasta, e Pravir Kumar. «Artificial intelligence to deep learning: machine intelligence approach for drug discovery». *Molecular Diversity* 25, fasc. 3 (2021): 1315–60. https://doi.org/10.1007/s11030-021-10217-3.
4. Vamathevan, Jessica, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, et al. «Applications of Machine Learning in Drug Discovery and Development». *Nature Reviews. Drug Discovery* 18, fasc. 6 (giugno 2019): 463. https://doi.org/10.1038/s41573-019-0024-5.
5. Lin, Zeming, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, et al. «Evolutionary-scale prediction of atomic-level protein structure with a language model». *Science* 379, fasc. 6637 (17 marzo 2023): 1123–30. https://doi.org/10.1126/science.ade2574.
6. Rintala, Teemu J, Francesco Napolitano, e Vittorio Fortino. «Multi-task deep latent spaces for cancer survival and drug sensitivity prediction». *Bioinformatics* 40, fasc. Supplement_2 (1° settembre 2024): ii182–89. https://doi.org/10.1093/bioinformatics/btae388.
7. Yang, Zhenyu, Xiaoxi Zeng, Yi Zhao, e Runsheng Chen. «AlphaFold2 and Its Applications in the Fields of Biology and Medicine». *Signal Transduction and Targeted Therapy* 8, fasc. 1 (14 marzo 2023): 115. https://doi.org/10.1038/s41392-023-01381-z.
8. Li, Zhongxiao, Antonella Napolitano, Monica Fedele, Xin Gao, e Francesco Napolitano. «AI identifies potent inducers of breast cancer stem cell differentiation based on adversarial learning from gene expression data». *Briefings in Bioinformatics* 25, fasc. 3 (1° maggio 2024): bbae207. https://doi.org/10.1093/bib/bbae207.

9. Napolitano, Francesco, Trisevgeni Rapakoulia, Patrizia Annunziata, Akira Hasegawa, Melissa Cardon, Sara Napolitano, Lorenzo Vaccaro, et al. «Automatic Identification of Small Molecules That Promote Cell Conversion and Reprogramming». *Stem Cell Reports* 16, fasc. 5 (11 maggio 2021): 1381–90. https://doi.org/10.1016/j.stemcr.2021.03.028.
10. «Dartmouth Workshop». In *Wikipedia*, 27 maggio 2025. https://en.wikipedia.org/w/index.php?title=Dartmouth_workshop&oldid=1292598192.
11. Rumelhart, David E., Geoffrey E. Hinton, e Ronald J. Williams. «Learning Representations by Back-Propagating Errors». *Nature* 323, fasc. 6088 (ottobre 1986): 533–36. https://doi.org/10.1038/323533a0.
12. Raman, Raghu, Robin Kowalski, Krishnashree Achuthan, Akshay Iyer, e Prema Nedungadi. «Navigating Artificial General Intelligence Development: Societal, Technological, Ethical, and Brain-Inspired Pathways». *Scientific Reports* 15, fasc. 1 (11 marzo 2025): 8443. https://doi.org/10.1038/s41598-025-92190-7.
13. Mienye, Ibomoiye Domor, e Theo G. Swart. «A Comprehensive Review of Deep Learning: Architectures, Recent Advances, and Applications». *Information* 15, fasc. 12 (dicembre 2024): 755. https://doi.org/10.3390/info15120755.
14. WhatIs. «DeepSeek Explained: Everything You Need to Know». https://www.techtarget.com/whatis/feature/DeepSeek-explained-Everything-you-need-to-know.
15. DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, et al. «DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning». arXiv, 22 gennaio 2025. https://doi.org/10.48550/arXiv.2501.12948.
16. DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, et al. «DeepSeek-V3 Technical Report». arXiv, 18 febbraio 2025. https://doi.org/10.48550/arXiv.2412.19437.

17. Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, e Illia Polosukhin. «Attention is All you Need». In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 2017. https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.
18. Raza, Mubashar, Zarmina Jahangir, Muhammad Bilal Riaz, Muhammad Jasim Saeed, e Muhammad Awais Sattar. «Industrial Applications of Large Language Models». *Scientific Reports* 15, fasc. 1 (21 aprile 2025): 13755. https://doi.org/10.1038/s41598-025-98483-1.
19. Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. «Language Models are Few-Shot Learners». In *Advances in Neural Information Processing Systems*, 33:1877–1901. Curran Associates, Inc., 2020. https://papers.nips.cc/paper_files/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.
20. ar5iv. «Large Language Models: A Survey». https://ar5iv.labs.arxiv.org/html/2402.06196.
21. Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, et al. «LLaMA: Open and Efficient Foundation Language Models». arXiv, 27 febbraio 2023. https://doi.org/10.48550/arXiv.2302.13971.
22. Zhou, Ying, Yintao Zhang, Donghai Zhao, Xinyuan Yu, Xinyi Shen, Yuan Zhou, Shanshan Wang, Yunqing Qiu, Yuzong Chen, e Feng Zhu. «TTD: Therapeutic Target Database describing target druggability information». *Nucleic Acids Research* 52, fasc. D1 (15 settembre 2023): D1465–77. https://doi.org/10.1093/nar/gkad751.